Computer Science Department of Polytech Tours

University of Sherbrooke Computer Science Department

Fifth year engineering internship report

Realised at Orange Labs

26 April 2021 - 31 August 2021

Feature engineering for regression using classification

Company
Orange Labs
2 Avenue Pierre Marzin
22300 Lannion

Student
Colin Troisemaine
colin.troisemaine@gmail.com

Company tutor
Vincent Lemaire
vincent.lemaire@orange.com
http://vincentlemaire-labs.fr/
Researcher

Academic tutor
Christophe Lenté
christophe.lente@univ-tours.fr
Research professor

Version of the July 6, 2023

Contents

List of Figures

Acknowledgement

This internship opportunity at Orange Labs Lannion was a wonderful occasion to improve my understanding of many aspects of machine learning and the skills necessary in the research process. I am also grateful to have had the chance to meet many brilliant individuals who have helped me during this internship period.

I would like to express my deepest and most sincere gratitude to my tutor : Vincent Lemaire for his motivation, his patience and the endless support he gave me throughout my internship. His mentorship greatly influenced my vision on the professional and research world. His optimism and advice encouraged me to pursue a doctorate after this internship. Vincent also proofread this report countless times, and his keen eye and wealth of experience has driven me to tremendously improve my writing proficiency. The availability he has shown was also very important to me in this particular pandemic context.

In addition to my tutor, I would like to thank Fabrice Clérot, the manager of the PROF team in which my internship took place. His insightful comments and plentiful ideas inspired me to broaden my research from various angles. His immense knowledge also supported me in some of the most difficult phases of my internship.

Finally, my appreciation also goes to my colleagues from the PROF team for the lively conversations we had together in the office and in social settings.

Index of notation

The following notations and definitions will be used:

Variable	Definition
y & y_i	The target variable & The target variable of an instance x_i
$\hat{y}_i$	Predicted value of x_i
$\bar{y}$	Mean value of y
X & x_i	A set of instances & An instance i of a X
β_i	Scalar weight (found in the linear regression equation)
$\beta_i(\tau)$	A given function of parameter τ
τ	Quantile level
$Q_\tau(Y\|X)$	Conditional quantile of Y given X
$Q_\tau(y_i)$	Quantile regression function
RSS	Residual Sum of Squared Error
G	Gini impurity test
C	Number of classes
n	Number of instances
p	Number of attributes
$P(i)$	Probability of randomly selecting a data point with class i
MAE	Mean Absolute Error
MSE	Mean Squared Error
$RMSE$	Root Mean Squared Error
R^2 & $\bar{R}$	R-Squared & Adjusted R-Squared
$AUReC$	Area Under Recall Curve
S	Number of thresholds
k	Number of attributes of a dataset
Z	Number of extracted features
$P(C_i\|X)$	The conditional probability of the class C_i given X
EW	Equal Width intervals
EF	Equal Frequency intervals
KM	K-Means clustering
CDF	Conditional Distribution Function

Table 1: Table for notations

1 Introduction

This document is a report of a 4 months internship at Orange Labs Lannion, France. It has been conducted in the "Innovation" branch, and more precisely in the "PROF" team. During this internship, I had the opportunity to explore the use of classification models to execute a *feature engineering* step before using a regression model to do a prediction.

This report is constituted of 5 main sections: Section 1 introduces this report. Section 2 defines the most important concepts that will be useful for the understanding of this report. Section 3 describes the current state-of-the-art regressors along with different relevant topics that will be addressed in this report. Then, section 4 details the method that is the main focus of this study. Section 5 presents the experimentation that is implemented to determine wether or not our method has a positive impact on performance. Finally, section 7 tackles the project scientific approach aspects of the study, while section 8 concludes this study.

1.1 Presentation of the subject and its objectives

In supervised learning, two problems can be distinguished from each other: Classification and regression problems. The kind of problem depends on the nature of the explained variable Y (quantitative or qualitative). In the case of regression problems, training a model to predict its quantitative variable involves fitting the model as closely as possible to a dataset in order to estimate a function Y_i of X_i and β, such as:

$$f(X_i) = Y_i = f(X_i, \beta) + e_i$$

The goal is therefore to minimize the additive error term e_i.

Aside from the optimization of the model and its parameters, data scientists can rely on other methods to reduce this error. One of them is feature engineering (see section 2.2), which consists of creating new features from raw data. The method proposed in this report is a feature engineering step which aims to create new informative variables using classifiers. Take X'_i the created features, the previous estimation function now becomes:

$$f(X_i \cup X'_i) = Y_i = f(X_i \cup X'_i, \beta) + e_i$$

To be able to use classifiers on a regression problem, the target variable must be transformed in classes. To do so, we rely on a process called *discretization* (see section 4.2.1). This report will explore some of the methods of discretization of the state-of-the-art. The features that can be extracted using classifiers are diverse, and some of them are the number of the node used to make the prediction in a tree-based method, or the conditional probability of the prediction. A focus is made on the latter for this report. We expect these new features to give the regression models a better intuition of the value of the target variables, hence an increase in performance.

The method proposed in this report is thus a framework that can make use of any kind of classification model to generate various features in order to improve the performance of a regression model.

1.2 Presentation of the company

Orange is the largest french telecommunications company with 146 768 employees in the world in 2019, including 90 000 in France. At the end of 2019, it had nearly 266 million customers worldwide.

Orange was originally a British company that was bought by France Télécom (the first french telecommunications company) to become its international brand and advertise its multiple mobile network brands. France Télécom was born in 1991, but its history goes back to 1792, when the very first french communication network was created, the Chappe optical telegraph network. After the invention of the electric telegraph, the french state creates in 1878 the Ministry of Posts and Telegraphs which quickly annexes the phone services in 1889. Which is now known as the PTT (Posts, Telegraphs and Telephones). This administration experiences a rapid growth between 1950 and 1970 when the requests for phone subscriptions explodes. It is only in 1988 that France Télécom

is born when a European directive calls for the reorganization of the French telecommunications sector following the model existing in the United States of a competitive telecommunications market in which publicly traded international trading companies operate. And in 2000, France Télécom purchases Orange for 40 billion euros. It then constitutes the second European mobile network. Numerous other acquisitions of companies (GlobalOne, Equant, Internet Telecom, Freeserve, EresMas) allow it to become the fourth largest operator in the world. In September 2004, the French State sold part of its shares and France Telecom became a private company. In 2006, most of the group's activities went under the Orange brand and logo. This is the case for Internet, television and mobile telephony services as well as digital services. Finally, the name change is voted on during a general meeting in 2013, France Télécom then becomes Orange definitively.

Orange has several research and development sites. Each of these sites works in many areas of research around the issues of the company, or knowledge related to its issues. The building of Orange Labs Lannion is one of the largest of the research and development centers with around 1000 employees.

The team in which the internship takes place is the PROF team (PROFiling and datamining) which is specialized in data handling, statistics and development of powerful data-mining tools and has 25 members including 5 PhD students. These tools are used for many different applications, like profiling, scoring or recommendation. This team passes on its expertise to the operational teams. It helps them by offering them technical solutions and strategic choices (methodology, architecture, internal and external technologies, etc.). This team is very mature in machine learning techniques, and in partitioning techniques.

2 Key concepts

2.1 Regression and its different forms

Regression designates the process of estimating a continuous numeric variable using other variables that are correlated to it. Regression problems are distinguished from classification problems. They are related to the prediction of a quantitative variable, while classification problems relate to the prediction of a qualitative variable. This means that regression models take the form of $y = f(x)$ where y can take an infinite set of values.

The most widely known regression model is the linear regression, which can only output a simple value as a prediction. But if the goal is to obtain a model that can take a quantile as a parameter to produce its prediction of the distribution of the target variable, quantile regression can be used [20]. When the target variable can be modeled with a binomial distribution, logistic regression is commonly used. And if the shape of the distribution is unknown, nonparametric regression is used.

2.1.1 Example of deterministic regression: Linear Regression

Linear regression [10, Chapter 2.2] allows modelling the relationship between one or more variables and a target variable using a linear equation which is:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} = X_i^T \beta + \beta_0 \quad ; \quad i = 1, ..., n$$

This is the simplest regression method. It is often estimated using the least squares method, but many other estimation methods exist like the maximum likelihood or the bayesian inference.

2.1.2 Example of probabilistic regression: Quantile regression

As opposed to linear regression which is a deterministic way of making a prediction, quantile regression is a type of regression that fits conditional quantiles on the distribution of the target variable. Quantiles are more widely known as percentiles, and conditional quantiles are functions that go from probabilities to the target variable space. They are often written as: $Q_\tau(Y|X)$. These quantiles make no assumption on the parametric form of the conditional

response.
The standard form of linear regression models calculates the conditional mean of the target:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} \quad ; \quad i = 1, ..., n$$

Where p is the number of features and n the number of data points.
So quantile regression implements the concept of quantiles in the linear regression models with the following equation:

$$Q_\tau(y_i) = \beta_0(\tau) + \beta_1(\tau) x_{i1} + ... + \beta_p(\tau) x_{ip} \quad ; \quad i = 1, ..., n$$

In this version, the beta coefficients are not scalar weights but now functions with a quantile parameter. Finding these functions and their values at precise quantiles is a process very similar to the linear regression [28].

Figure 1 is an example dataset on which a linear regression was fitted (blue line), as well as a quantile regression of which 4 conditional densities were extracted (green curves).

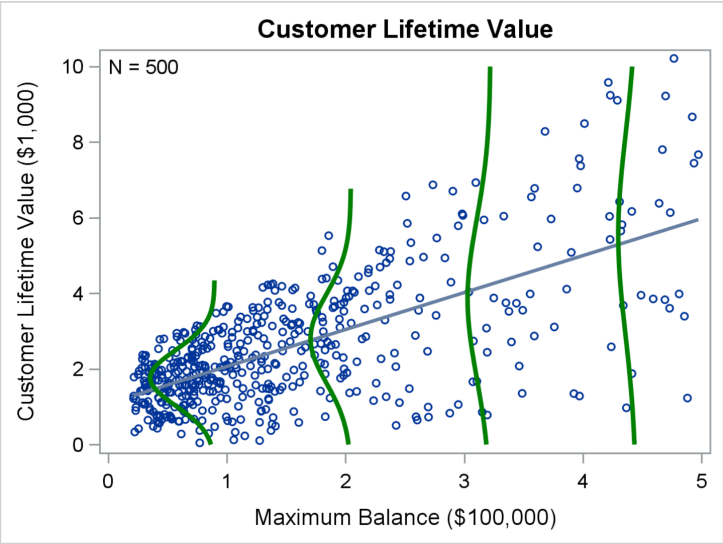


Figure 1: Taken from: R. Rodriguez *et al.*, Five Things You Should Know about Quantile Regression (2017) [28].

2.2 Feature engineering

Feature engineering is a very important step for the conception of a successful model that is often neglected despite its usefulness [22]. It consists of creating or extracting new features from existing data, and as such, it can be used for almost every type of problem (regression, classification, clustering, natural language processing, ...). It also allows data scientists to input their expertise of the domain and to highlight important information. Common examples of this practice would be decomposing a date-time so the model will have less difficulty taking advantage of the information contained into it. Another alternative would be creating a new cardinal feature that classifies date-times into 4 categories: morning, midday, afternoon and night. Because the method proposed in this report uses classifier to generate new features that will be added to the original dataset before training the regression model, it is close to a feature engineering step.

3 Brief state of the art of 'usual' regressors

3.1 Usual regressors used in this report

3.1.1 Introduction and Scope

Regression algorithms use various techniques, among which the 7 most widely used [14] are: Linear regression, Ridge regression, LASSO (Least Absolute Shrinkage and Selection Operator), Polynomial regression, Stepwise regression and ElasticNet regression. Each of these techniques have their own characteristics and are best designed to resolve different types of problems. Thus, in order to get the best results, it is important to choose the regression algorithm based on the attributes of the dataset, the target variable and the multicollinearity of the dependant variables. In this section, the regression models that are used in this report will be described.

3.1.2 Random Forest

Before introducing the random forest model, the concept of classification and regression trees, which abbreviates in CART, is going to be introduced. It was proposed by Leo Breiman to denote decision trees that can be used for classification or regression problems [24].

The main idea behind CART is quite simple: A set of rules is identified to allow the association of a class label to any data point. When growing a single tree, the algorithm will determine the optimal new split by exploring all allowable splits of all rxplanatory features of the dataset and select the split that minimizes a certain metric. When no further splits are possible (either because there are too few data points to consider or because the maximal depth of the tree has been reached), the node becomes a leaf. In classification, a class label is predicted, so the leaves outputs can take a finite set of values (e.g., True of False) and in regression the output can take an infinite set of values (e.g., a price).

In a classification context, the CART algorithm can use either the Gini Impurity test or the entropy (or *information gain*). The Gini Impurity test measures how often a randomly selected element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset considered for the split.

Given C the number of classes and $p(i)$ the probability of randomly selecting a data point with class i, the formula for calculating the Gini Impurity is:

$$G = \sum_{i=1}^{C} P(i) * (1 - P(i))$$

A perfect split with no impurities will have a Gini Impurity of 0, whereas a value of 0.5 would be obtained when half of the points are correctly classified in the subset.

For regression problems, the splits can be chosen by minimizing a simple squared error.

Since decision trees decide for each split which feature of the data should be used for the split, the usual scale issue of the different features of the dataset that appear in many machine learning algorithms is not present here. This is why decision trees based algorithms work well even without the pre-processing steps (like normalization).

The first algorithm using random decision in forests was created in 1995 by Tin Kam Ho [15] and was later extended by Leo Breiman [7]. The extension used Breiman's own bagging concept and random selection of features.

A random forest is composed of an ensemble of tree classifiers (or regressors). When growing the trees, it chooses the feature to use for each split of every tree from a random subset of all of the features. To make a prediction, each tree casts a vote to determine the class of each input. Many previous methods already used random selection to improve their accuracy and reduce their variance (e.g., *bagging* (Breiman, 1996) randomly selected examples in the training set and in *random split selection* (Dietterich, 1998) splits were randomly selected from the **K** best splits). Where random forest diverges from the other methods is that it selects a random sub-sample of variables of the individuals for each split, as well as selecting the individuals randomly (with replacement). So each tree is

grown to the largest extent possible and there is no pruning of the nodes. New data is then predicted by majority vote of the trees for classification and average for regression. Breiman explains in his paper that the Strong Law of Large Numbers shows that the trees always converge. This means that the variance of the generalization error will approach zero when more trees are added to the forest. But the bias of generalization wont necessarily approach zero, it only means that the forest will always predict the same values which is not necessarily accurate. For accurate random forest regression, low correlation between the trees and low prediction error for each individual tree is needed. But once these conditions are met, random forest are able to handle large data sets with high dimensionality.

3.1.3 XGBoost

XGBoost [9] is a boosting method, it sequentially combines weak learners that would individually produce poor performance to improve the complete algorithm's prediction. A weak learner is a classifier that has a low correlation to the correct classes (it performs arbitrarily better than random guessing). So in a boosting algorithm, high weights are associated to weak learners with good accuracy, and lower weights to weak learners with poor accuracy. In the training phase, high weights are associated to data that was misclassified so that the next weak learner in the sequence will focus more on the data that the previous weak learner misclassified.

The term XGBoost actually refers to an open-source library which is an optimized and improved version of the gradient boosting method. It was specially designed to be computationally efficient in order to get the best performance in the shortest amount of time. Which is why it was written in C++ (it has a python interface and a model in scikit-learn which we will be using). Some of the most important features of the implementation of the algorithm are: the handling of missing values, the support of parallel tree construction and the ability to further enhance the performance of an already fitted model using new data.

XGBoost supports three different forms of gradient boosting methods: Gradient Boosting, Stochastic Gradient Boosting and Regularized Gradient Boosting. It also has many useful features like built-in cross validation and model tuning methods or compatibility with many languages (Python, Java, R, C++, Julia, Scala, Hadoop). It can be used for both regression and classification.

3.1.4 Khiops

Khiops [16] is the tool proposed by Orange Labs for data preparation and modelization, for supervised and unsupervised learning. It allows a non-parametric evaluation of the correlation between variables of any nature for the unsupervised case, and the estimation of the predictive importance of variables or pairs of variables in the supervised case. These evaluations rely on supervised discretization methods when variables are numeric, and on grouping methods when variables are categorical. The methods inside Khiops have been the subject of numerous publications.

In the regression context, Khiops performs 2 main steps to generate a Naive Bayes regressor which weighs the explanatory variables. At first, for each explanatory variables, it creates 2D grids as an estimation of $P(X, Y)$ which is illustrated in figure 2.

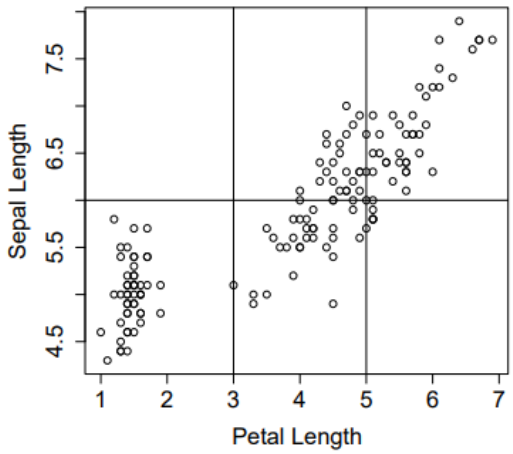


Figure 2: Example of a 2D grid discretization with 6 cells. Taken from [16].

We do not detail here how these grids are found but in a few words, an algorithm explores the potential 2D supervised grid (one per explanatory variable) and using a criterion from the MODL family [5], the 'best' grid is found. The MODL criterion allows to find the supervised 2D grid which maximize a gain compare to the grid which follows the assumption of independence between X and Y (see [5] for more details).

Then, in a second step, all the variables are grouped together in a Forward Backward algorithm [6] to estimate their informativeness in the context of a Naive Bayes regressor. At the end of the second step, the final model (to be deployed) is a Naive Bayes (which uses the 2D discretization found in the first step) where variables are weighed (weights are found in the second step). This tool has been designed to handle large datasets, with hundreds of thousands of individuals and tens of thousands of variables, and has successfully participated in several international challenges.

3.1.5 Linear Regression

As linear regression was presented in section 2, it wont be described any further here. However, this regression model will be used as a baseline during the experimentation. Baselines can be defined as simple models that can still produce decent results while being quick to train. The linear regression model used in this report is the one from sklearn.linear_model.LinearRegression. Since it is in closed form and doesn't require to be iteratively trained, it also doesn't have any hyper-parameter to optimize. Which is why it will serve as a baseline for the other more complex models that will be used.

3.2 Pre-processing of the datasets

3.2.1 Normalization of the target variable using Box-Cox

In many statistical techniques, the errors are assumed normally distributed. This assumption allows to construct confidence intervals and conduct hypothesis tests. The Box-Cox transformation [3] transforms a non normally distributed positive variable so that it resembles a normal shape.

The Box-Cox transform is given by:

$$y(\lambda) = \begin{cases} (y^\lambda - 1)/\lambda, & \text{if } \lambda \neq 0. \\ log(y), & \text{if } \lambda = 0. \end{cases} \tag{1}$$

Since the Box-Cox transformation is designed to be used on non-negative responses, the target variable is rescaled to be between 1 and 2 using a simple min-max scaler (sklearn.preprocessing.MinMaxScaler). The Box-Cox method used is sklearn.preprocessing.PowerTransformer with the parameter *method="box-cox"* which uses the equation previously defined and estimates λ through maximum likelihood estimation.

In a production environment, it would be needed to re-transform the target variable back in its original shape in order for the results produced by the models to really make sense. But since this is only a theoretical work, the target variable wont be re-transformed before computing the residuals (i.e., the errors). Nonetheless, the inverse function of the Box-Cox transform that would normally be used is given by:

$$y(\lambda) = \begin{cases} (y^\lambda - 1)/\lambda \\ log(y) \end{cases} \iff \begin{cases} \lambda \times y(\lambda) + 1 = y^\lambda \\ y(\lambda) = log(y) \end{cases} \iff \begin{cases} (\lambda \times y(\lambda) + 1)^{1/\lambda} = y & \text{if } \lambda \neq 0 \\ e^{y(\lambda)} = y & \text{if } \lambda = 0 \end{cases} \tag{2}$$

3.2.2 Elimination of collinear variables

Collinearity means that there is a significant correlation between 2 features (also called *independant variables*). If this relation exists between 3 or more features, it is called multicollinearity. For some models such as linear regression, a multicollinear set of features will negatively impact the estimation of the relationship of certain features to the target variable (or *dependant variable*). Indeed, as one change in a variable will also imply a change in the variables collinear to it, this will result in a bigger change in the target variable. This means that models will overestimate the influence of the said variable on the target variable.

The best practice to detect collinearity is to calculate the Variance Inflation Factor (VIF) as collinearity can occur between 3 variables of more and the correlation matrix is not enough to detect all cases of collinearity [12].

We chose to keep the collinear variables since the goal of this report is only need to demonstrate that our method has a positive impact on performance. Removing the collinear variables will only improve the overall performance of the models, whether the proposed method is used or not.

3.2.3 Elimination of uninformative explanatory variables within the meaning of an ML

When a feature has a single unique value across all instances, it is considered as an uninformative feature (also called *zero variance predictor*). For instance, such features can appear when splitting a categorical variable into several variables. Uninformative features do not necessarily have a negative impact on performance for all models: Decision trees based methods will simply never use them in a split. However some models like the linear regression will likely be negatively influenced [21]. Max Kuhn and Kjell Johnson proposed in Applied Predictive Modeling, 2013, that uninformative features be defined by:

1. A low fraction of unique values over the sample size (under 10%).
2. A high ratio of the frequency of the most present feature to the second most present feature (above 20%).

Nonetheless, this definition is not a universal rule and all data that check both criteria do not automatically have to be removed. To detect constant (or almost constant) features, some packages have built-in functions. In R, the *caret* package has a function called *nearZeroVar* that removes predictors with one unique value across all samples, and detects predictors that match the definition of Kuhn and Johnson. For python, a similar function can be found in the scikit-learn library and is called *VarianceThreshold*.

During the experimental protocol, and more precisely during the datasets selection and pre-processing, variables with only unique value were removed. However, near-zero variance features were not removed for the sake of simplicity.

3.2.4 Normalization of the explanatory variables

Normalization is a process that is very commonly used in the data preparation phase of machine learning. Its goal is to define a common scale that will be used by every feature of the dataset, without altering or changing the information of the features.

There are different types of normalization in statistics, some of them are: Standard normalization, or standardization. It re-scales the data to have a mean of 0 and standard deviation of 1. Min-Max scaling is another type of normalization, it is used to bring all the values between two defined values (commonly [0; 1]).

This practice has many benefits, of which can be cited:

- Making training less sensitive to the scale of the features: If a dataset has two features, A that ranges from 0 to 1 and B that ranges from 100 to 1000, a linear regression model will have two coefficients w_1 and w_2 of A and B respectively. Now, a small change in w_2 will have a very large impact on the result of the prediction compared to the same change for w_1, making w_2 unnecessarily dominate the optimization process.
- Preventing regularization to have variable impact for different scales: l_2 regularization terms add the sum of the square of the weights to the cost function. Because of that, it has a strong impact for large weights, and it reduces as the weights get smaller. So, for a given optimal weight parameter w_1, if the scale of its feature is multiplied by 10, the weight w_1 would be expected to go down by the same factor. But in reality, the l_2 term will have a smaller effect during optimization because w_1 will be smaller. Which will result in a larger than expected value for w_1.
- Consistency for comparing results across models: as scaling impacts performance, it is useful to scale the features of every dataset used in order to be able to compare the results produced by a model.

However, in tree-based algorithms (like decision trees or random forests), this process is not required as this type of models will divide the space by looking at which cut will yield the best improvement in terms of the error function that is used. And a change in the scale wont influence the position of the splits, features are not compared to each other but are only taken individually. It is also unnecessary for Naive Bayes models since the features are only compared to each other and unit issues don't exist in this case.

Despite this, it was decided for more simplicity that in the experiments conducted in the internship, all datasets will go through the same pre-processing and will all have their dependant features (or explanatory variables) standardized.

3.2.5 Handling of the categorical variables

Datasets used for real applications are typically a mixture of continuous and categorical variables. Continuous variables are not a problem for classifiers. But most of them, because of their implementation in a libraries like scikit-learn, are not able to natively handle categorical variables. To use these datasets, the categorical variables have to be re-coded. The most common choice is the one-hot encoding, but it seems to negatively impact the performance of decision trees [26], especially when categorical variables have high cardinality. Since one-hot encoding will create many variables, the resulting trees will be very sparse and simple, which is undesirable. So the different alternatives and their performance have to be explored.

1. Numeric Encoding (also called label encoding):

   The operation of numeric encoding is very simple. An arbitrary number is assigned to each category. When the cardinality of categorical variables remains low, performance will be very good. But as it increases, decision trees will exhibit more splits (but shallower depth) compared to one-hot encoding. When the cardinality is large, this encoding requires a lot of data to avoid having too random performance.
   It can be noted that this method imposes a false feeling of ordinal relation between the classes which have been encoded (e.g., 5 <81).

2. One-Hot Encoding:

   One-hot encoding works by creating as many columns as there are unique values taken by the categorical variable at the time of encoding. The result is a number of columns with a majority of 0s and a few 1s. This method has the disadvantage of increasing the size taken in memory by the dataset in proportion to the cardinality of the encoded categorical variables (many 0s will be stored in memory).

In the blog article "Visiting: Categorical Features and Encoding in Decision Trees" - (Damien Soukhavong, 2017) [29], the author compares the performance of 4 encoding methods. Her conclusion is that with a cardinality under 32, One-Hot has a performance equal to other methods, but above 32 it rapidly falls under other methods like numeric encoding. There is however an issue with his experimentation: The author used a single dataset to produce his results with only one attribute which was perfectly descriptive. This is not a real-world scenario so it was decided to produce more robust results before choosing the preferred method of encoding.

Three different encoding methods were used (Numeric encoding, One-Hot encoding and Binary encoding) on 3 datasets (*Categorical Feature Encoding Challenge* [19], *Income Prediction* [23] and *Car Evaluation* [11], which were all taken from Kaggle). The model that was selected is the decision tree classifier from scikit-learn, since it seems to be the model that is the most affected by encoding methods.

The experimentation was the following: Only the categorical attributes of each dataset were used and each dataset was encoded with the 3 encoding methods before running 10 training and prediction with the decision tree classifier on these encoded datasets.

Encoding method	Dataset 1	Dataset 2	Dataset 3	Mean accuracy
Numeric Encoding	0.623	0.818	0.969	0.803
One-Hot encoding	0.658	0.820	0.956	0.811
Binary encoding	0.664	0.819	0.953	0.812

Table 2: Comparison of the performance of 3 encoding methods

It can be noted that the one-hot encoding method added around 16,000 attributes to the first dataset, which severely impacted the training time of the classifiers. However the performance didn't drop as it did in the report [29]. This can be explained by the fact that this dataset had 17 attributes that could be used for the prediction. So in this more realistic scenario, One-Hot encoding does not have a particular disadvantage compared to numeric encoding.

Using more realistic datasets therefore demonstrates that the one-hot encoding is not at a disadvantage compared to other methods, but only has the drawback of increasing the dimensionality of the input data when some features have high cardinality. Each of the datasets that is used in the experimentation was studied, and none of them had a feature with a cardinality higher than 4. Which is why one-hot encoding will be the method used for the method proposed in this report.

Note: Datetime features are a special case of categorical data. It doesn't make sense to use one-hot encoding on raw datetimes as the cardinality is way too high, and while numerical encoding may be better since it will preserve the ordinal relation of the dates, encoding the datetimes of the data to predict will be a struggle since there will be little chance that the same exact datetimes would have been encountered in our training data to fit our encoder. The best approach is to extract features for the datetimes to enrich the features set. The the year, month, day, hour, minutes, etc... can obviously be extracted from the data, but features like the season, semester, holiday or not, week of the year, name of the day, etc... can also be extracted The choice of features is context dependant (e.g., the year can be excluded if the model needs to be usable in the future, and the future years wont be included in the training dataset).

When working with time values, one of their essential features needs to be handled carefully: **periodicity**. Indeed, let's take the times 23:00 and 04:00 and ask which one is the closest to 00:00, the answer is obvious to us: 23:00. But this is not so obvious for models, as in raw numerical terms it is 04:00 that is closer.

To express this cyclicality (or periodicity), a sine and a cosine transform can be derived to create two new features and carry the cyclical nature of the 24-hour time.
The approach that will be usde to encode datetime features in our project is as follows: Extract year, month and day. Then convert hours, minutes and seconds in a seconds only variable. And finally apply a sine and a cosine transformation on our seconds to create our last two features.

The one-hot encoding method will therefore be used accross all the categorical attributes of the tested datasets, along with the cyclical encoding of the datetime features.

3.3 Metrics for comparison

When comparing multiple regression models with the intention of selecting the best performing one, the goal is to find a model with no over or under-fitting that reaches a low generalization error, which characterizes its prediction performance. As no consensus on what is the "best" metric to use, this section will explore some of the most popular metrics available. In [4], Alexei Botchkarev extensively described and compared the most popularly used metrics in the literature.

3.3.1 Mean Absolute Error (MAE)

This is the average of the absolute error between the prediction and the actual value. The absolute error is simply the error whose negative sign is ignored if present. The MAE is computed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|$$

This error has the advantage of having the same scale (or unit) as the original data. Thus, it is only possible to compare two MAEs if both models have the same units. Which is why it is called a scale-dependent accuracy measurement.

3.3.2 Mean Squared Error (MSE)

The Mean squared Error is a metric that calculates the average of the squared errors. That is, the average of the differences between the predicted values and their actual values. Here is its formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2$$

The root mean square error is an estimator of the quality of a model, where a value close to zero indicates good prediction quality.
As the error is squared here, outliers will be more penalizing than when calculating the MAE, and thus better detected. On the other hand, the unit is also squared.

3.3.3 Root Mean Squared Error ($RMSE$)

This is the square root of the squared error:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2}$$

This metric therefore has the advantage of getting rid of the MSE unity problem. Compared to the MAE, RMSE gives a larger error value because it penalizes outlier predictions in the same way as MSE on which it is based. If the goal is therefore to obtain a model with the fewest possible outliers, the choice should be directed towards RMSE rather than MAE.
It can be noted here that when the size of the dataset used for the assessment increases, the difference between MAE and RMSE will also increase since $\sqrt{\frac{1}{X}}$ tends more slowly towards zero than $\frac{1}{X}$.

3.3.4 R-Squared (R^2)

As the previous section described, MAE and RMSE both have the advantage of giving a result in the same unit as the prediction. This makes it easy to interpret the error. On the other hand, it is difficult to compare the results of two predictions which have different units. It is therefore here the goal of the R-Squared metric which is calculated according to the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where y is the true value, $\bar{y}$ is the mean value and $\hat{y}$ is the prediction.

This metric is often seen as the percent of variation in the target variable that the model explains. So an R-Squared of 0.42 would mean that the model explains 42% of the variance of the data. In general, the results will be between 0 and 1, but it can happen that values outside of this range appear when the wrong model has been selected (If for example, the prediction is worse than the average, the score will be negative). In other words, R-Squared explains how much better the modeling is than just predicting the mean.

3.3.5 A note on R-Squared

Despite being one of the most popular metrics used to asses the quality of a model, R^2 is a metric that must be used with caution. Indeed, high R^2 values aren't inherently good, and low values inherently bad. Another way to see R^2 is:

$$R^2 = \frac{Var[m(X)]}{Var[Y]}$$

So if the variance of the target variable $Var[Y]$ is large or the variance of the model Var[m(X)] is small, the R^2 can be arbitrarily low even if the model fits perfectly the data. In the same manner, R^2 can be close to 1 even when the model is completely wrong when $Var[m(X)]$ is large.

R^2 cannot be compared between different datasets. Look at the equation of R^2 and see that the same model can have radically different R^2 values on different datasets. For the same reasons, R^2 cannot be compared before and after the transformation of the target variable Y.

The only situation where R^2 can be compared is when the same dataset is used to fit different models. And in that case, an increase in the R^2 will be the same as a decrease in the MSE.

Because the method that will be used later in this report will work by adding new features to the dataset, it would be unwise to use R^2 for the reasons stated above. And if adding new features increased the noise in the dataset, and so the variance of the model, the R^2 would be lower for no valid reason, making it unfit to be used as our metric of choice.

3.3.6 Adjusted R-Squared ($\overline{R}$)

One of the disadvantages of R-Squared is that the value it calculates is (falsely) influenced by the number of features. If it increases, so does the result. And this without taking into account the importance of the added

features, which is generally undesirable. Adjusted R-Squared is therefore a metric which adjusts its result using the number of features p relative to the number of data n. It is defined as:

$$\overline{R} = 1 - (1 - R^2) \times \frac{n - 1}{1 - p - 1}$$

For the reasons described above, Adjusted R-Squared is a much more interesting metric when performing multivariate linear regression. Since it penalizes the addition of new features that do not improve the performance of the model.

In other words: Adjusted R-Squared is an R-Squared-based metric that describes the proportion of data variance described by our model, while adjusting its result to the number of features used in the prediction.

3.3.7 Conclusion

MSE evaluates the error with a squared unit which is generally undesirable, and mathematically, using the absolute value for evaluation is not justified. RMSE therefore has a distinct advantage over the two errors previously considered, but still has the disadvantage of not being able to be compared between two datasets relating to data of different type since it is also domain dependant. However, the target variable of every dataset will be transformed by a min-max scaler to a [1; 2] value range, followed by a Box-Cox transformation, so the errors computed will all have the same scale. Using these metrics to compute a heuristic value wont be an issue either since they will only be used to compare similar data, the goal is here to compare the performances of different models against each other on the same dataset, not between datasets with different target values.

And as stated before, R-Squared is not a metric that can be used in this case since the method of this paper will potentially increase the variance of the models by adding new features, thus unfairly reducing the value of this metric.

RMSE therefore seems the best suited because it does not have any of the shortcomings of the metrics considered previously.

4 The explored method

4.1 Context

In a regression problem, the target values are taken from a continuous space, while in a classification problem, they are taken from a discrete finite space. Classification algorithms are easily understood, but are not applicable when the target values are continuous. This means that the target variable of the regression problem has to be discretized (see section 4.2.1) to allow the use of classification models. Once done, the problem becomes simpler and the performance of the classifiers will generally be good. For this reason, the idea of using classification algorithms to solve a regression problem in the hope of having similarly good performance is not a new concept.

It has been described in many papers, such as in [1, 2, 17, 18, 25, 30] and usually consists of two main stages:

1. The discretization (also called partitioning or binning) of the target variable in order to allow the use of classifiers on the dataset. Some popular methods of discretization are equal-width, where the output variable is split in equally large fragments, and equal-freq, where an even number of instances is distributed across each fragment [8].

2. The regression's prediction is then generally realized by computing the mean ([1, 2, 17, 25]) or the median ([2, 18, 30]) of the instances inside the fragment of the discretized output that the classifier predicted. The median has the advantage of being more robust to outliers.

Our method diverges from this usual way of using classification models for regression problems because we will still be using a regression model to make the final prediction. The classifiers will only allow us to extract more information on the dataset before giving it to the regressor to help him make a more accurate prediction (which is why it can be associated to a feature engineering step [31]).

4.2 Classification for regression: our suggested general principle

Our method consists of a few distinct steps. The goal of the first step is to cut the target variable's space into bins in order to create and associate classes to every instance in the dataset. This process is known as discretization and has been well studied [13]. It can be either supervised or unsupervised (clustering methods like K-Means). The equal-frequency approach was chosen here for its simplicity.

The general idea is that once the target variable's continuous space has been discretized into bins that are delimited by thresholds, a number of classifiers will predict whether the input instance is above or below the threshold the classifier is associated to. The conditional probabilities predicted by each classifier will then be added to the dataset as new features. After this step, the regression model can take this extended dataset as input to be trained and make its prediction. The assumption is that the regression model will have a better intuition of the general position of the instances in the target variable's space and will have better results.

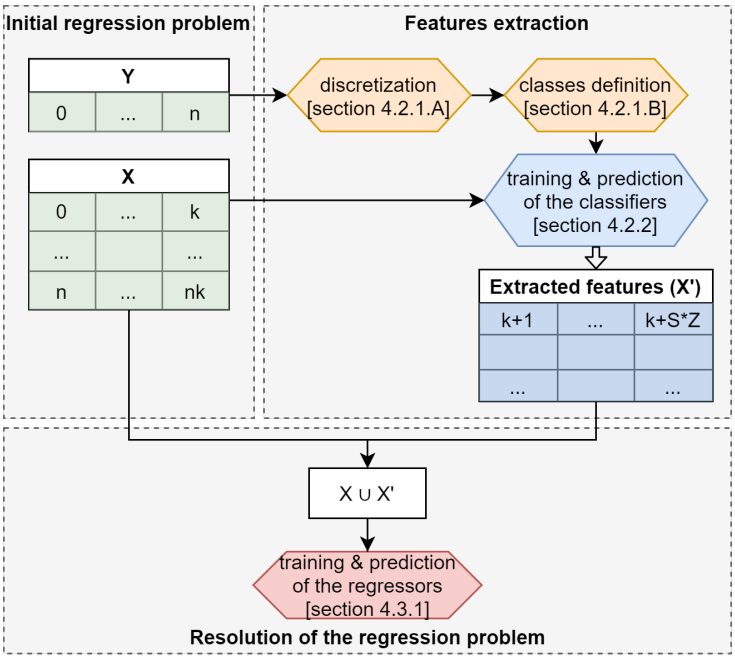


Figure 3: Schematization of the general principle of the method

Figure 3 summarises the process followed by the method proposed in this report: The first step is to transform the regression problem into a classification problem. To do so, the target variable Y is first discretized (section 4.2.1.A), then the classes are defined using the discretized Y (section 4.2.1.B). The classifiers can now be trained using the initial descriptive variables of X and the new classes derived from Y (section 4.2.2). The prediction of the classifiers on the initial X is then used to extract new features into X'. And finally, the regression model can be trained on $X \cup X'$, and we can measure its new performance.

4.2.1 Discretizing a continuous target variable

A. Defining a set of intervals

One of the key aspects of the proposed method is the definition of the intervals. In this report, only the methods of equal-width and equal-frequency discretization were explored. They only need the target variable's values as input. The equal-width method creates as many equally large bins as needed between the maximum and minimum of these values. On the other hand, the equal-frequency method needs to create bins that will contain the same number of instances each. This last method comes with some challenges that will be discussed in a later section.

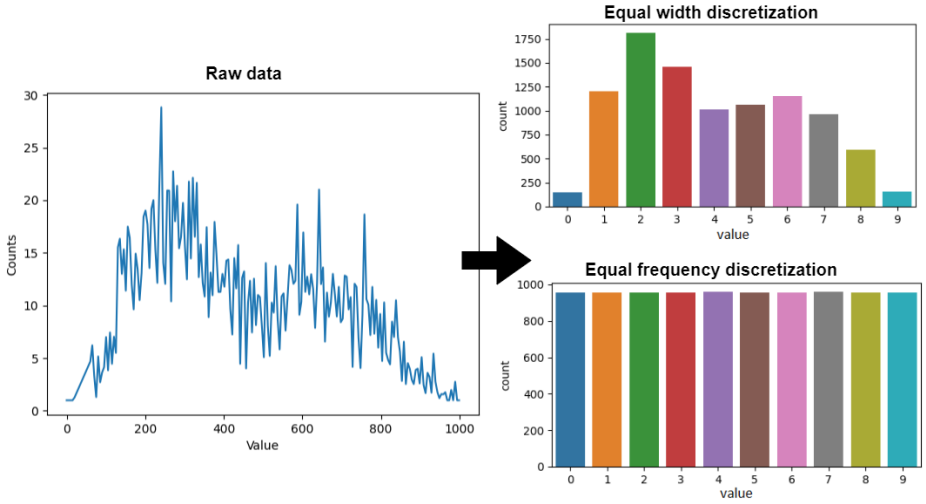


Figure 4: Example of data discretization on the Combined Cycle Power Plant Dataset

The figure above illustrates the result of the process of discretization with the two studied methods. The left part of the figure represents the distribution of the data across the target variable's space and the right part represents the distribution of the data in the ten bins that were defined with the two methods. The equal-width method naturally has a varying number of instances in each bin, whereas, by definition, equal-frequency has a fixed number of instances across all its bins.

B. Associating classes to the intervals

We define three principal ways to train classifiers once the thresholds have been defined on the target variable space. The type of classification depends on the classes definition. The most straightforward way is to give each bin a number, and each instance's class will be the number of the bin that it is contained in.

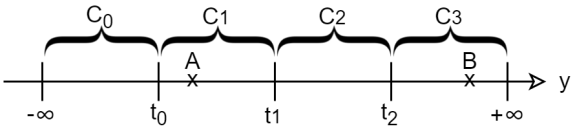


Figure 5: Association of classes by number of bin

In figure 5, the target variable's space was split using 4 bins, so 3 thresholds were defined. In this case, the point A belongs to class C_1 and the point B belongs to class C_3. This means that only a single classifier has to be trained because it will directly predict the class that the data belongs to.

The two other methods are both based on the following encoding: We start by defining thresholds as the separation between each bin (so for S bins, $S-1$ thresholds will be defined). Then, for each instance in the dataset, we evaluate for each threshold if the data is above or below that particular threshold. If it is above, the class is 0 and if it is below, the class is 1. This means that there will be as many classifiers created as there are thresholds.

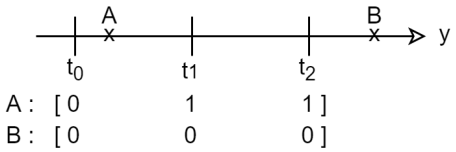


Figure 6: Association of classes by position relative to each threshold

In figure 6, since point A is above the threshold t_0, its first class value is 0, and because it is below t_1 and t_2, its second and third class values are 1. And because B is above every threshold, its class vector is only composed of zeroes.

Once this discretization process has been applied, there are two approaches that can be used: Either train a single classifier capable of outputting multiple labels at once (multi-label classification) such as a Logistic Regression classification model. Or train a classifier for each column in the class vector (so a classifier for each threshold). In the latter, this is a binary classification problem, which we expect to be an easier problem to solve than the multi-label or multi-class methods we described above.

C. Discussion

The choice of exploring only the equal frequency discretization method was made for two main reasons: The first is simply because measuring the effect of two different methods across all datasets adds a heavy cost in terms of computational time that could not be afforded. The second is that the equal frequency method is more robust than the equal width one: It can be expected that the equal width method wont show any significant improvement between a small and a large number of bins when outliers are present in the datasets, or when the target values are grouped in clusters. This is the biggest drawback of the equal-width method: it is prone to creating very unbalanced bins (empty or heavily populated bins) as it is unable to split narrow clusters of target values. This makes the equal frequency method more reliable in comparison which will create, by definition, bins that are as balanced as possible, despite having some implementational challenges.

Concerning the selection of the method of class generation, it was decided to explore the performance of two of the three methods that were defined above: The binary classes generation method with multiple classifiers that predict for each threshold if an instance is above or below a given threshold, and the numbered bins method, where a single classifiers predicts which bin an instance belongs to. The case where a classifier predicted all of the binary classes of an instance using multi-label classification was not implemented. This is a method that returns a result similar to the multi binary classifiers, but because this is a problem with a much higher degree of difficulty, the performance could only be inferior to the multi-classifiers method. And the quality of the result of the classification will directly influence the performance impact of the method proposed in this report.

Note: When using the multi binary classifiers, it is possible to increase the number of bins without having to re-train every classifier. When using a number x of bins that is a power of 2, the next power of 2 provides a discretization that is 2 times finer, but instead of having to retrain $2 \times x$ classifier, only x new classifiers have to be re-trained.

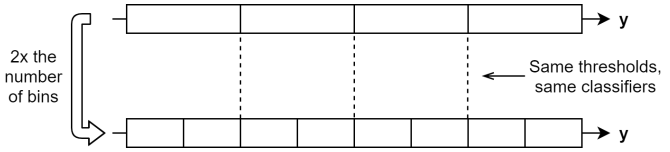


Figure 7: Illustration of the process of adding 2 times the number of bins

Figure 7 illustrates the process described above: When defining two times more bins than before, only half of the classifiers needed actually have to be trained.

4.2.2 Using classifiers to extract new features

The next step of the proposed method is the extraction of the new features using classifiers. To illustrate this process, we will take the equal-frequency discretization method with the binary class encoding of the instances (see fig. 6).

Given a dataset with k attributes that had its target variable split into $S+1$ bins using S thresholds and had classes associated to each instance following the above-or-below manner described in the previous section. We first train S classifiers, one associated to each threshold. Then, the classifiers are used to predict if the instances are above or below a given threshold.

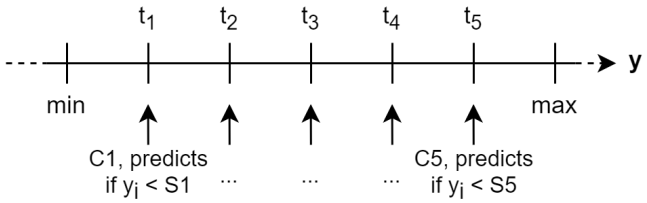


Figure 8: Representation of the classifiers associated to each threshold

Each of these S classifiers can now extract Z new features on each instance of the dataset. These new features depend on the kind of classifier that is being used: For instance with a decision tree we can extract the conditional probabilities $P(C|X)$ of each class and/or the number of the leaf that was used to make the final prediction. The new dataset augmented with the new $Z \times S$ features (so its new size is $k + (Z \times S)$) is then simply used for training and prediction of the initial regression model.

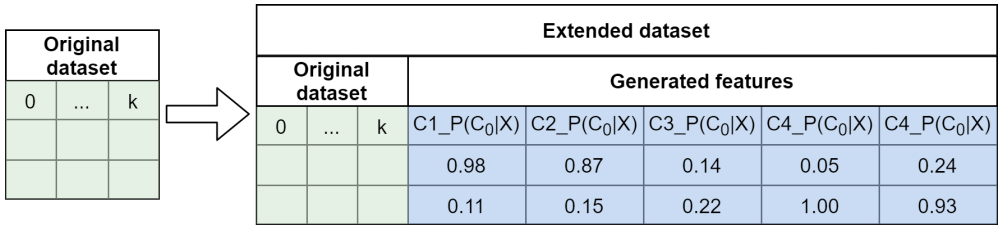


Figure 9: Dataset extension

Figure 9 illustrates the addition of features to the dataset. The dataset initially has k feature,s and 5×1 others are added, so the final dataset has $k + 5$ features in total.

4.3 Implementation

4.3.1 The method

The implementation of the method that was developed for the study was split into 5 main scripts that represent the different steps of the method. Each one of them save the different versions of the dataset (pre-processing, train-test split, extension, ...) and allow for better control and verification of its proper functioning. But this way of doing things is not the most optimized as it requires writing the datasets to the permanent memory before reading it again for the execution of the next script. Ideally, a single script would execute all the steps while only working in the temporary memory. This is this kind of design that was translated in the pseudo-code below:

Algorithm 1: Classification for Regression with binary classification

Inputs: $R \leftarrow$ A regressor model;
$C \leftarrow$ A classifier model;
$S \leftarrow$ A number of thresholds;
$X_{train} \leftarrow$ The training dataset with **k** features;
$X_{test} \leftarrow$ The testing dataset with **k** features;
$Y \leftarrow$ The training target variable;
;
S_list $\leftarrow$ **S** thresholds distributed across [min(Y); max(Y)] (by equal-width or equal-frequency);
A $\in R^{N \times len(S_list)} \leftarrow$ The binary classes defined by below-or-above method;
K $\in R^{N \times (S*Z)} \leftarrow$ The container for the extracted variables;
for *index* $\leftarrow$ *0* ***to*** *length(S_list)* **do**
  Train the classifier C_{index} on X_{train} to predict the column *index* of A;
  Extract into K the Z variables of C_{index} by predicting every instance of X_{test};
end for
Train the regressor R on $X_{train} \cup K \in R^{(k+S*Z) \times N}$ to predict Y_{train};

4.3.2 Other implemented algorithms

For the needs of the experimentation set up in this report, other a algorithms had to be implemented. In this section, the two most important ones are described.

A. Implementation of equal-frequency discretization

The equal-frequency discretization method is another key part of our method, since this is the discretization method that we chose to explore in-depth. It ensures that each bin is equally populated so that every classifier has a big enough sample of instances of each class to learn. The most straightforward implementation of this algorithm would be the following:

1. Sort the values of the target variable Y.
2. Cut this sorted Y into N lists of equal sizes.
3. Define the thresholds as the frontier between every list.

However, this only works if no value appears more than one time in the target variable passed to the algorithm:

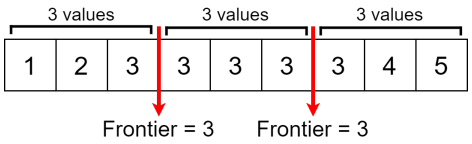


Figure 10: Example of wrong equal-frequency discretization

In the example above, the value 3 appears five times, which results in two frontiers having the same value when we try to define 2 thresholds. To overcome this problem, the following algorithm was implemented:

1. Sort the values of the target variable Y.
2. Define the ideal thresholds as the frontier between every list once Y has been cut into N lists of equal sizes.
3. For each threshold, find the closest frontier between two different values of Y.
4. Remove the duplicate thresholds.

So this new algorithm on the same example of figure 10 now returns the following result:

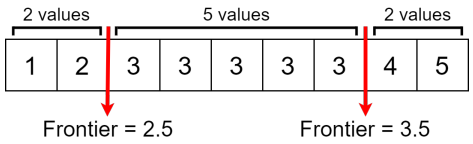


Figure 11: Example of fixed equal-frequency discretization

This way, the bins are as balanced as possible while the thresholds are the closest to their theoretical values as possible. The deletion of the duplicate thresholds still means that is not always feasible to create the desired number of thresholds, and as we try to define a larger number of thresholds, more duplicate values will appear.

The pseudo-code below translates the version of equal-frequency discretization that was implemented for this paper with more details:

Algorithm 2: Equal frequency discretization

Input: $Y \leftarrow$ The target variable;
$n\_bins \leftarrow$ the number of bins;
;
$thresholds \leftarrow []$ // An empty list;
$Y\_sorted \leftarrow Y$ sorted by increasing values;
possible_thresholds $\leftarrow$ Frontiers between all the unique sorted values of Y;
for $bin\_index \leftarrow 0$ ***to*** n_bins **do**
  $ideal\_bin\_index \leftarrow bin\_index \times \text{length}(Y) \ / \ n\_bins$;
  $ideal\_bin\_index \leftarrow \text{floor}(ideal\_bin\_index)$ **if** $bin\_index <= n\_bins/2$ **else** $\text{ceiling}(ideal\_bin\_index)$;
  $ideal\_bin\_position \leftarrow Y\_sorted[ideal\_bin\_index]$;
  $real\_bin\_position \leftarrow$ the closest value of $possible\_thresholds$ to $ideal\_bin\_position$;
  insert $real\_bin\_position$ **into** $thresholds$;
end for
remove duplicate values from $thresholds$;
return: $thresholds$;

B. Implementation of the AUROC metric

The Area Under the Receiver Operating Characteristic (AUROC, also ambiguously called AUC) is the metric that was chosen to evaluate the performance of the classifiers used to extract the features. As the name suggests, it is computed as the area under the ROC curve which is plotted as the True Positive Rate (TPR) against the False Positive Rate (FPR) for multiple decision thresholds:

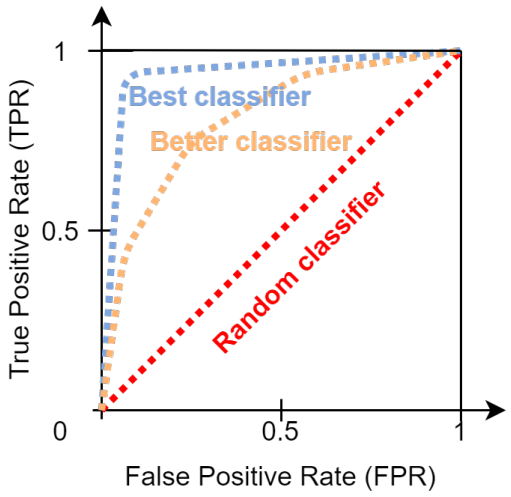


Figure 12: Illustration of the ROC Curve

An advantage of the AUROC is that it doesn't require a decision threshold, which is not the case for other classification metrics such as accuracy or F1-score. In a scenario of imbalance of classes, the accuracy is not a relevant metric as a classifier could predict that all the observations belong to the majority class and still have great accuracy. On the other hand, AUROC is not sensitive to such imbalance of classes, which is desirable and probably its best asset.

Initially, the sklearn.metrics.roc_auc_score method was used, but as this method returned errors in some cases, a custom method had to be implemented to compute the AUROC:

In a binary classification configuration, it is impossible to compute the AUROC when only one class is present in a given bin, as a division by zero occurs when computing the FPR or the TPR. In this case, it was chosen not to define a value and ignore these bins. As this only happens for the very first or very last bins of some datasets, an average AUROC value can still be computed for the other bins.

For multi-class classification, the method from sklearn returns an error when one class that the classifier is trained to predict is never present in the given dataset. For the same reason that was mentioned before, it is indeed impossible to plot a ROC curve for this class. However, the documentation from sklearn cites the section 6.2 of [27] as the reference used to define the multi-class score in the one-versus-rest way. And this is the definition of the one-versus-rest AUROC introduced in the paper:

> For data sets with more than two classes, we computed the expected AUC, which is the weigthed average of the AUCs obtained (by) taking each class as the reference class in turn (i.e., making it class 0 and all other classes class 1). The weight of a class's AUC is the class's frequency in the data.

This means that if a class is not represented at all (0 instance of the class in the data), its weight is 0 and there is no need to compute an AUC or to return an error. Which is what was implemented.

5 Experimental evaluation

This section describe the different steps that were followed during the experimental evaluation, which make for a benchmark of the method described in this report.

5.1 Datasets selection

To realise the benchmark of our algorithm, a large share of the collection of regression datasets from the UCI machine learning repository was selected. In total, 35 datasets were used, among which 24 datasets are formed of more than 10,000 instances since large datasets were the main focus of the study. Indeed, Orange is a commercial company that deals with large amounts of data on a day to day basis, and has little to no application for small sized datasets. So the 11 remaining datasets ranged from 1,030 to 9,568 instances.

This choice of datasets was largely influenced by the paper of M. Fernandez-Delgado ([14], 2019). This paper is a comparison of 77 popular regression models on 48 datasets. For this reason, datasets and regression algorithms they tested were selected so their paper would be a baseline of the performance that should be expected for each algorithm.

Original name	Dataset name	#instances	#attributes
3D Road network	3Droad	434,874	3
Airfoil self-noise	airfoil	1,503	5
Air quality	air-quality-CO	1,230	8
Appliances energy prediction	appliances-energy	19,735	27
Beijing PM2.5	beijing-pm25	41,758	11
Bias correction of numerical prediction model...	temp-forecast-bias	7,752	24
Bike sharing	bike-hour	17,379	16
Blog feedback	blog-feedback	60,021	13
Buzz in social media	buzz-twitter	583,250	77
Combined cycle power plant	combined-cycle	9,568	4
Communities & crime	com-crime	1,994	122
Communities & crime unnormalized	com-crime-unnorm	2,215	126
Concrete compressive strength	compress-stren	1,030	8
Condition based maintenance of naval propulsion plants	cond-turbine	11,934	13
Cuff-less blood pressure estimation	cuff-less	61,000	2
Electrical Grid Stability Simulated	electrical-grid-stab	10,000	13
Facebook comment volume	facebook-comment	40,949	48
Geographical original of music	geo-lat	1,059	72
Greenhouse gas observing network	greenhouse-net	955,167	15
Individual household electric power consumption	household-consume	2,049,280	5
KEGG metabolic reaction network (undirected)	KEGG-reaction	65,554	25
KEGG metabolic relation network (directed)	KEGG-relation	54,413	17
Online news popularity	online-news	39,644	55
Online video characteristics and transcoding time	video-transcode	68,784	8
Parkinsons telemonitoring	park-total-UPDRS	5,875	16
Physicochemical properties of protein tertiary structure	physico-protein	45,730	9
PM2.5 Data 5 Chinese Cities	pm25-chengdu-us	27,368	13
Production quality	production-quality	29,184	18
Relative location of CT slices on axial axis	CT-slices	53,500	355
Seoul Bike Sharing Demand	seoul-bike-sharing	8,760	13
SGEMM GPU kernel performance	gpu-kernel-perf	241,600	17
SML2010	SML2010	4,137	18
Uber location price	uber-location-price	205,670	7
UJIIndoorLoc	UJ-lat	21,048	373
YearPredictionMSD	year-prediction	515,345	90

Table 3: Datasets description. The first column refers to the original name on the UCI repository, the second is the sub-dataset name that is used and the last two columns designate the number of instances in the dataset and the original number of attributes (or features).

Some datasets included multiple sub-datasets with different target variables, and so different regression problems. It was chosen not to include multiple regression problems from the same datasets because it would give more weight to some datasets and create a bias in the comparison of the methods that could favor one method over another, which is undesirable. Thus, refer to the column *"dataset name"* that designates which regression problem was used for each dataset.

Dataset name	Target	Removed columns	#attributes	Initial R^2
3Droad	altitude: 3	Ø	3	0.03
appliances-energy	appliances: 0	0	27 → 26	0.32
beijing-pm25	PM2.5: 4	8	11 → 14	0.39
bike-hour	count: 8	0, 1, 7, 11, 14, 15	16 → 17	0.49
blog-feedback	target: 18	0-49, 54, 59, 62-269, 277-279	280 → 18	0.31
buzz-twitter	discussions: 70	70-76	77 → 70	0.49
combined-cycle	PE: 4	Ø	4	0.93
cond-turbine	gt turbine: 15	8, 17	17 → 15	0.91
cuff-less	APB: 1	Ø	2	0.36
facebook-comment	target: 53	Ø	53	0.54
greenhouse-net	synthetic var: 15	16	16 → 15	0.58
household-consume	power consumption: 12	Ø	8 → 12	0.77
KEGG-reaction	edge count: 27	0	28 → 27	0.97
KEGG-relation	clustering coef: 22	0	23 → 22	0.60
online-news	shares: 59	0	60 → 59	0.13
video-transcode	utime: 18	0	21 → 26	0.84
pm25-chengdu-us	PM_US Post: 4	0, 6, 7	16 → 20	0.36
physico-protein	RMSD: 0	Ø	9	0.29
CT-slices	reference: 384	0	385 → 384	0.87
UJ-lat	latitude: 520	520	528 → 527	0.97
park-total-UPDRS	total_UPDRS: 0	0, 1, 2, 3, 4	21 → 16	0.11
SML2010	indoor temp: 0	0, 1, 3	23 → 26	0.94
com-crime-unnorm	Violent crimes: 124	1-5, 130-145, 147	146 → 134	0.64
com-crime	Violent crimes: 122	1-5	127 → 122	0.73
year-prediction	Year: 0	Ø	90	0.27
airfoil	scaled sound: 5	Ø	5	0.49
air-quality-CO	PT08.S1(CO): 1	0, 1, 6, 8, 10, 11	14 → 8	0.89
geo-lat	latitude: 116	116	117 → 116	0.37
compress-stren	compressive strength: 8	Ø	8	0.61
seoul-bike-sharing	Rented Bike Count: 0	0, 11, 12, 13	13 → 9	0.51
gpu-kernel-perf	Run1 (ms): 14	15, 16, 17	17 → 14	0.60
uber-location-price	fare amount: 0	1	6 → 5	0.01
electrical-grid-stab	stab: 12	13	13 → 12	0.64
production-quality	quality: 17	0	18 → 17	0.76
temp-forecast-bias	Next_Tmax: 22	1, 24	24 → 22	0.78

Table 4: Datasets Information. The first column is the dataset's name and the second is the name and the index of the target variable after the pre-processing of the dataset. The third column lists the indexes of the attributes that were removed in the pre-processing (Red = dates, blue = other target variables, green = indentifiers, orange = text, brown = constants, magenta = redundant information). The fourth column counts the number of attributes before pre-processing (left side of the arrow) and after pre-processing (right side of the arrow). If there is a single number, it means no column was added or removed.

Each dataset's quantity of missing values was studied beforehand, and since they never exceeded more than 8% of the total size of the dataset, the instances which had missing attributes were simply removed. Only for both *com-crime-unnorm* and *com-crime* datasets this share of missing values accounted up to 84% for some attributes so it was decided to fill the missing values with the mean of each column. In order to be able to compare the results to the benchmark [14], the attributes that were removed by authors were also deleted from the datasets (it

was mainly dates, constant, collinear columns and alternative target variables).

5.2 Hyperparameters selection

During the experimentation, it was observed that regression models with default hyperparameters often overfitted and / or displayed poor performance. So a grid search with the Root Mean Square Error (RMSE) as an indicator for model performance has been set up for every regressor to find better hyperparameters on each dataset before training. These grid searches explored the parameters as described in table 11 from annex F. Concerning Random Forest, it was chosen not to optimize the parameter "n_estimators" which corresponds to the number of trees in the forest. Indeed, after a first study, it appeared that this parameter had very little impact on the overall test performance of the Random Forest across all of the datasets. So it was removed from the grid, which allowed to more finely optimize the other parameters since the computing time was reduced.

It is important to add here that a grid search was executed for each dataset and for each regressor. Which means that a grid search was executed for the original datasets used to compute the baselines and for their *extended* versions used to evaluate our method. And the same values were explored between the grids of baselines and the grids of the extended versions.

For a more robust evaluation of the hyperparameters, the grid searches were executed on each of the 10 training subsets of the 10-fold cross validation. And for each fold, one third of the training subset was dedicated to hyperparameter tuning and two thirds were dedicated to the actual training of the regressors. This means that because 3 regression models required tuning of their hyperparameters, $3 \times 10 = 30$ grid searches were executed for every dataset, so $30 \times 35 = 1,050$ in total. The linear regression model of scikit-learn uses the Ordinary Least Squares (OLS) method to estimate the parameters, and the Khiops regressor is an Automated Machine Learning (AutoML) model, so none of them have hyperparameters to tune.

In table 8 of annex C, the performance of the Random Forest regressor in training and testing were recorded with and without the use of a grid search to tune the hyperparameters. A relatively small decreased in performance can be observed in most of the datasets. This is presumably because the default value of the parameter max_depth is None, which means that the trees can grow to their maximum size. So each node of the trees can have as low as two instances to make their split. This is probably a cause for overfitting, but it still results in a improvement of the performance, compared to the grid search that limits the depth of the trees to 32.

Note: It was chosen not to tune the hyperparameters of the classifiers used to extract the features. Indeed, it was observed (see section 5.3) that the classifiers had very high performance when the binary classification with multiple classifiers method was used (a mean AUROC of 0,95 across all number of thresholds).

5.3 Classes generation and method selection

In order to determine which of the methods described in section 4 was going to be used to generate the results, the multi-class and binary classification methods were compared. It was chosen not to explore the multi-label method because not all classification models are able to produce multi-label outputs, but it would certainly be interesting to compare it as well.

To chose the method, the Ratio of the test ROC AUC score divided by the train ROC AUC score was compared across a number of thresholds for each method. A ratio closer to 1 obviously means that there is little overfitting of the classifiers and not necessarily better performance, but as the minimum train ROC AUC score is 0,9944 across all the datasets and methods, this ratio can also be interpreted as an indicator of test performance.

	Multi-class classification						Binary classification				
Number of bins:	**2**	**4**	**8**	**16**	**32**		**2**	**4**	**8**	**16**	**32**
3Droad	1,00	1,00	0,99	0,98	0,96		1,00	1,00	1,00	1,00	1,00
air-quality-CO	0,98	0,95	0,93	0,90	0,87		0,98	0,98	0,98	0,98	0,98
airfoil	0,97	0,93	0,85	0,79	0,72		0,98	0,98	0,98	0,98	0,98
appliances-energy	0,91	0,90	0,86	0,85	0,82		0,96	0,94	0,93	0,93	0,92
beijing-pm25	0,96	0,93	0,90	0,85	0,79		0,96	0,96	0,96	0,96	0,96
temp-forecast-bias	0,98	0,96	0,93	0,90	0,87		0,98	0,98	0,98	0,98	0,98
bike-hour	0,98	0,96	0,92	0,89	0,85		0,98	0,98	0,98	0,98	0,98
blog-feedback	0,86	0,86	0,84	0,81	0,82		0,88	0,89	0,90	0,91	0,92
buzz-twitter	0,98	0,96	0,94	0,92	0,90		0,99	0,98	0,98	0,98	0,98
combined-cycle	0,99	0,98	0,96	0,94	0,91		0,99	0,99	0,99	0,99	0,99
com-crime	0,91	0,83	0,80	0,76	0,74		0,91	0,91	0,92	0,91	0,91
com-crime-unnorm	0,88	0,83	0,79	0,77	0,75		0,88	0,90	0,90	0,89	0,89
compress-stren	0,97	0,94	0,90	0,85	0,80		0,98	0,98	0,98	0,98	0,97
cond-turbine	1,00	1,00	0,99	0,98	0,95		1,00	1,00	1,00	1,00	1,00
cuff-less	0,92	0,85	0,80	0,76	0,70		0,92	0,92	0,91	0,90	0,89
electrical-grid-stab	0,98	0,92	0,87	0,82	0,77		0,98	0,98	0,98	0,98	0,97
facebook-comment	0,91	0,89	0,87	0,86	0,85		0,91	0,94	0,95	0,96	0,96
geo-lat	0,78	0,81	0,82	0,82	0,84		0,80	0,83	0,80	0,82	0,83
greenhouse-net	0,83	0,80	0,78	0,76	0,74		0,89	0,90	0,90	0,90	0,90
household-consume	1,00	1,00	1,00	1,00	1,00		1,00	1,00	1,00	1,00	1,00
KEGG-reaction	1,00	1,00	1,00	1,00	1,00		1,00	1,00	1,00	1,00	1,00
KEGG-relation	1,00	0,99	0,99	0,98	0,97		1,00	1,00	1,00	1,00	1,00
online-news	0,68	0,65	0,63	0,60	0,59		0,72	0,72	0,72	0,72	0,72
video-transcode	1,00	1,00	0,99	0,99	0,97		1,00	1,00	1,00	1,00	1,00
pm25-chengdu-us	0,95	0,91	0,88	0,83	0,77		0,95	0,95	0,95	0,95	0,95
park-total-UPDRS	0,82	0,82	0,80	0,79	0,77		0,82	0,83	0,83	0,83	0,84
physico-protein	0,96	0,93	0,91	0,89	0,87		0,96	0,95	0,95	0,95	0,95
production-quality	0,98	0,95	0,92	0,90	0,87		0,98	0,98	0,98	0,98	0,98
CT-slices	1,00	1,00	1,00	1,00	1,00		1,00	1,00	1,00	1,00	1,00
gpu-kernel-perf	1,00	1,00	1,00	0,99	0,99		1,00	1,00	1,00	1,00	1,00
SML2010	1,00	1,00	1,00	0,99	0,96		1,00	1,00	1,00	1,00	1,00
seoul-bike-sharing	0,96	0,93	0,91	0,88	0,85		0,96	0,95	0,95	0,95	0,95
UJ-lat	1,00	1,00	1,00	1,00	1,00		1,00	1,00	1,00	1,00	1,00
uber-location-price	0,91	0,88	0,85	0,82	0,79		0,93	0,94	0,94	0,94	0,94
year-prediction	0,80	0,73	0,70	n/a	n/a		0,80	0,79	0,77	0,79	0,79
Mean:	0,94	0,92	0,89	0,88	0,85		0,95	0,95	0,95	0,95	0,95

Table 5: Ratio of the test ROC AUC score divided by the train ROC AUC score of the Random Forest classifier (with default parameters) when trained on the datasets which were either associated a single class number for the multi-class case, or a class vector of zeroes and ones in the binary classification case. Since multiple classifiers were trained for binary classification (one for each threshold), this is the mean score. For both cases, the results are the mean of a 10 fold cross-validation.

It can be observed in table 5 that the mean ROC AUC score ratio of the multi-class configuration steadily decreases as the number of thresholds increases. This is expected behavior because as the number of thresholds increases, so does the number of possible classes and the classification problem gets more difficult. It can also be seen as a finer and finer approximation of the actual target value of the regression problem. On the other hand, the performance of the binary problem configuration remains very good (around 0,95) regardless of the number

of thresholds. This can be explained by the fact that this method uses a different classifier for each threshold that is defined, so the difficulty of the task remains the same for any number of thresholds. This means that the multi binary classifiers method will consistently do a better job at approximating the target value of the instances. Which is why it was decided to only use the binary class generation method to generate the results for our method. Generating the results with the 3 class generation methods defined in section 4.2.1 and for all the classifiers initially considered could lead to better performance in some cases, but computation time is the limiting factor which led to this choice.

Note: As illustrated in figure 11 of section 4.3.2, as the number of thresholds increases, the probability that the equal frequency discretization method wont be able to define as many thresholds as there were initially required increases. This is an issue that happens more frequently in datasets that have many target values that are of the same value. In some cases, it was possible to define 32 thresholds with no trouble, and in some other only half of the thresholds could be defined. This limitation is part of the reason that the learning curve observed in the experimentation slows down as the number of thresholds increases.

5.4 Experimental protocol

The experimental work that was set up to generate the results uses the following methodology: The first step is the pre-processing of the datasets, which can be split into 3 smaller sub-steps:

1. The *cleaning* of all the datasets, which includes: The deletion of unwanted variables, the one-hot encoding, the deletion of instances with missing values and the identification of the target variable.
2. The split of the datasets using k-folds into 10 test and train sub-datasets
3. The actual "pre-processing" of the datasets, where the Box-Cox transformation [3], the normalization and the definition of the thresholds (by equal-frequency) for our method are all fitted to each training dataset and applied to both the training and testing datasets. As described in section 3.2.5.

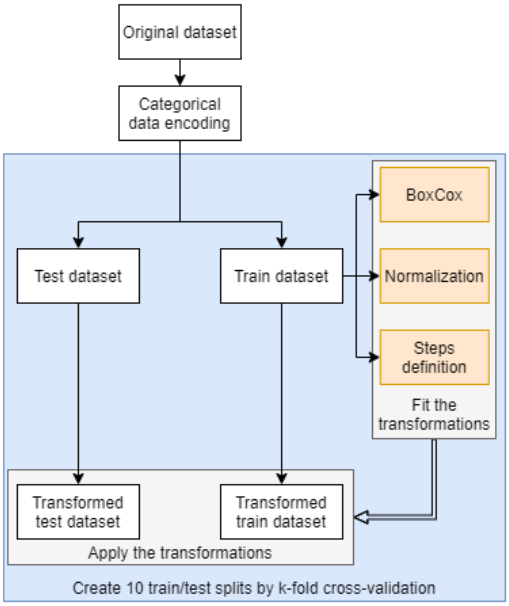


Figure 13: Datasets pre-processing

The next step is to define the classifiers following the method that was selected (so in this case, it is the multi-classifiers where each one has to resolve a binary classification problem). Then, train each one on every training sub-dataset and make a prediction for the training and testing sets to extract the features (during our experimentation, the only extracted features were the conditional probabilities of the prediction).

At this point, the regression models can be trained, either on the non-extended datasets in order to compute a baseline to compare our method, or on the extended datasets. But just before training, if the regressor has hyper-parameters that need to be tuned, we split the training dataset in a validation (one third) and a training (two thirds) subset, then apply a grid-search on the validation dataset (as discussed in section 5.2). Once they have been trained and a prediction has been made on the training and testing datasets, the multiple metrics that were used in this report can be computed (log loss and AUROC for the performance of the classifiers and RMSE for the regressors).

This process is repeated on each of the 35 datasets and for each of the sub-datasets of the 10-fold was defined. This way, the results are averaged on the 10 sub-datasets for both testing and training for the metrics and the graphs that are generated.

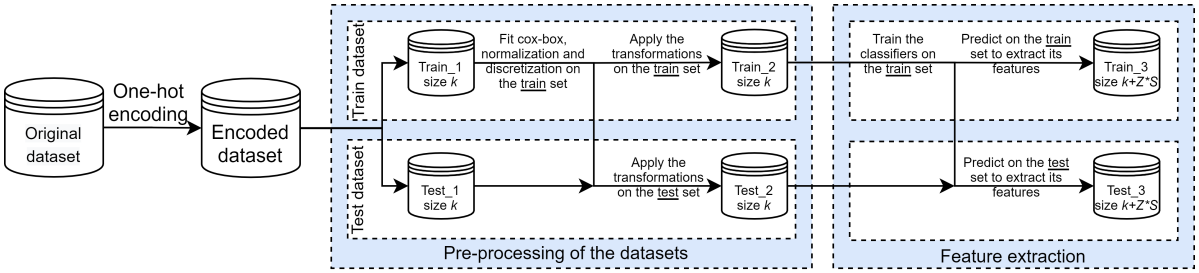


Figure 14: Dataset flowchart

Figure 14 is a summary of all the main steps that are used during the experimental protocol that was followed for generating the results of this report.

5.5 Results

The experimental protocol that was described in the previous section was executed for each of the 35 datasets, and for each of the 5 regression models considered in this report (Linear Regression, Decision Tree, Random Forest, XGBoost and Khiops). Each regression model has been evaluated without the usage of our method to act as a baseline of the performance, then we recorded the performance of our method by varying the number of thresholds of the equal-frequency discretization process for 2, 4, 8, 16 and 32 thresholds. This procedure therefore developed a total of $35 \times 5 \times (1 + 6) = 1,050$ evaluations. Which accounts for $1,050 \times 10 = 10,500$ grid searches and training phases. Theses experiments were carried out on two different machines: a calculation server equipped with an Intel Xeon Gold 6150 processor with 36 cores and 192 GB of RAM, and on a personal desktop computer equipped with a AMD Ryzen 9 3900X with 12 cores and 32 GB of RAM.

The main metric of evaluation in this section is the RMSE (see section 3.3.3). And the comparison of the results is done using the Nemenyi test after a Friedman rank of the RMSE. The Friedman test is a non-parametric test that ranks the models that are being compared for each dataset independently. The best model will get the rank 1, the second rank 2, ... This test can be used when the following assumptions are verified:

- The group of samples used for the comparison is a random sub-sample of the population.
- There is no interaction between the samples and the methods.
- 3 or more groups are being compared.
- The data is ordinal or continuous.

The null-hypothesis H_0 states that the methods are equivalent. So the Friedman test will aim to determine if there is or not a significant difference between the methods for a given confidence interval α (usually 0.05). To do so, this first step is to find the chi-square value (χ^2) using the Friedman table of critical values with $\alpha = 0.05$ and degree of freedom = [number of groups - 1]. Then, compute the Friedman statistic using:

$$\chi_F^2 = \frac{12}{Nk(k+1)} \sum_{j=1}^{k} (R_j^2 - 3n(k+1))$$

With R_j the sum of the ranks of instance j, k the number of groups and N the number of models. And if the computed value χ_F^2 is greater than the χ^2 value from the table, the null-hypothesis that the groups are not statistically different can be rejected.

Once the null-hypothesis has been rejected, we can use a post-hoc test (i.e., a test that is conducted after the data were seen) to determine which groups are different from each other. The Nemenyi test falls under these criteria since it is a post-hoc test whose goal is to find groups of data that are different from each other after a statistical test (in this case the Friedman test). To conduct the Nemenyi test, we must first determine the mean of the Friedman ranks for each method. And then compute the critical difference with:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

With q_α the critical value of the two-tailed Nemenyi test which can also be found in any statistical book. If the difference between the mean ranks of two methods is greater than the computed CD, we can consider that they are statistically different.

Table 6 below summarises the results obtained by our method for each regression model and each dataset of the study, and compares these results to the values that the regression model obtained without our method.

Dataset name	Linear Regr.		Decision Tree		Random Forest		XGBoost		Khiops	
	Base RMSE	**New RMSE**	**Base RMSE**	**New RMSE**	**Base RMSE**	**New RMSE**	**Base RMSE**	**New RMSE**	**Base RMSE**	**New RMSE**
3Droad	0,9867	0,0930	0,1213	0,0831	0,0908	0,0792	0,0935	0,0781	0,5580	0,0827
air-quality-CO	0,3269	0,2672	0,3255	0,2727	0,2667	0,2642	0,2682	0,2633	0,3428	0,2768
airfoil	0,7176	0,2351	0,4373	0,3194	0,2987	0,2865	0,2814	0,2752	0,7709	0,2963
appliances-energy	0,8260	0,4865	0,7431	0,5267	0,5456	0,5164	0,5790	0,5203	0,8354	0,5206
beijing-pm25	0,7833	0,4180	0,6013	0,3948	0,4168	0,3874	0,4149	0,3885	0,8082	0,3931
temp-forecast-bias	0,4749	0,2847	0,4668	0,2848	0,3237	0,2771	0,3085	0,2773	0,4782	0,2907
bike-hour	0,7163	0,2547	0,3294	0,2564	0,2415	0,2499	0,2208	0,2500	0,4858	0,2534
blog-feedback	0,8307	0,6434	0,6741	0,6694	0,6481	0,6618	0,6444	0,6652	0,7013	0,6601
buzz-twitter	0,7248	0,2183	0,2299	0,2265	0,2170	0,2200	0,2162	0,2205	0,2238	0,2191
combined-cycle	0,2702	0,1967	0,2647	0,2014	0,2079	0,1952	0,2041	0,1950	0,2570	0,1995
com-crime	0,6312	0,5485	0,6418	0,5552	0,5525	0,5505	0,5665	0,5503	0,5612	0,5523
com-crime-unnorm	0,6751	0,6031	0,7036	0,6251	0,6163	0,6157	0,6186	0,6124	0,6453	0,6340
compress-stren	0,6331	0,2732	0,4519	0,3173	0,3137	0,2896	0,2790	0,2955	0,5962	0,2959
cond-turbine	0,3002	0,1096	0,1764	0,1094	0,1117	0,1064	0,1062	0,1094	0,4481	0,1096
cuff-less	0,7996	0,5791	0,5165	0,6255	0,5025	0,6105	0,5012	0,6225	0,5450	0,5922
electrical-grid-stab	0,5961	0,3156	0,5545	0,3245	0,3352	0,3068	0,2634	0,3092	0,5951	0,3157
facebook-comment	0,6828	0,4242	0,4655	0,4429	0,4162	0,4384	0,4095	0,4397	0,5014	0,4347
geo-lat	0,8752	0,8066	0,9594	0,8538	0,8321	0,8287	0,8540	0,8353	0,8668	0,8408
greenhouse-net	0,6492	0,5418	0,5397	0,5686	0,5158	0,5638	0,5137	0,5618	0,6016	0,5599
household-consume	0,4821	0,0801	0,0680	0,0636	0,0447	0,0613	0,0388	0,0589	n/a	n/a
KEGG-reaction	0,1803	0,0845	0,0919	0,0861	0,0837	0,0835	0,0835	0,0840	0,1172	0,0855
KEGG-relation	0,6342	0,1750	0,2454	0,1748	0,1782	0,1700	0,1809	0,1715	0,4314	0,1738
online-news	1,1822	1,0677	0,9516	0,9622	0,9177	0,9521	0,9124	0,9521	0,9228	0,9942
video-transcode	0,3975	0,0561	0,0846	0,0652	0,0604	0,0597	0,0590	0,0568	0,3704	0,0627
pm25-chengdu-us-post	0,8022	0,4283	0,5578	0,4164	0,4111	0,4084	0,4125	0,4079	0,7716	0,4131
park-total-UPDRS	0,9472	0,7816	0,9416	0,8057	0,8020	0,7893	0,8196	0,7860	0,9428	0,8117
physico-protein	0,8453	0,5179	0,7676	0,5338	0,5528	0,5218	0,5744	0,5249	0,8494	0,5330
production-quality	0,4872	0,2807	0,3886	0,2830	0,2836	0,2779	0,2794	0,2781	0,3790	0,2832
CT-slices	1,9533	0,0504	0,1332	0,0586	0,0544	0,0418	0,0693	0,0355	0,1104	0,0376
gpu-kernel-perf	0,6347	0,0696	0,0300	0,0557	0,0246	0,0501	0,0228	0,0500	0,5927	0,0620
SML2010	0,2536	0,1157	0,2179	0,1036	0,1241	0,0857	0,1058	0,0920	0,3473	0,1221
seoul-bike-sharing	0,6990	0,4839	0,5801	0,5217	0,5101	0,5149	0,5171	0,5139	0,5891	0,5169
UJ-lat	~~631,59~~	~~31,226~~	0,0662	0,0414	0,0383	0,0354	0,0393	0,0377	0,0527	0,0384
uber-location-price	0,9998	0,4589	0,6153	0,4789	0,4635	0,4699	0,4705	0,4688	0,8419	0,4879
year-prediction	0,8569	0,8079	0,8783	0,8137	0,8059	0,8062	0,7929	0,8131	0,8822	0,8138
Average:	0,7016	0,3752	0,4520	0,3749	0,3659	0,3650	0,3635	0,3657	0,5595	0,3813
Mean reduction:	46,52%		17,06%		0,25%		-0,62%		31,85%	

Table 6: Comparison of the test RMSE of each regressor with and without our method. "Base RMSE" designates the test performance of the regressors without our method, while "New RMSE" designates the test performance of the regressor on the datasets that were augmented using our method with 32 thresholds. The features were extracted using a Random Forest classifier for the 5 regressors. All the predictions were made after a grid search of the optimal hyper-parameters in table 11 of annex F. All of the results are averaged over a 10 fold cross-validation. The values obtained by the Linear Regression model for the dataset *UJ-lat* were not used to compute the mean since they were abnormally large.

Note: The lines 'Average' and 'Mean reduction' are to be taken with a grain of salt. Because every dataset is a problem of unique difficulty, the scale of the errors differ between each dataset. So if the new test rmse is higher than the base test rmse, or if the mean reduction is negative, it does not necessarily mean that the regressors are performing worse.

Using this table, the following figure was drawn. It compares the performance of the regressors without using our method (so only the "Base RMSE" columns were used).

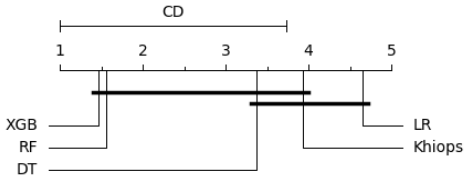


Figure 15: Comparison the regressors against each other with the Nemenyi test. Rank 1 designates the best method and the horizontal lines group together methods that are not significantly different according to the Nemenyi test (with $p = 0.05$).

Considering all of the datasets, it can be observed in figure 15 that the default version of XGBoost displays the best performance by achieving the lowest RMSE for the most datasets, closely followed by Random Forest. Both of these regressors widely outperformed the 3 other regressors, with Linear Regression unsurprisingly being the worst performing model with the highest RMSE for the most datasets. It is closely followed by Khiops and Decision Tree that performed comparably.

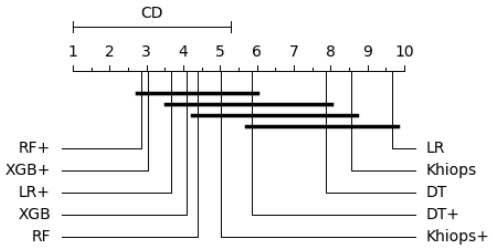


Figure 16: Comparison of all the regressors against each other, with and without our method for 32 thresholds, with the Nemenyi test. A '+' indicates that the model was augmented using our method.

Figure 16 compares the performance of all of the regressors, with and without our method. This time again, two distinct groups of models are present, with the best group containing both the default and augmented version of XGBoost and Random Forest, as well as the augmented Linear Regression. Figure 16 also shows that no regressor was negatively impacted by our method, as all the regressors that used our method are ranked higher than their basic versions. The Linear Regression displays the most impressive improvement, with a mean rank of 9.4 for the basic version and a mean rank of 3.4 when using our method, coming very close to the Random Forest and even XGBoost regressors that used our method. This was initially surprising, as Linear Regression is known to be negatively impacted by collinear features. The way the features were extracted using classifiers in the implementation of our method added both conditional probabilities of each binary classifier. In other words, for each threshold, the values $P(C_0|X)$ and $P(C_1|X)$ were added for the extension of the dataset. And they are obviously perfectly collinear since $P(C_0|X) = 1 - P(C_1|X)$. However, collinear variables do not have a negative impact on the performance of Linear Regression. They only impair the interpretability of the model, as explained in section 3.2.2. This was confirmed when 20 of the 35 datasets were used to compare the performance of the regressors when the collinear extracted features were removed from the extended datasets. Table 9 of annex E compares theses results to the original performance where both conditional probabilities were added, and none of the regressors showed that this had a negative nor positive impact on performance.

In the same way as for the other figures, figure 16 connects with horizontal lines the methods that are not significant different from each other according to the Nemenyi test for $p = 0.5$. Methods are only considered significantly different when the difference between their average ranks is greater than the critical difference computed by the Nemenyi-test. However, the Nemenyi test seems to compute a rather large critical difference. For instance,

on figure 16, the augmented version of Decision Tree (DT+) and the Linear Regression (LR) are connected despite DT+ having a mean test RMSE of 0.3749 and LR having a mean test RMSE of 0.7016, which is 1.87 times larger. To determine if our method had a significant impact for each regressor individually, a Wilcoxon sign-ranked test was conducted. This test is used to determine if there is a significant difference on scores when there is a "before" and "after" condition of some treatment (so in this case, using our method to extend the datasets). The "wilcoxon" method from the *scipy.stats* library calculates the pvalue, which corresponds to the likelihood of getting data as extreme as the observed values assuming that the null hypothesis is true. So here, a pvalue larger than 0.05 means that the compared samples are not significantly different.

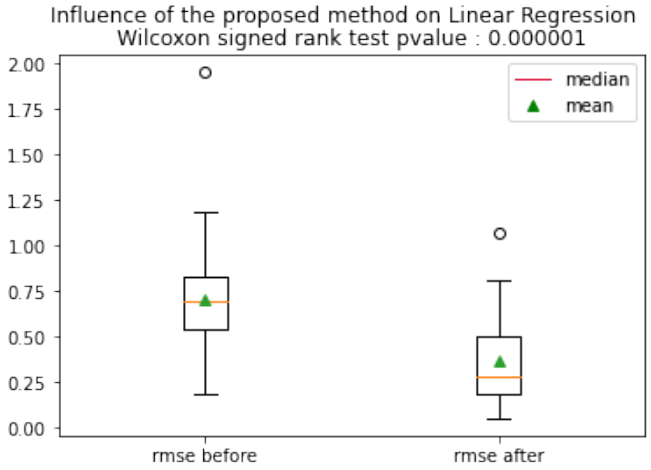


Figure 17: Box plot of the RMSE of the **Linear Regression** on the 35 datasets without (label 'rmse before') and with (label 'rmse after') our method for 32 thresholds to extract features with the Random Forest classifier.

Figure 17 is the comparison of the test RMSE samples obtained by a Linear Regression with and without our method on all of the datasets. The sub-title of the figure contains the pvalue of the Wilcoxon sign-ranked test. In this case it less than 0.05, which means that the Linear Regression has a significantly better performance when using our method. This is also the case for the Decision Tree and Khiops regressors, but not for the Random Forest and XGBoost regressors (they had a pvalue of 0.104 and 0.357 respectively). Please refer to annex D to find the box plots of the 4 other regression models.

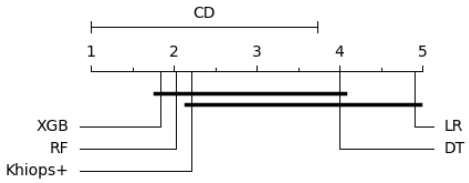


Figure 18: Comparison of Khiops with our method (using Random Forest classifier) against the state of the art regressors with the Nemenyi test.

Figure 18 compares the test performance of the Khiops regressor with our method against the other regressors used in this report. While the initial performance of the Khiops regressor was comparable to that of the Decision Tree regressor (see fig. 15), once augmented using our method, Khiops displays a performance very close to that of Random Forest and XGBoost. This is a huge improvement, as the test RMSE was reduced by 31.85% on average

across all of the datasets. The Khiops software allows to visualize which variables are selected by the Khiops regressor after fitting it to the training data. And it was discoverd that only the extracted features were used in the final model. This means that all of the original features were discarded because their level of informativity was too low compared to those of the extracted features. Khiops is also the regressor that benefited the most consistently of the extraction of new features using our method, as even the use of 2 thresholds was enough for the Khiops regressor to display an immediate improvement of performance. The Linear Regression and Decision Tree regressors also benefited of the use of our method for even for 2 thresholds quite regularly, but Khiops was the most consistent with only 1 of the 35 datasets (*geo-lat*) having a higher test RMSE for 2 thresholds when using our method.

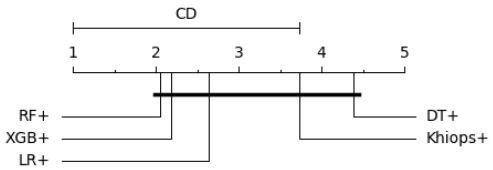


Figure 19: Comparison of all of the regressor with our method against each other with the Nemenyi test.

When comparing all of the regressors with Random Forest classifier to extract new features using our method, it is clear that Linear Regression is the method that stands out the most. While Random Forest and XGBoost stayed relatively at the same position, Linear Regression joined their group as one of the best performing models. And Khiops benefited of the new features a bit more than Decision Tree.

6 Discussion

To gain additional insight about the mechanisms of the method proposed in this report, some additional experiments will be performed in this section. The dataset *Combined Cycle Power Plant* is going to be the subject of these experiments.

	Original dataset		$X \cup X'$		X' **only**	
Regressor	**Train RMSE**	**Test RMSE**	**Train RMSE**	**Test RMSE**	**Train RMSE**	**Test RMSE**
Linear Regression	0,2702	0,2702	0,0654	0,1964	0,0792	0,1989
Decision Tree	0,2088	0,2647	0,0325	0,2008	0,0341	0,1996
Random Forest	0,0841	0,2079	0,0315	0,1953	0,0332	0,1954
XGBoost	0,0961	0,2041	0,0322	0,1950	0,0333	0,1955
Khiops	0,2510	0,2570	0,0489	0,1995	0,0489	0,1995

Table 7: Comparison of the performance of the regressors on the Combined Cycle Power Plant dataset for different augmentations using our method.

Table 7 records the performance of the five regressors of this paper on the dataset Combined Cycle Power Plant under different conditions. The first column, 'Original dataset', represents the performance of the regressors on the un-augmented (original) dataset. The second column, '$X \cup X'$', records the performance of the regressors on the dataset that was augmented using our method with a Random Forest classifier and 32 thresholds. The last column, 'X' only', is the performance of the regressors trained on a dataset composed only of the conditional probabilities extracted using our method with a Random Forest classifier.

In the column 'Original dataset' of table 7, it can be observed that Linear Regression and Decision Tree are the two worst performing regressors, closely followed by Khiops. Random Forest and XGBoost show the best performance, as it was seen in table 6 of section 5.5. Once the method proposed in this paper is used, the results in the column '$X \cup X'$' show a significant improvement for Linear Regression, Decision Tree and Khiops, while Random Forest and XGBoost improve only slightly. The last column, 'X' only', shows no difference in test performance with the previous column for all regressors. This proves that the features extracted using our method are highly informative and that the regressors rely almost exclusively on them to make their prediction.

It was not mentioned before in this report, but by extracting the conditional probabilities of the position of an instance relative to each threshold, we are (attention au language 'parlé') in reality estimating the conditional Cumulative Distribution Function (conditional CDF). The CDF represents the probability of observing $Y < S_i$, with Y a random variable and S_i a threshold, and is noted :

$$F_Y(S_i) = P(Y \leq S_i) = \int_{-\infty}^{S_i} P(Y)\, dY$$

The classifiers are estimating the conditional CDF. And with $Y \in [a; b]$ it is noted as :

$$F_{Y|X}(S_i) = P(Y < S_i|X) = \begin{cases} 1 & \text{if } S_i > b \\ \frac{F_Y(S_i) - F_Y(a)}{F_Y(b) - F_Y(a)} & \text{if } a \leq S_i \leq b \\ 0 & \text{otherwise} \end{cases}$$

To determine if the classifiers are accurately estimating this conditional CDF, the empirical and estimated conditional CDF will be plotted for each threshold S_i. The empirical conditional CDF is computed as $\frac{N_1}{N_1+N_2}$, with N_1 the number of instances less than or equal to S_i, and N_2 the number of instances great than y. And the estimated conditional CDF of the threshold S_i is computed as the mean of the predicted $P(C_0|X)$ of all of the instances of the dataset *Combined Cycle Power Plant* for the classifier i.

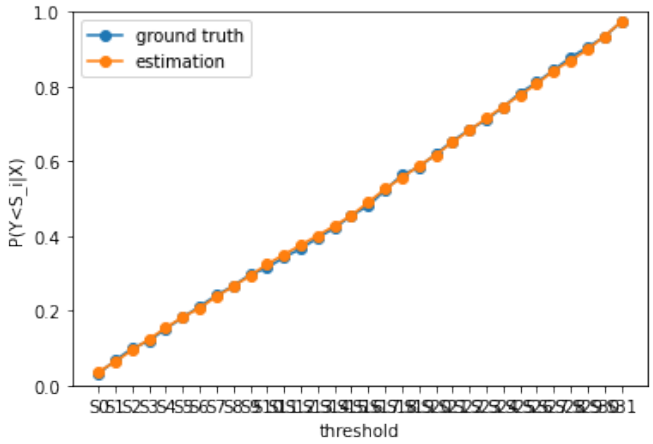


Figure 20: Empirical and estimated conditional cumulative distribution function of the classifiers on a test sample of the dataset Combined Cycle Power Plant.

Figure 20 was plotted following the instructions above. *Ground truth* corresponds to the empirical conditional CDF and *estimation* to the conditional CDF estimated by the classifiers. Since the test ROC AUC score was very high in all datasets, the estimated conditional CDF is unsurprisingly very close to the ground truth. This means that the Random Forest classifiers accurately estimate the cumulative distribution function, and this explains why the features extracted by our method contain so much information.

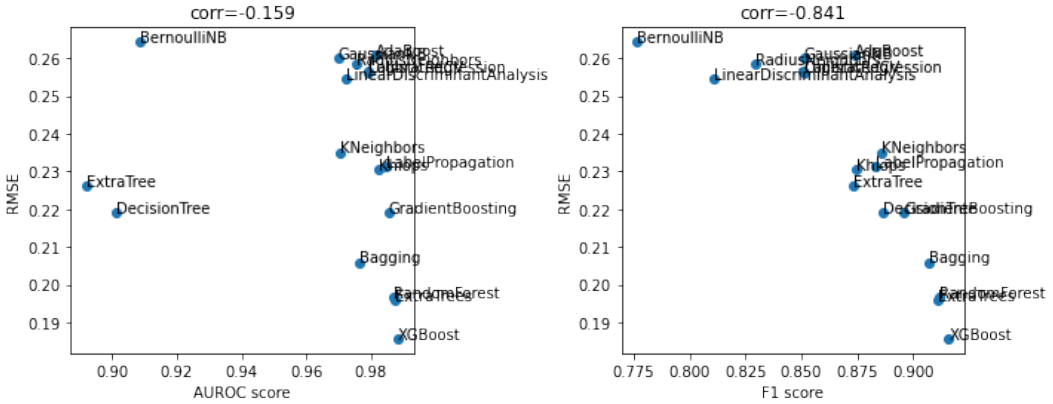


Figure 21: Each point corresponds to the performance of a different classification model used for the extraction of features with our method, against the RMSE of a **Linear Regression** fitted on the extended *Combined Cycle Power Plant* dataset. The AUROC and F1 scores are the mean of the 32 classifiers defined on each thresholds. All the scores are averaged on the folds of a 10-fold cross validation.

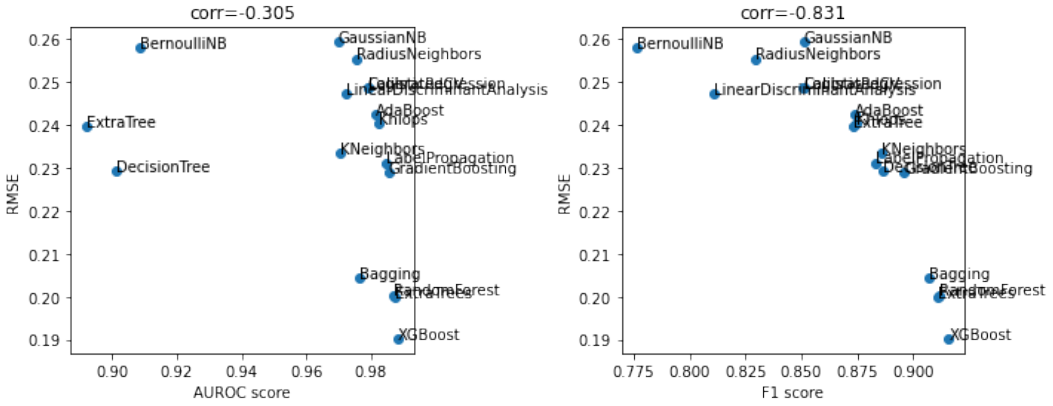


Figure 22: Each point corresponds to the performance of a different classification model used for the extraction of features with our method, against the RMSE of a **Khiops** fitted on the extended *Combined Cycle Power Plant* dataset. The AUROC and F1 scores are the mean of the 32 classifiers defined on each thresholds. All the scores are averaged on the folds of a 10-fold cross validation.

Sentence here to explain that the classifiers needed to support the method 'predict_proba" to extract the conditional probabilities. Some of them could not be used on the Combined Cycle Power Plant dataset as they did not support negative values in the features. The 17 remaining classification models that were used to extract the features are : AdaBoostClassifier, BaggingClassifier, BernoulliNB, CalibratedClassifierCV, DecisionTree, ExtraTreeClassifier, ExtraTreesClassifier, GaussianNB, GradientBoostingClassifier, Khiops, KNeighborsClassifier, LabelPropagation, LinearDiscriminantAnalysis, LogisticRegression, RadiusNeighborsClassifier, RandomForest and XGBoost.

1. Is our method estimating correctly the cumulative density function ?
   - How is the performance if we use only X' for the prediction / How much information does X' contains ?
     - Compare the performances of each regressor with only X vs $X \cup X'$
     - (optional) Check the weights of a Linear Regression with $X \cup X'$
     - Conclusion should be : X' brings the most information
   - Now how well are we estimating the density function ?
     - We can visualize it by plotting for each threshold $P(Y < S|X)$ and compare to the reality.
2. Is there a correlation between the quality of the classification and the improvement that our method brings ?
   - Check the correlation between the F-score or the AUROC and the performance impact.
3. Is the impact on performance of our method better when we apply f-1 before computing the RMSE ?
   - Re-compute the Box-Cox on all of the datasets and extract the lambda for each dataset.
   - This way, no need to re-train and predict every classifier/regressor

Example of discussion in http://vincentlemaire-labs.fr/publis/supplementary_material_pakkd_2020.pdf

Future work :

- Other variables to extract

- Test other pair of classifiers/regressors
- Training the classifiers on the same dataset that was used to define the thresholds could potentially induce overfitting
- ...

7 Project and scientific approach aspects of the study

7.1 Planning and scheduling

Being the internship of the fifth year of my Engineering Degree, the aspects of planning and scheduling are important, as they are taught during the courses and are meant to be developed during the multiple internships of the degree. During this internship, I was given many tasks, of different degrees of complexity, and sometimes many at the same time. For these reasons, I needed to have good planning throughout the internship. The tool on which I relied the most on a daily basis to achieve this planning was Trello. Trello is an online project management tool. It is based on an organization of projects in lists of cards, with each card representing a single task.

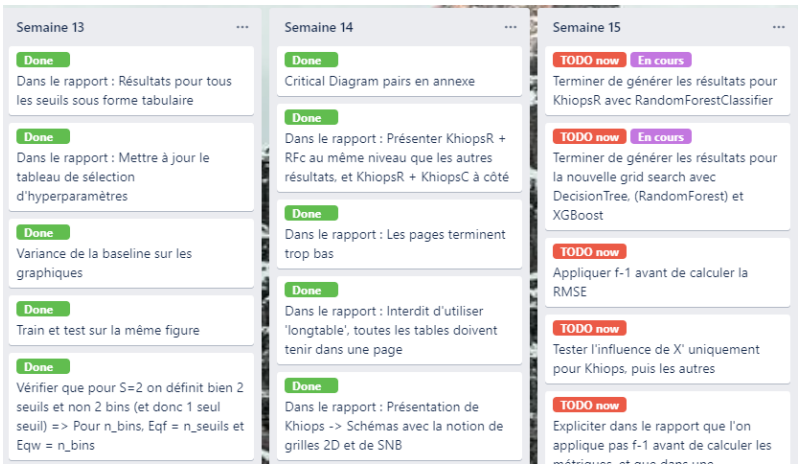


Figure 23: Example of Trello cards in the internship's board

By using colored tags on the cards, it was easy to keep track of which tasks needed to be done or were already a work in progress. And the internship tutor was able to monitor my progression easily.

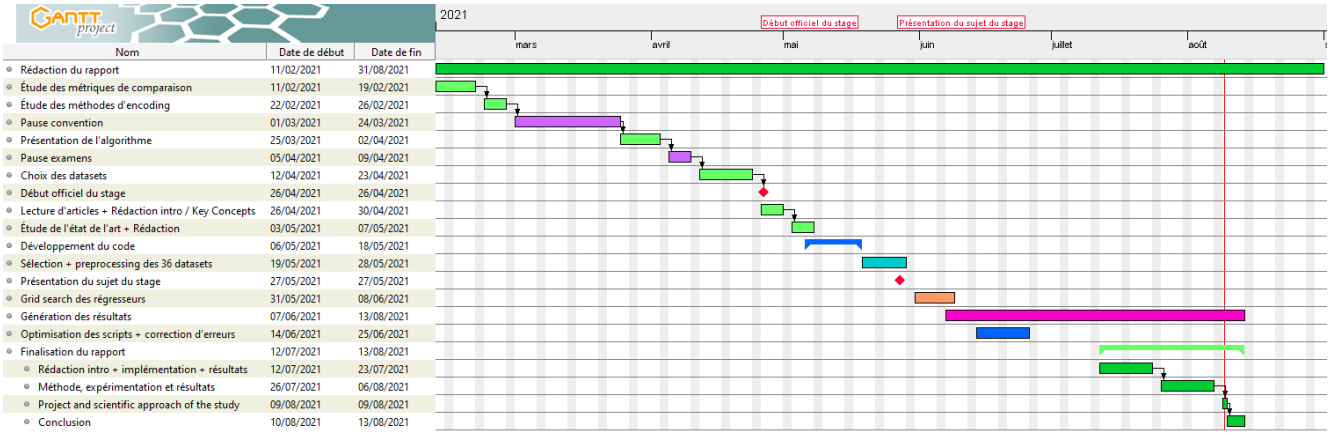


Figure 24: Gantt chart of the activities of the internship

The figure 24 is the Gantt chart of the activities that were carried out. Because I studied this last year of my engineering degree in the University of Sherbrooke (in Québec, Canada), the earlier I could start the internship

was the 26 April. For this reason, the duration of the internship was only 4 months. In order to completely carry out the task of this internship, it was decided by mutual agreement that I would start researching the subject and study the necessary notions beforehand a few hours each week. This is why the Gantt chart start in April and ends in August. The pre-study of the subject of the internship was mainly focused on the intricacies of the different metrics of evaluation of the regression models. During this time, the 35 datasets were also selected and I familiarized myself with the method proposed in this report. This upstream work had many advantages, as when the internship officially started I was able to start without a delay where I would research the regressors used or understand how the proposed method works. I also believe that this pre-study period allowed me to think in-depth about the method and bring many suggestions.

The first two weeks of the internship were dedicated to reading scientific articles about the regressors compared and writing sections related to the studied concepts (sections 1 and 2). Once this was done, I had all the knowledge necessary to implement the method, so the next two weeks were dedicated to developing the python scripts (section 4). The next step consisted in pre-processing the 35 datasets that were selected beforehand (section 3.2). This was quite a tedious task, as it required for each individual dataset to find which variable was the target one, remove the other target variables, encode the categorical variables, remove the missing values, etc... Once the datasets were cleaned, the next phase was the generation of the results (section 5.4). This step was the longest as it lasted for the next 10 weeks of the internship, which was partly because some results had to be re-generated after a change in the use of the grid-search was decided. Nonetheless, this didn't take all of my time, as the generation only had to be supervised, in case of an error or when the next scripts had to be launched. During this period, I had the opportunity to optimize the different scripts and to keep writing the report.

7.2 Code quality

The quality of the code that was written during this internship was a very important aspect of the project to me. Creating a project which I will be able to re-use (or at least partly) and that is easily understandable and maintainable is something that I have always been striving to do. This is why the project was split in 5 distinct scripts. Each one of them represent one of the main steps of the method: The first is the pre-processing of the dataset (with the k-fold split, the normalization, the definition of the thresholds, ...) The next script is the extraction of the new features using classifiers, then the training and prediction of the regressors, etc.

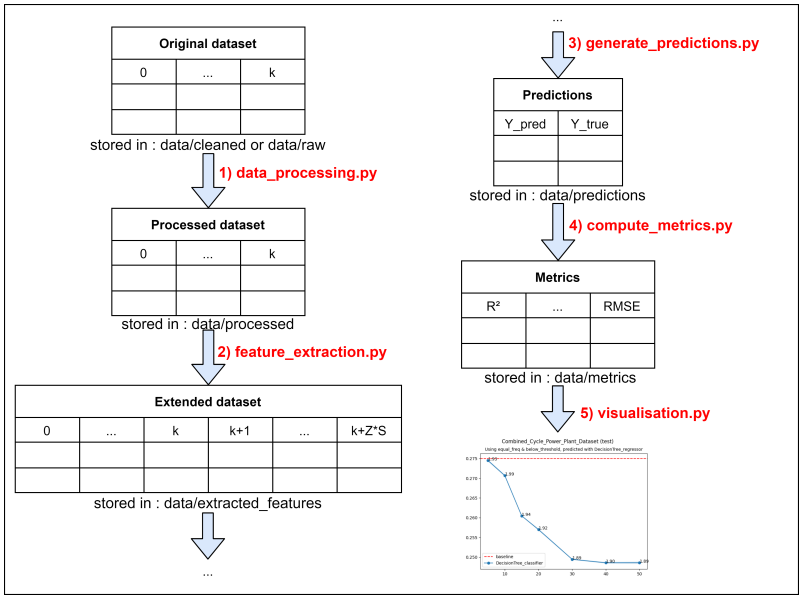


Figure 25: Diagram of the different scripts of the project

This approach had many advantages, as it allowed to stop the execution of the experiments at any point while losing almost no data and it was possible to easily monitor the contents of the files created by each script. The main drawback is the execution time, since the datasets had to be read and written in the storage memory between each script, which slowed down the overall process.

The documentation of the code was not neglected. Each class and method was described in the PEP 8 style (Python Enhancement Proposals 8) and line comments were included in the body of the methods when it was necessary for better clarity. Another element that greatly contributed to clarity was the used of different logging modes. Instead of using the basic *print(...)* function, the *logging.debug(...)*, *logging.info(...)* and *logging.warning(...)* were used early on in the project and give different level of information. The *debug* level gives the most details, printing the first lines of the dataset after each step, the reading and writing times of the datasets, etc. On the other hand, the *warning* level only prints a message when an error occurs. And the *info* level is midway between the other two levels, as it will inform of the progression of the execution, as well as the error messages.

7.3 Troubles encountered

Despite having a small number of errors in the code that was written for the experimentation, I still encountered a few situations that required reflection during this internship. A good example is the R^2 score that was suspiciously high for most of the datasets. After experimenting in a notebook with the least code possible to compare the performance of the regressors with the scores obtained after using all of the scripts of the project, it was clear that the code was not at fault. But after discussing with the members of the PROF team, it became clear that the R^2 is unsuited as a metric for the evaluation of the regressors in our context. This lead to the redaction of the section 3.3.5 which is a critique of this metric.

As explained in the section 4.3.2.B, implementing the AUROC metric to determine the performance of the classifiers had some challenges. When using the *roc_auc_score* function from the scikit-learn library, the scripts occasionally ran into some errors. To understand in-depth the mechanisms of the AUROC metric and comprehend the error, I read the article cited by scikit-learn that first introduced this metric ([27]). In this article, I noticed that the authors recommended computing a weighted mean with the class's frequency as weights. And since scikit-learn returned errors in the absence of some classes in the test samples, this meant that it was trying to computed values with a weight of 0. Once the origin of the errors was known, it was easy to fix them with a custom implementation of the AUROC metric.

Vincent Lemaire, my internship tutor, took some time every week when the code was in development to make sure that no incoherences or mistakes were present in the core code of the project. This proofreading of the code early on in the development was very important, as mistakes in this phase of the internship could influence the results and the decisions taken for the later stages of the project.

When observing the utilization of the CPU during the experimentation on my personal desktop or on the computation server, I noticed that the non-parallelized parts of the scripts took a large share of the total execution time. In order to discover which parts of the code were the slowest, I used python's *-m cProfile* option when running the scripts. This gives a detailed output of each instruction's execution time. Once sorted by cumulative time, it is possible to identify which instructions take the longest to compute. After some simplification and optimization, the total computation time was reduce by at least 30% in all of the scripts. As this was achieved without much effort, this means that it is something that could have been thought of during the development, and I will definitely pay more attention to the efficiency of the code that I write in the future.

7.4 Developed skills

During these 4 months of internship, I had the occasion to tackle many subjects related to machine learning. Given the very wide panel of materials that this topic is related to, I sometimes had to open my previous courses in statistics from Polytech and machine learning courses from the University of Sherbrooke. Given the variety of the regression and classification models that were used in the experimentation of this report, I definitely learned and re-learned many concepts.

Writing this report was a more difficult exercise than I had anticipated. My style of writing may have been pretty rough at the first, but thanks to my tutor's extensive proofreading, I believe I have come a long way in this area. My skills in reading and synthesizing scientific articles have also matured a lot, and I will continue reading new articles and books to further improve my reading speed and my understanding of abstract concepts.

I think my coding and technical skills were already sufficient before the start of this internship thanks to the very good Python courses that I took at the University of Sherbrooke. Even if the code quality aspect was not the priority for this project since it was a short term project on which I was working alone, I still managed to write clean and optimized code. I would have loved using a more recent framework, but scikit-learn was definitely enough for the needs of this internship.

During this internship, I also had the opportunity to make 2 presentations to the team of researchers I had joined. It was a great experience every time, as I learned how to create a clear and structured presentation. Preparing a slideshow explaining the subject of my internship from the start was a good way of getting a more general view of the project and getting fresh ideas.

Finally, this internship was for me a the first real research experience that I had, because it followed the same steps as a thesis (state of the art, implementation of the method, experimentation, results analysis, ...). It taught me that I really enjoyed doing research, and prompted me to apply for a thesis.

8 Conclusion

This section is the conclusion of the 4.5 month internship carried out at Orange Labs, Lannion, to obtain a computer engineering degree. In this section, I will give my personal feedback about the proposed method and my experience of the internship, as well as what could be done to further improve this work.

The main contribution offered by this report is the definition of a new method, *Feature engineering for regression using classification*. Papers that used classification models to solve a regression problem already existed ([1, 2, 17, 18, 25, 30]), but our method diverges from them since it still uses a regression model to solve the initial problem. The experimentation was also quite extensive, with 35 unique datasets used to compare the impact on performance of our method on 5 regression models.

The sections 2 and 3 cover the basic knowledge needed to understand the main concepts used in this report. I enjoyed writing these sections, as it required me to research in-depth various important subjects for machine learning in general (pre-processing, feature engineering, evaluation, etc.). The section 3.3.5, which is a critique of the R-Squared metric, was particularly interesting to me because I was surprised that a very widely used metric is actually unsuitable for most evaluations.

Regarding the performance of the method proposed in this report, I think that its current performance gain is a bit lacking compared to the computation time it requires for the XGBoost and Random Forest regressors. If we look at its impact on Linear Regression, it is true that it makes the worst performing method into a method that is not statistically different from a XGBoost or a Random Forest. But it uses many full size Random Forest classifiers to show results only a little bit better than those of a Random Forest regressor. The explainability of the models is still present, but it becomes harder, as a very high importance is given to the conditional probabilities extracted by our method. And they cannot be used to immediately extract useful information about the relation of the dependant variables to the target variable. To understand the final decision of the regressor, it requires analysing the classifiers.

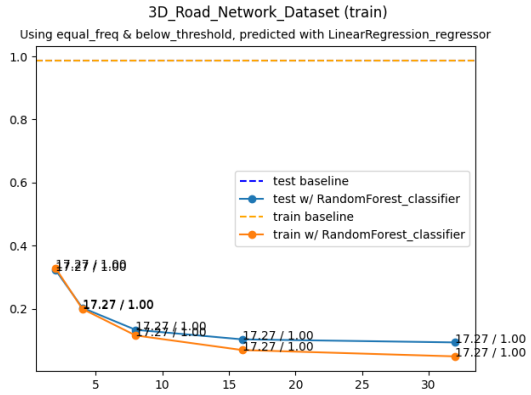
While still being small, our method has a positive impact on the performance on the best performing regressors of the state of the art, XGBoost and Random Forest. On the other hand, using a Random Forest classifier to augment the performance of the Khiops regressor yields an impressive 31.85% reduction in test RMSE! And even if the explainability is somewhat lost, the advantage of the quantiles of Khiops is still retained. This is mainly because this method of using classifiers to extract features for regression models was initially tailored for the Khiops regressor. However, it was decided to create a project and a report that was independant from Khiops. This allowed the internship to be less focused on Khiops and to create a report that could be read by anybody. So in the regard of the improvement of the performance of Khiops, it was a success !

Appendices

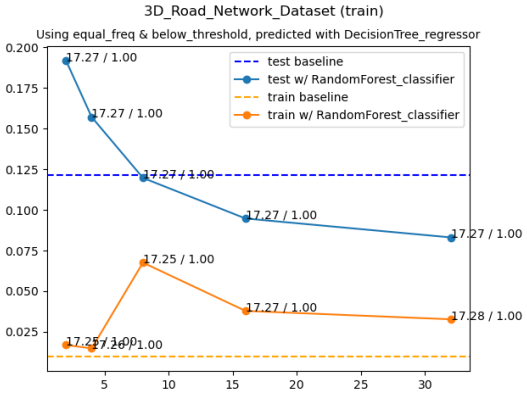
A Results figures

A.1 3D Road network Dataset

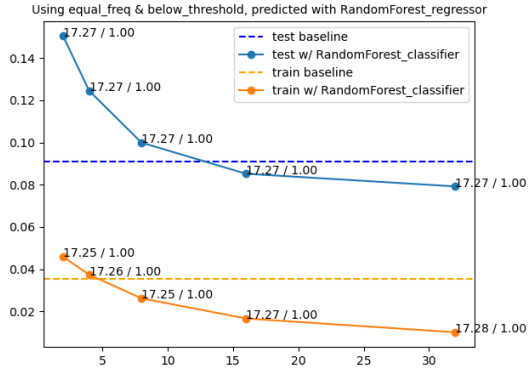
Linear Regression



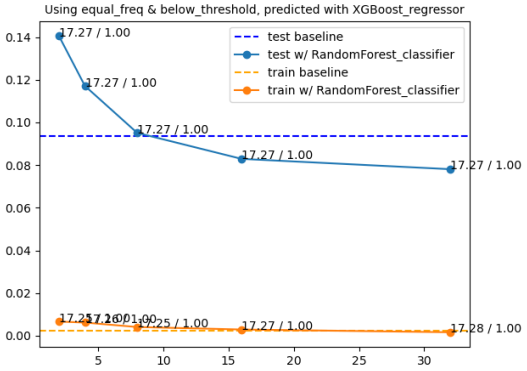
Decision Tree Regressor



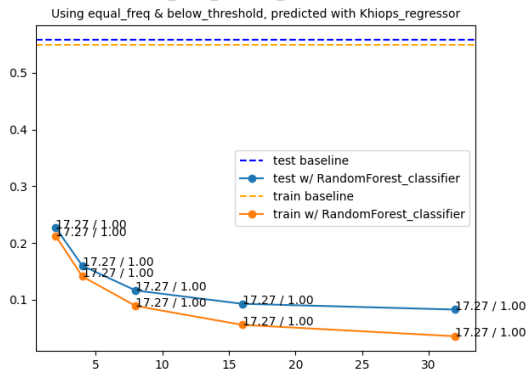
Random Forest Regressor



XGBoost Regressor

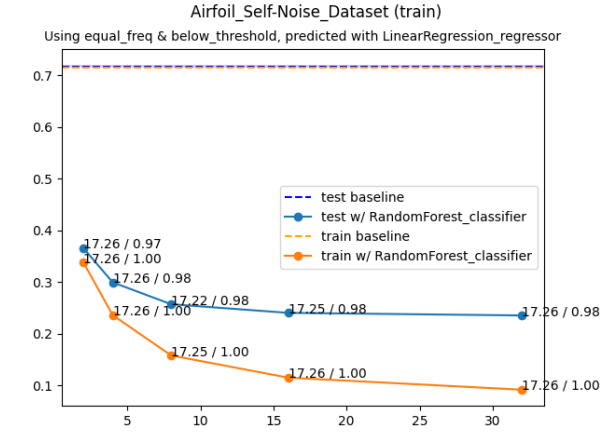


Khiops Regressor

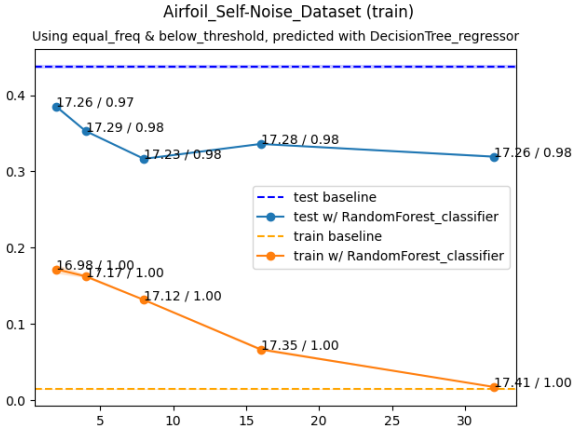


A.2 Airfoil self-noise Dataset

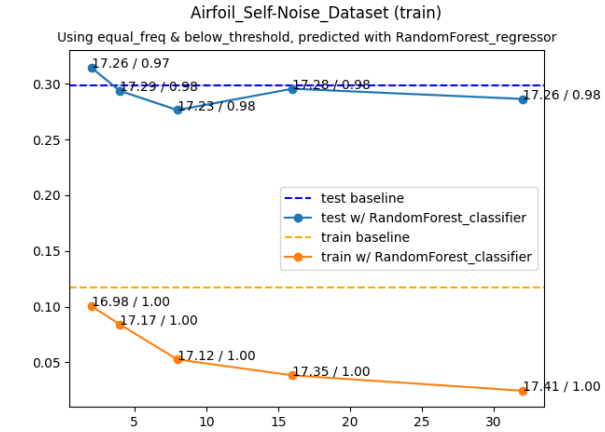
Linear Regression



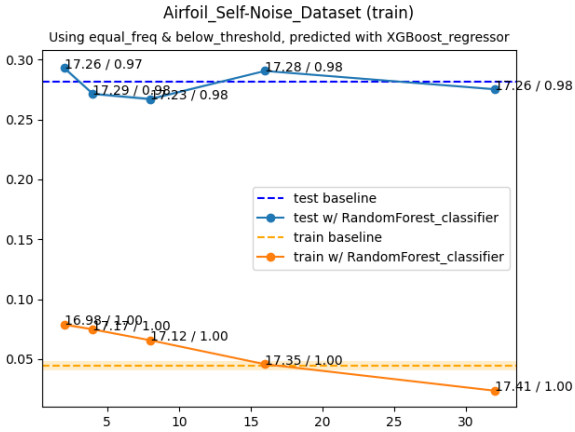
Decision Tree Regressor



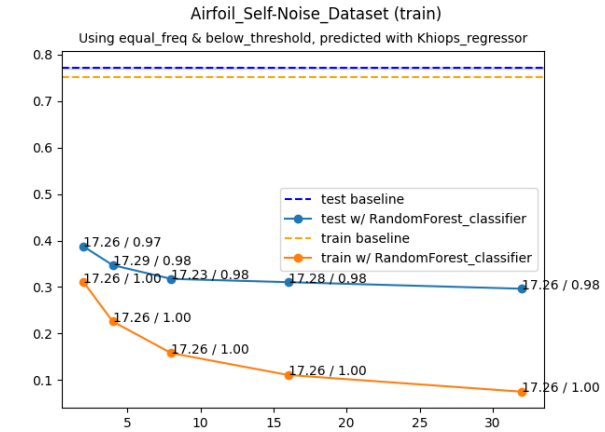
Random Forest Regressor



XGBoost Regressor

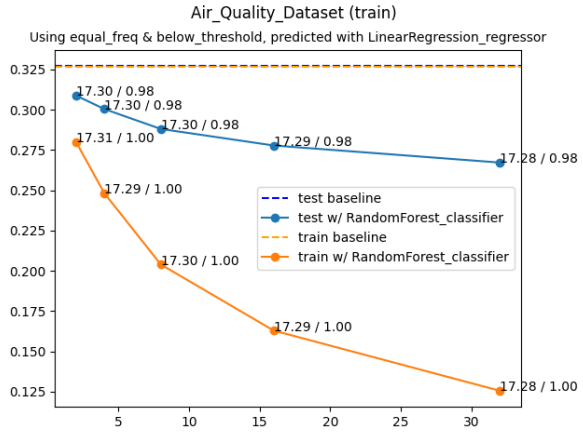


Khiops Regressor

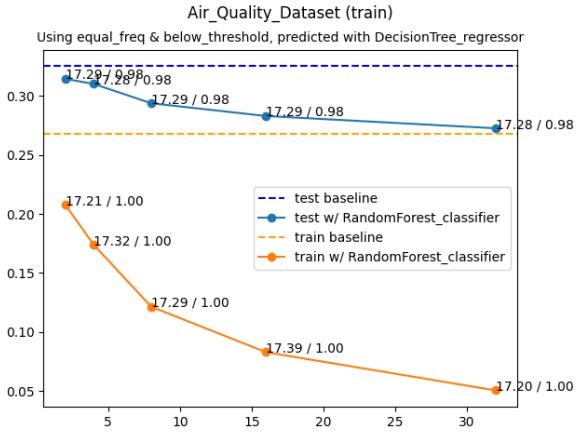


A.3 Air quality Dataset

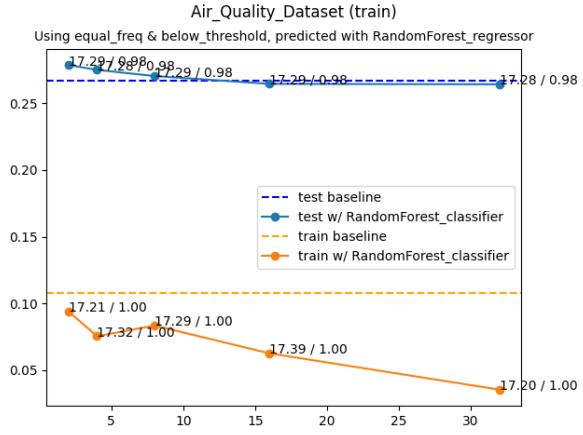
Linear Regression



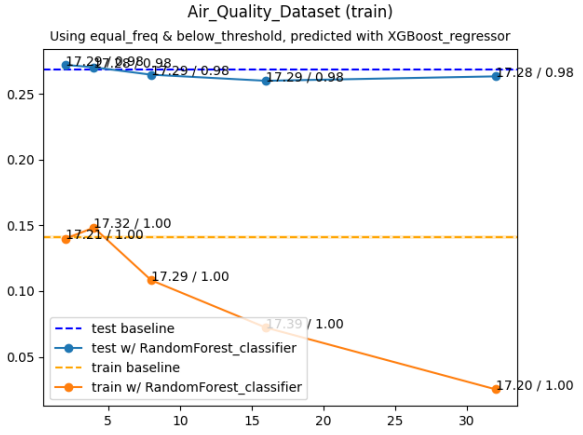
Decision Tree Regressor



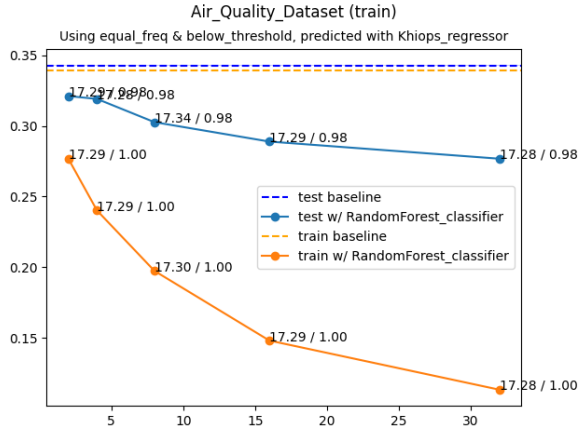
Random Forest Regressor



XGBoost Regressor

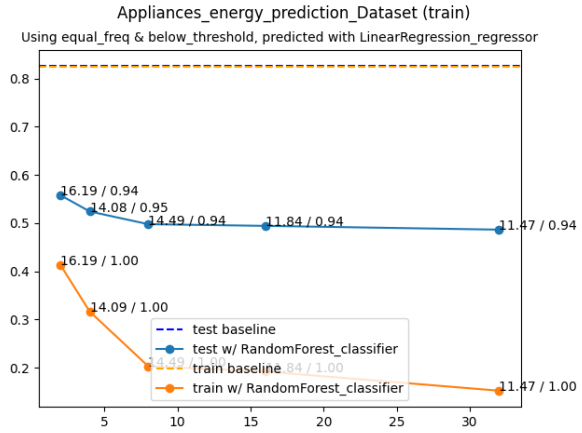


Khiops Regressor

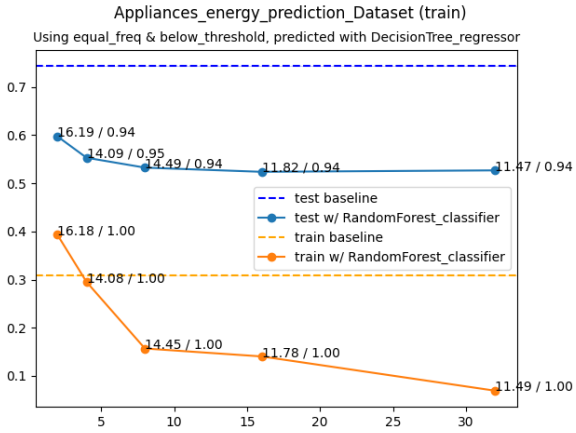


A.4 Appliances energy prediction Dataset

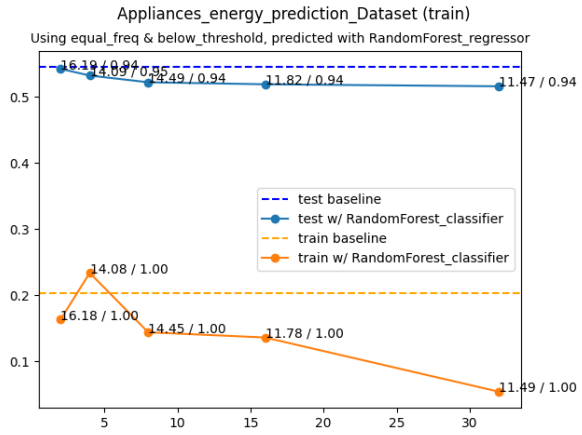
Linear Regression



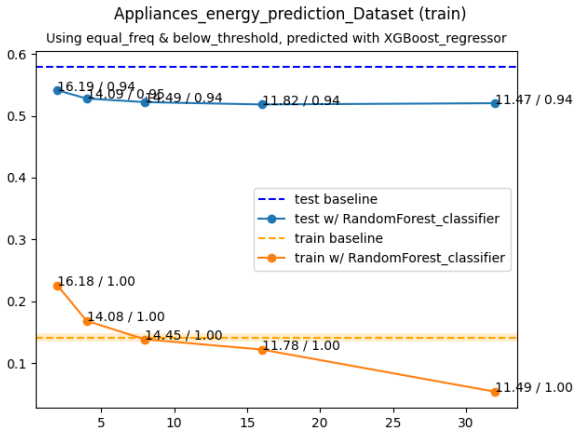
Decision Tree Regressor



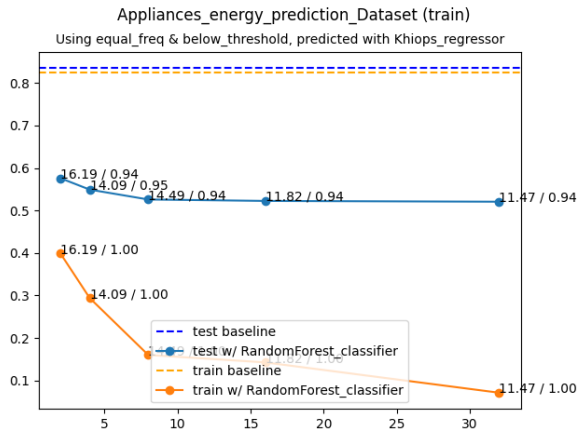
Random Forest Regressor



XGBoost Regressor

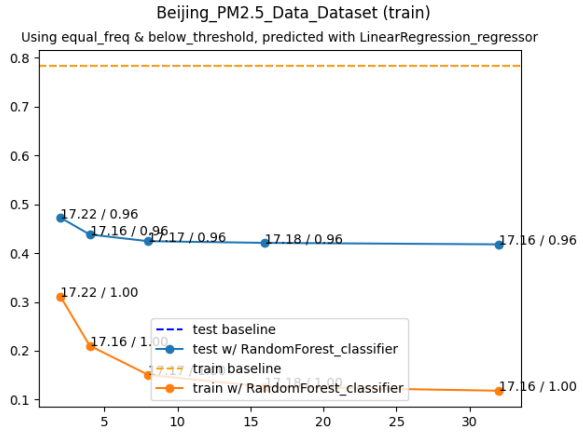


Khiops Regressor

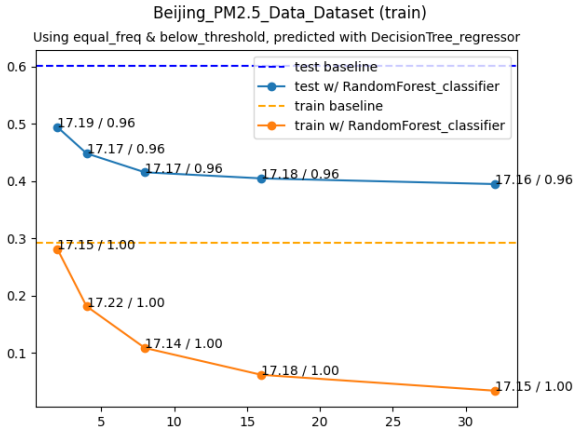


A.5 Beijing PM2.5 Dataset

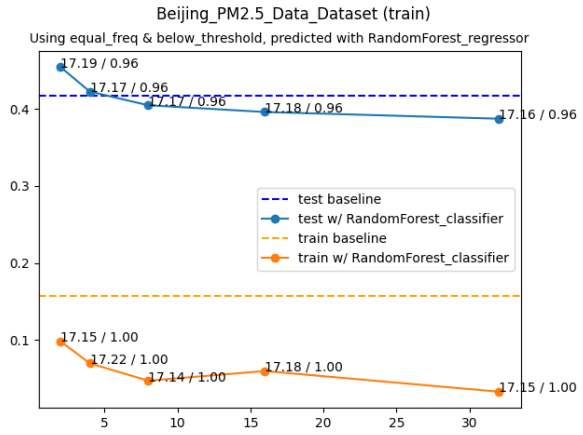
Linear Regression



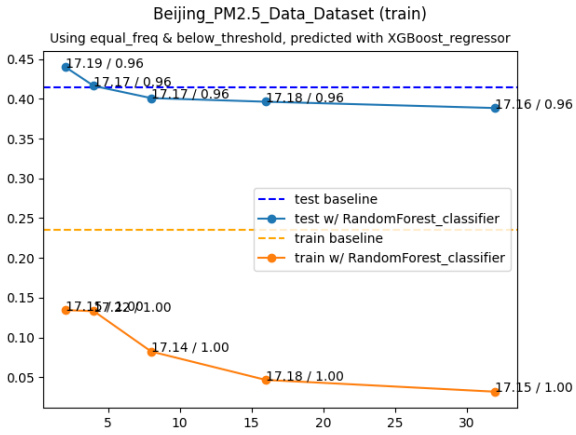
Decision Tree Regressor



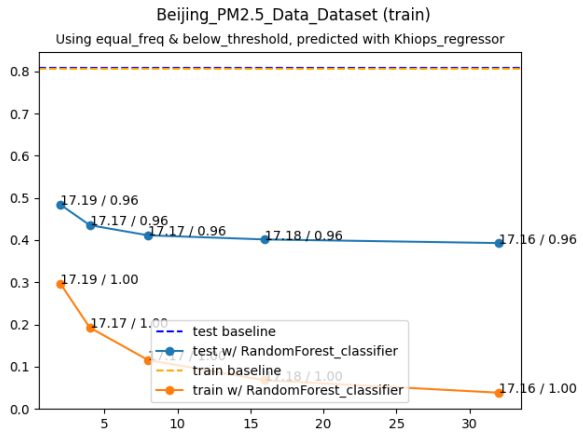
Random Forest Regressor



XGBoost Regressor

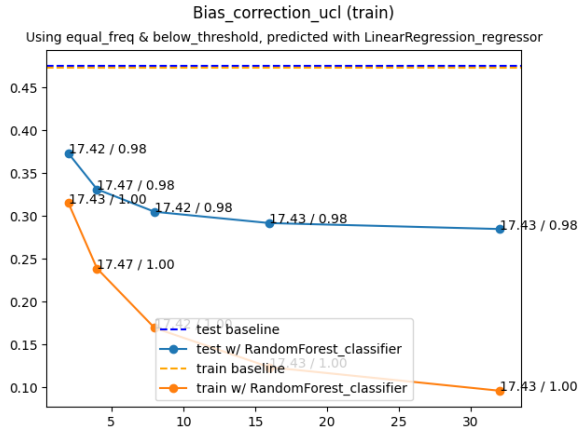


Khiops Regressor

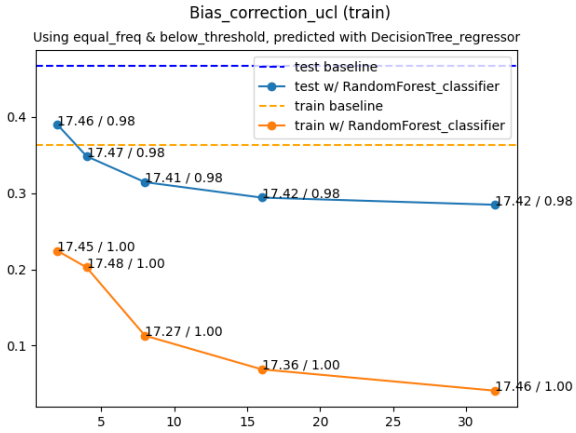


A.6 Bias correction of numerical prediction model temperature forecast Dataset

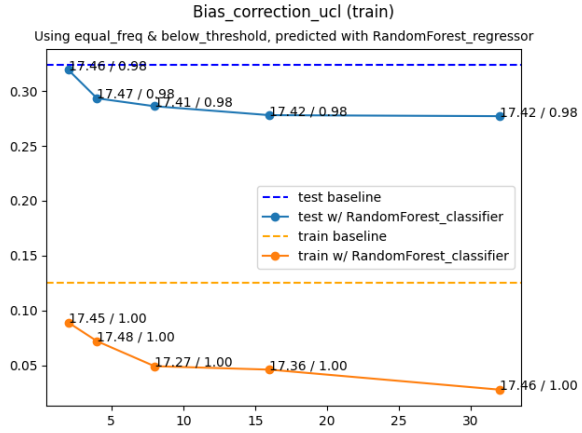
Linear Regression



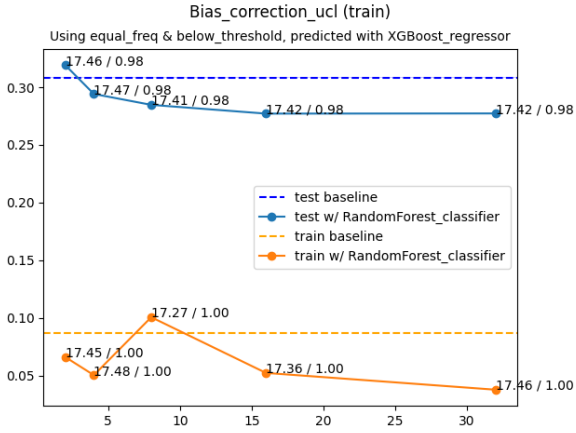
Decision Tree Regressor



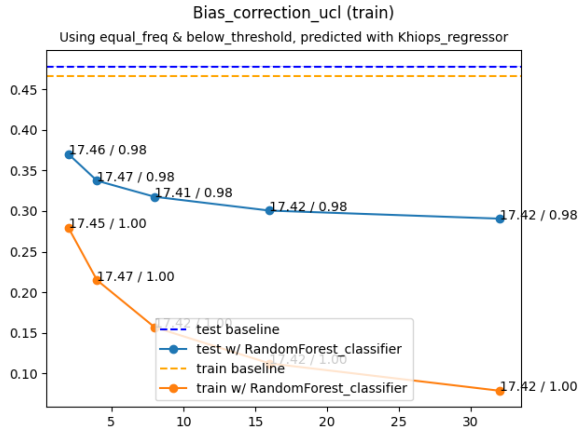
Random Forest Regressor



XGBoost Regressor

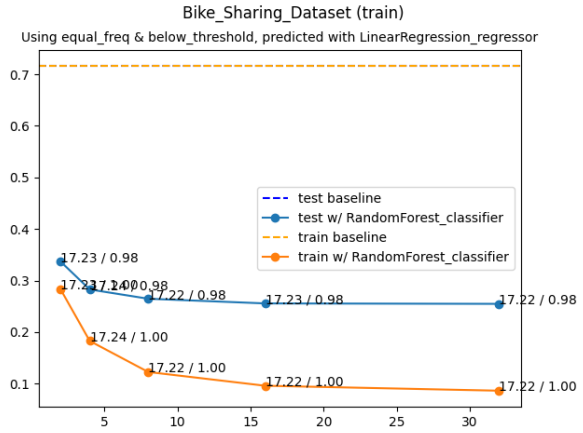


Khiops Regressor

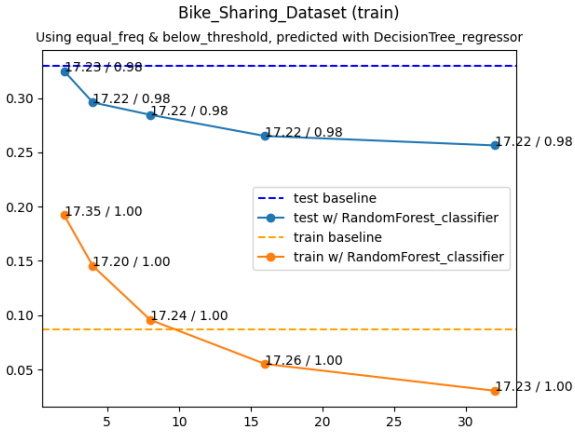


A.7 Bike sharing Dataset

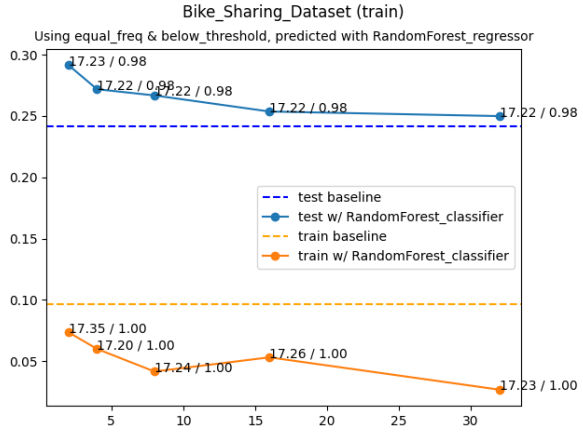
Linear Regression



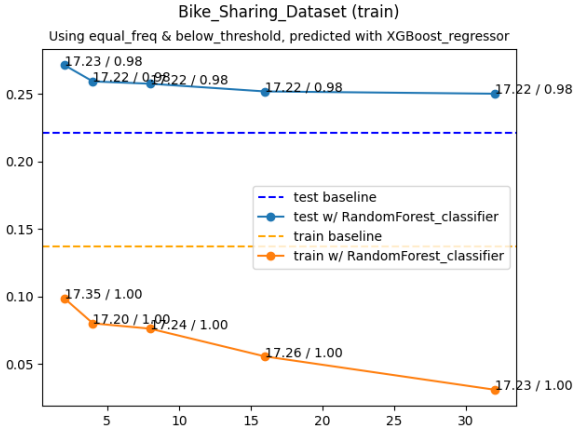
Decision Tree Regressor



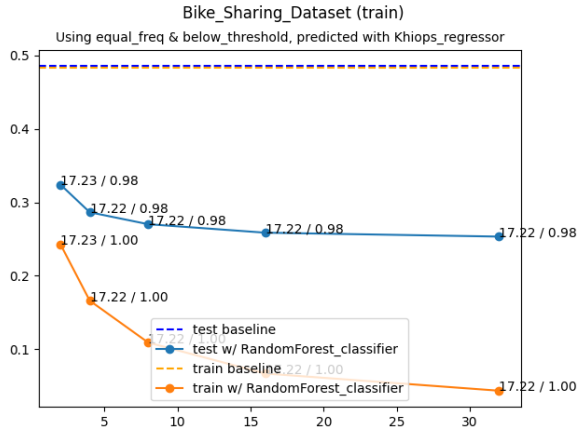
Random Forest Regressor



XGBoost Regressor

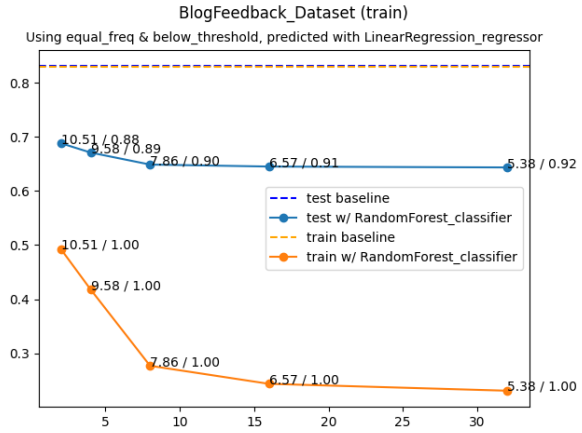


Khiops Regressor

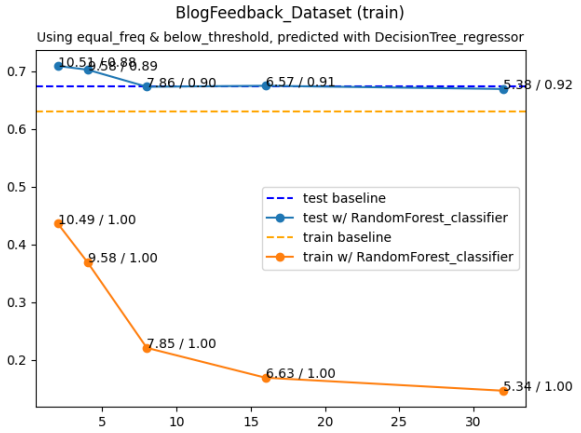


A.8 Blog feedback Dataset

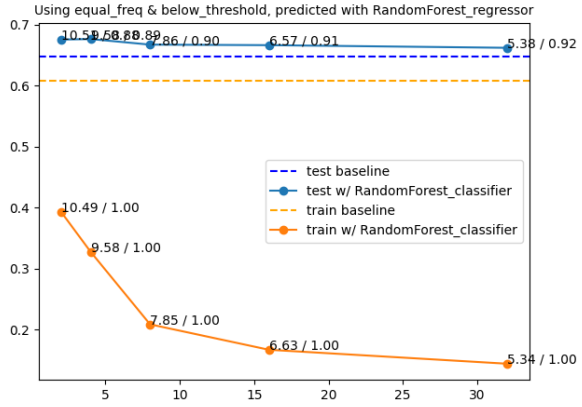
Linear Regression



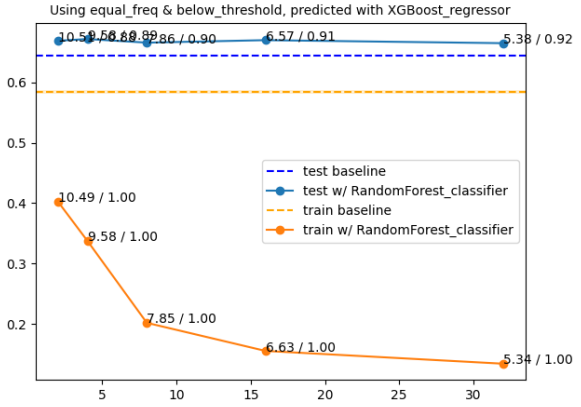
Decision Tree Regressor



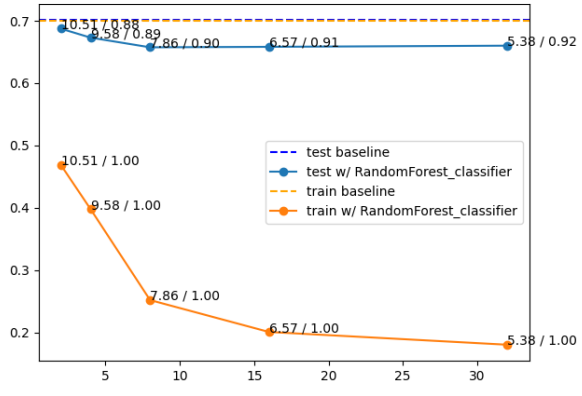
Random Forest Regressor



XGBoost Regressor

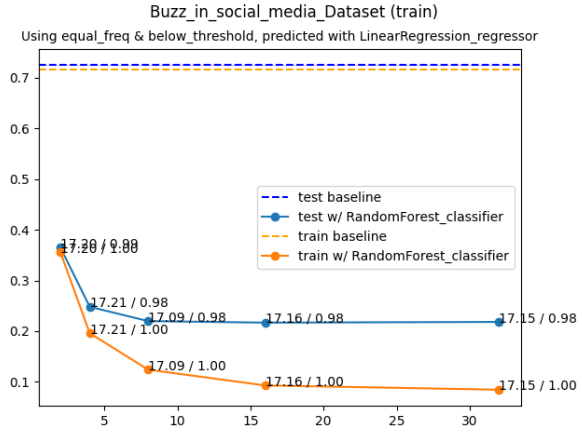


Khiops Regressor

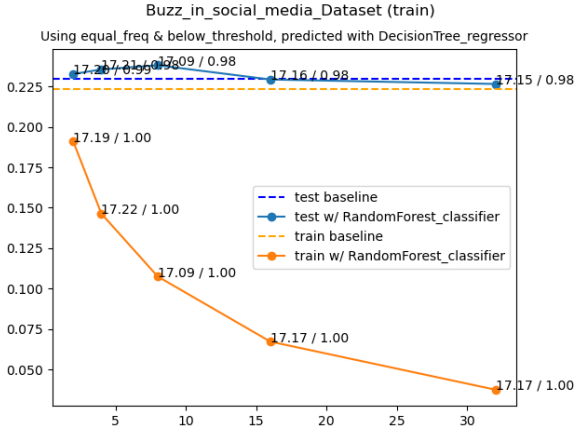


A.9 Buzz in social media Dataset

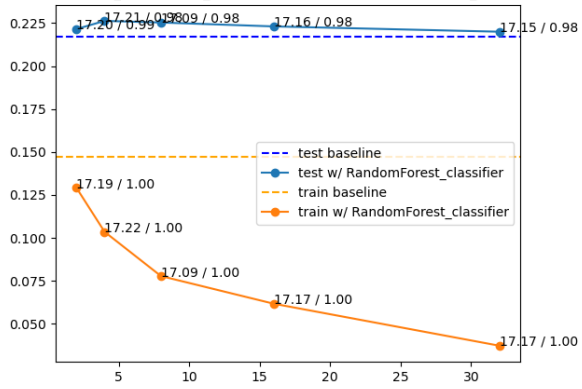
Linear Regression



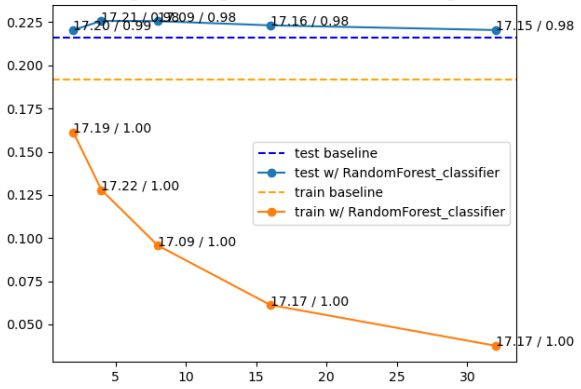
Decision Tree Regressor



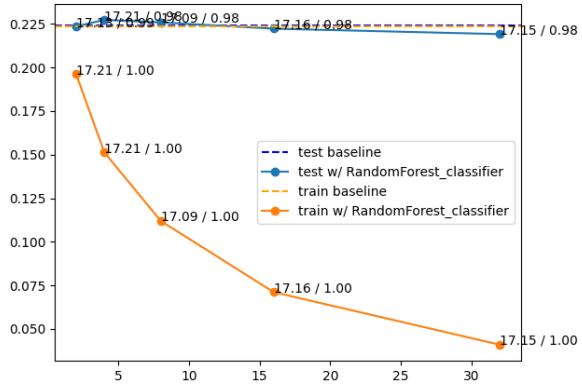
Random Forest Regressor



XGBoost Regressor

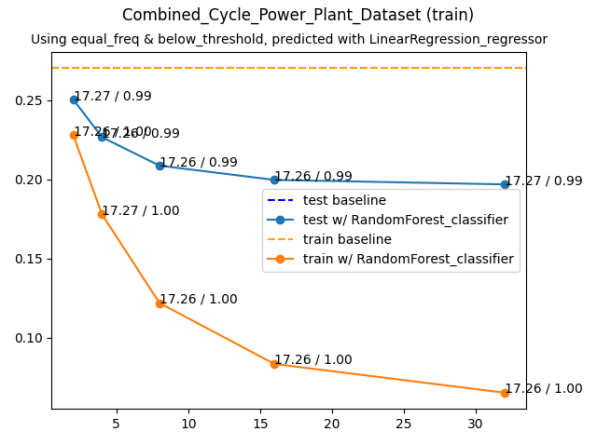


Khiops Regressor

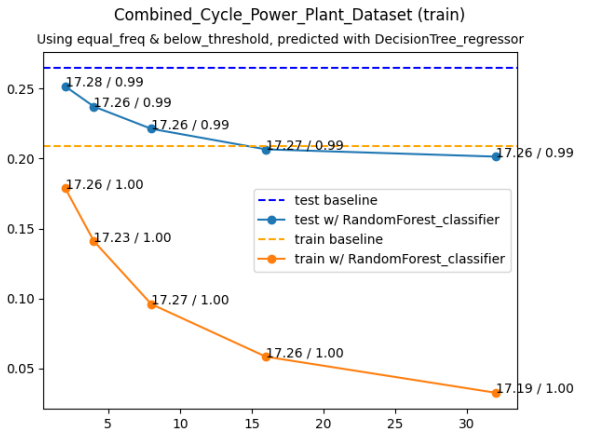


A.10 Combined cycle power plant Dataset

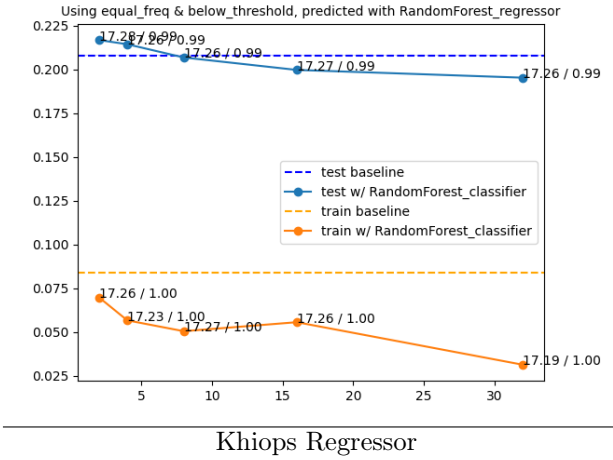
Linear Regression



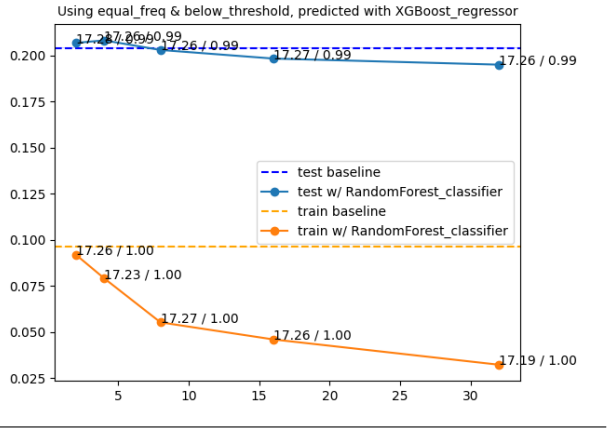
Decision Tree Regressor



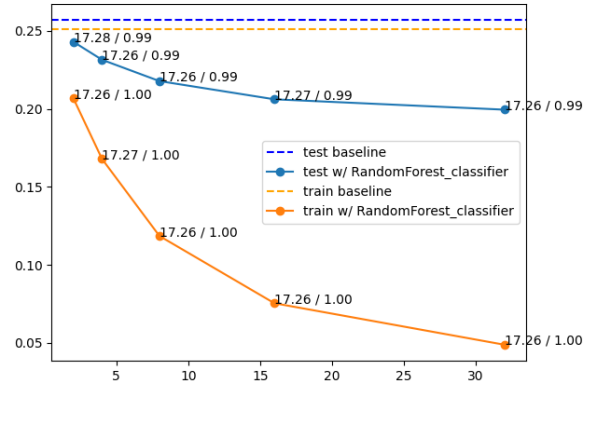
Random Forest Regressor



XGBoost Regressor

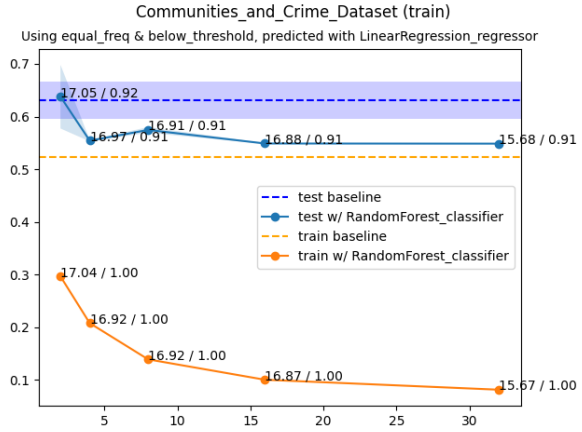


Khiops Regressor

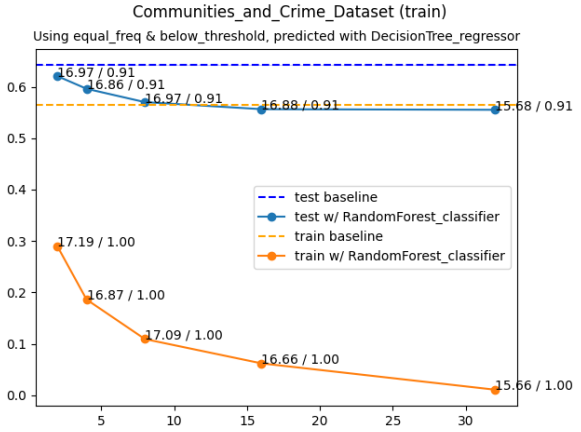


A.11 Communities & crime Dataset

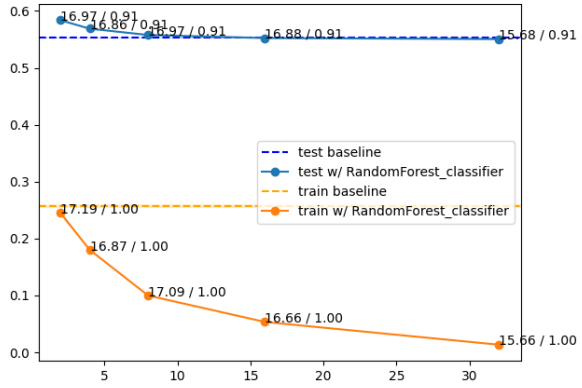
Linear Regression



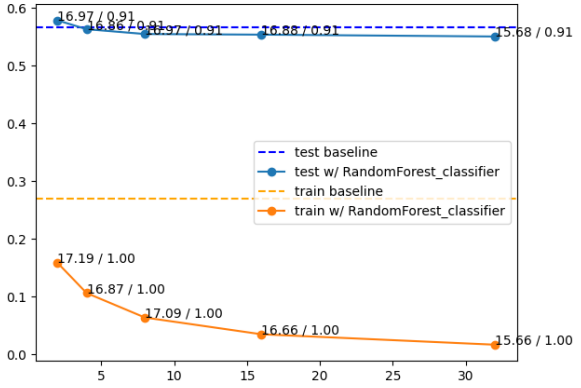
Decision Tree Regressor



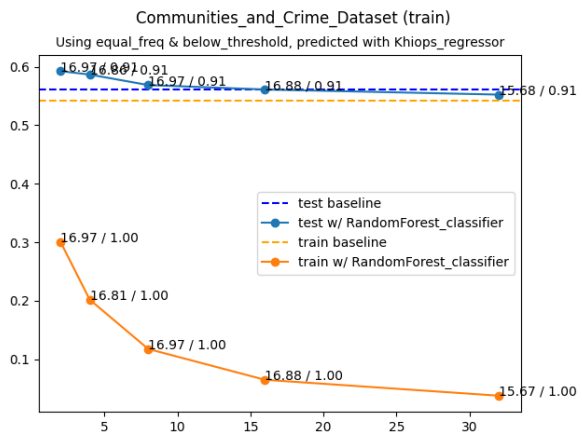
Random Forest Regressor



XGBoost Regressor

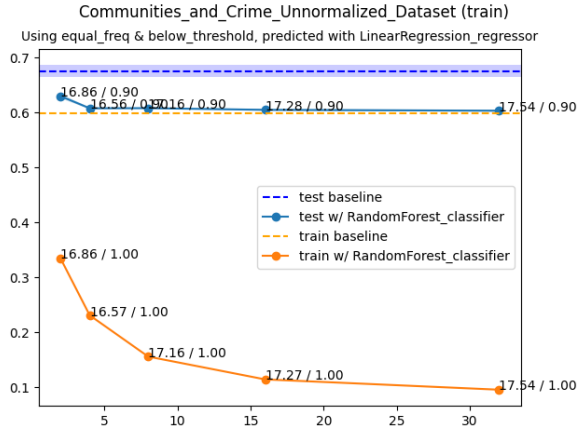


Khiops Regressor

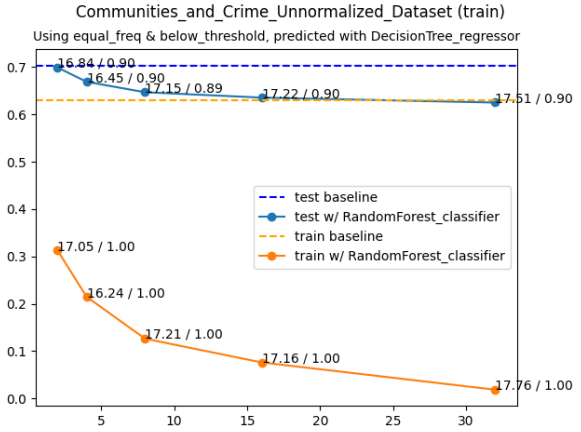


A.12 Communities & crime unnormalized Dataset

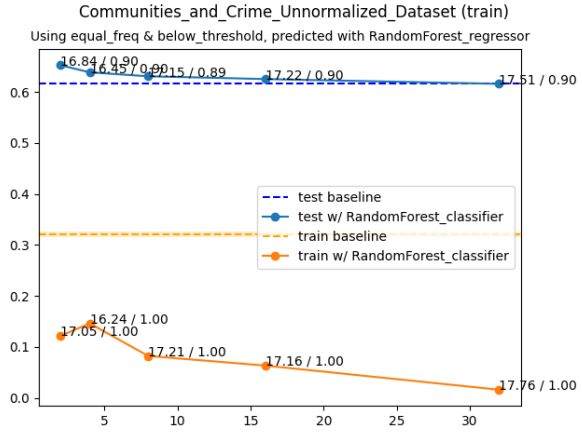
Linear Regression



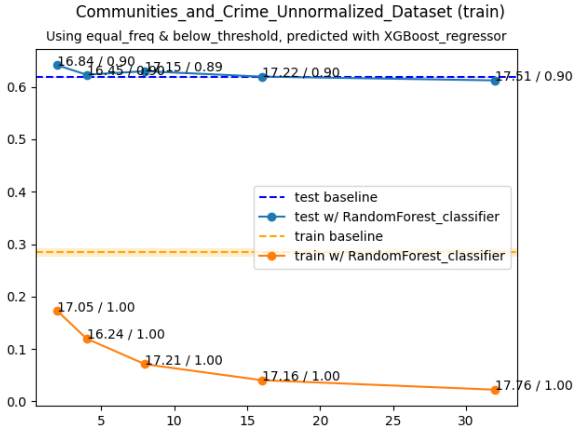
Decision Tree Regressor



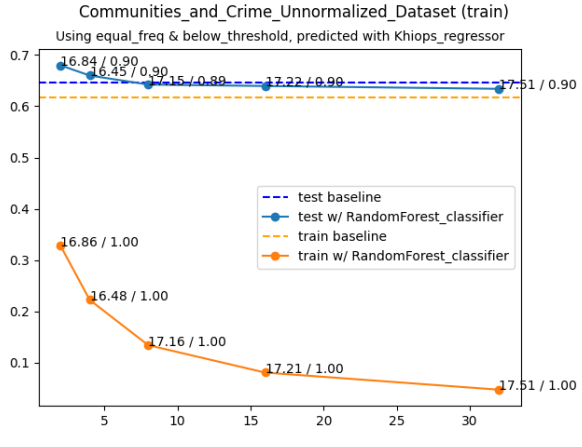
Random Forest Regressor



XGBoost Regressor

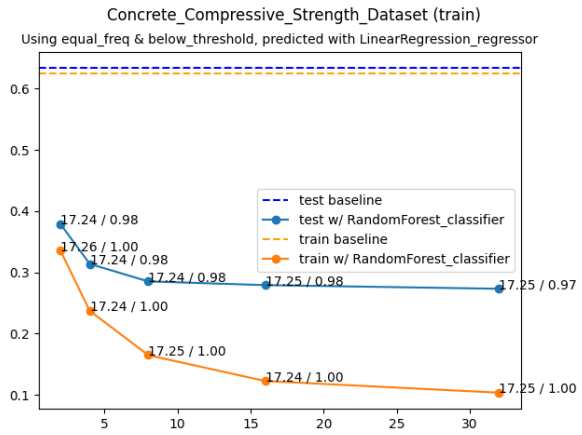


Khiops Regressor

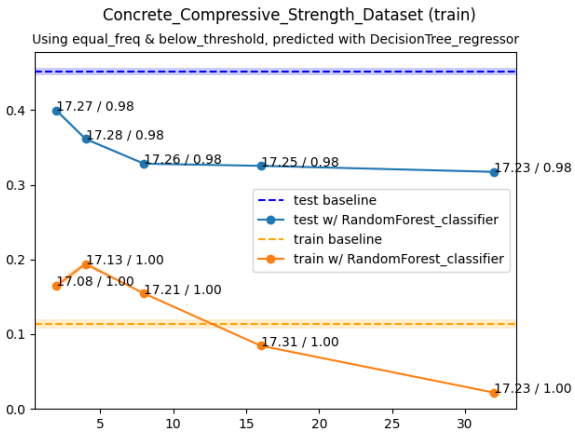


A.13 Concrete compressive strength Dataset

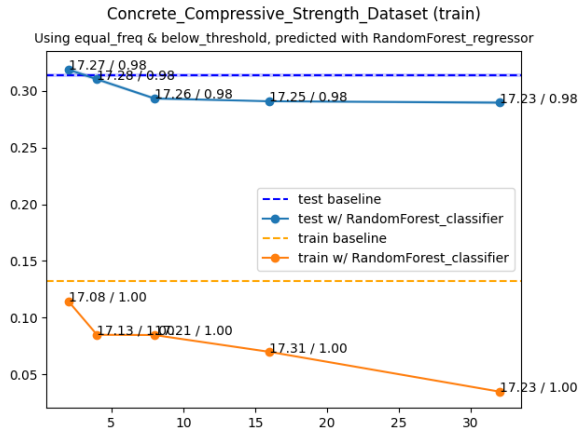
Linear Regression



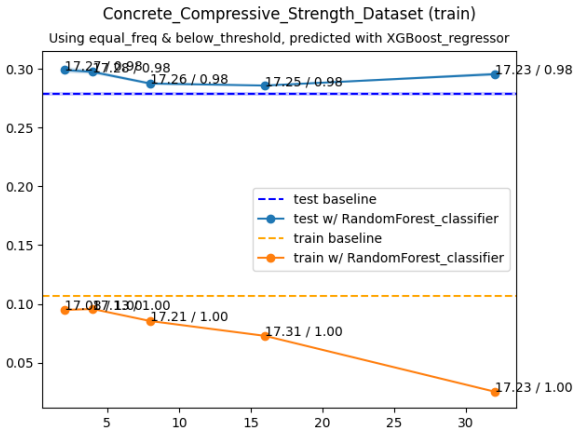
Decision Tree Regressor



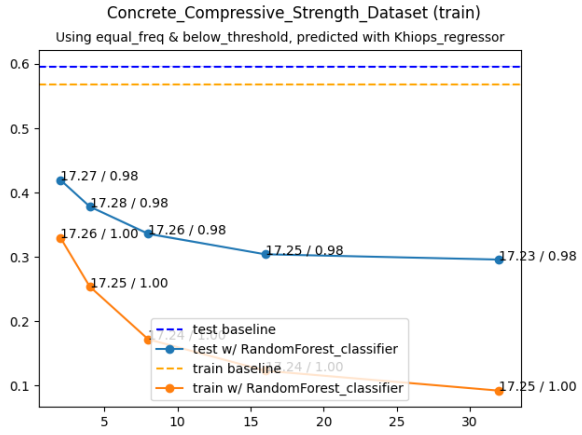
Random Forest Regressor



XGBoost Regressor

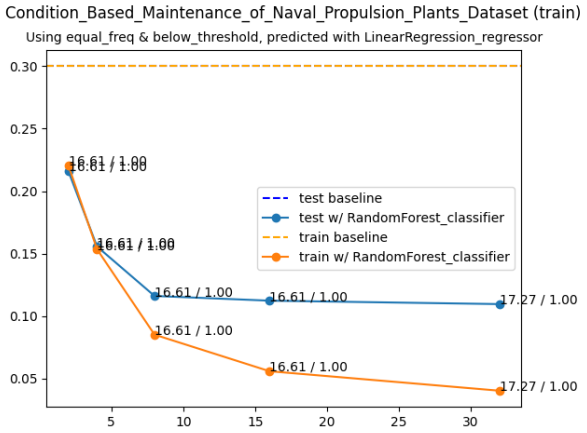


Khiops Regressor

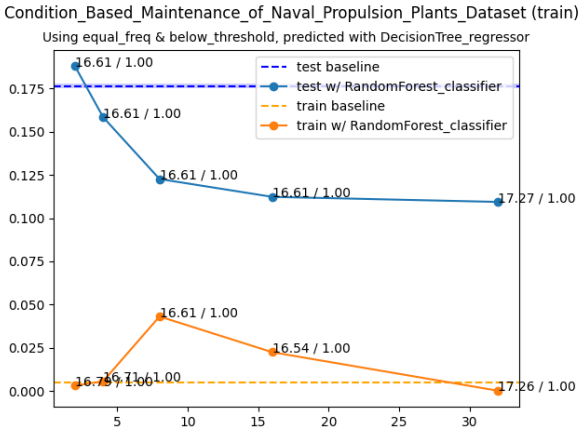


A.14 Condition based maintenance of naval propulsion plants Dataset

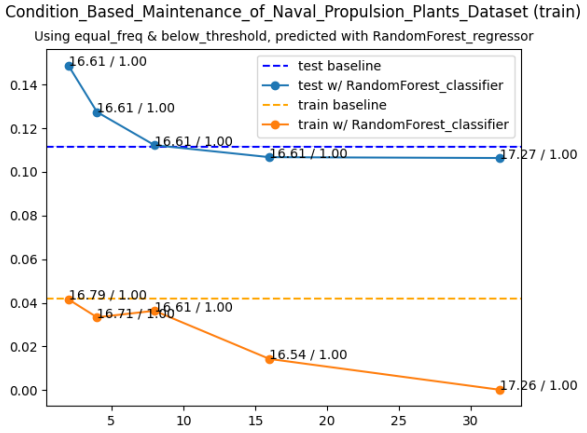
Linear Regression



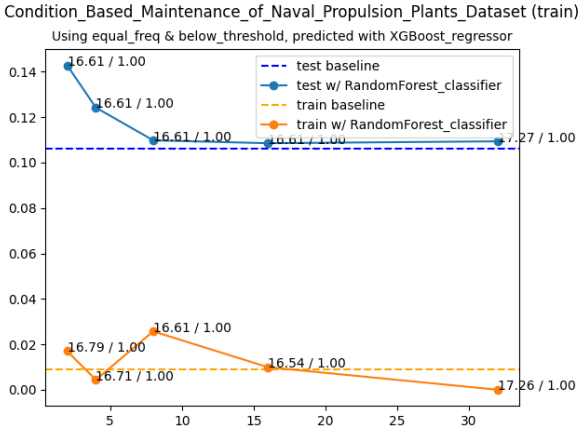
Decision Tree Regressor



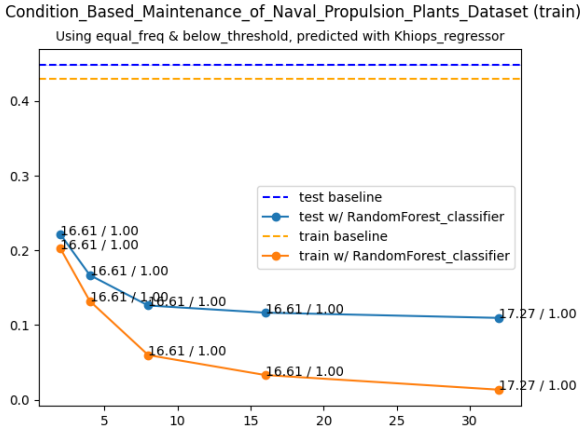
Random Forest Regressor



XGBoost Regressor

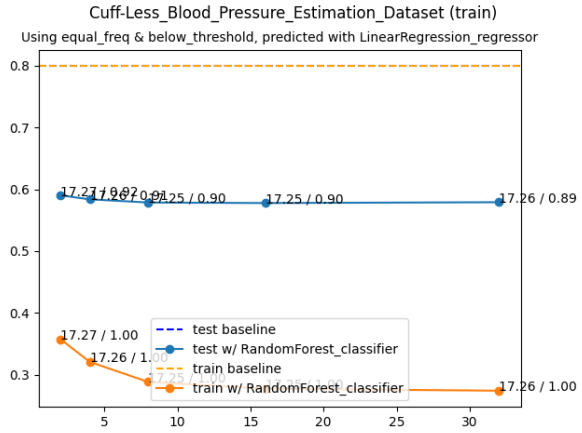


Khiops Regressor

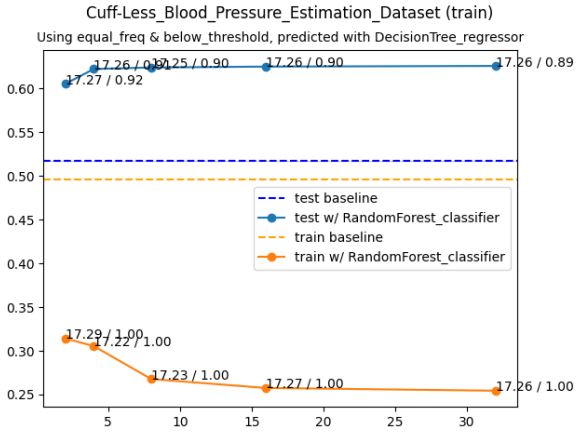


A.15 Cuff-less blood pressure estimation Dataset

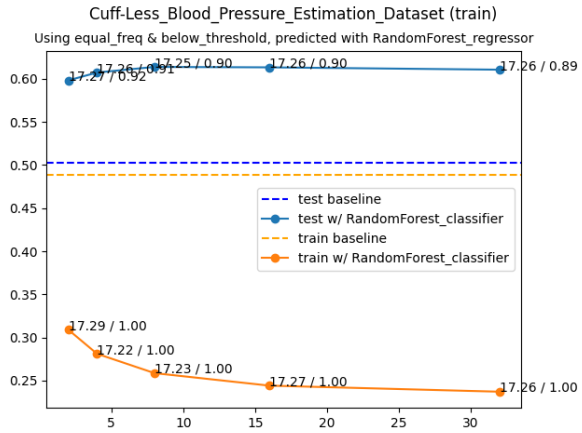
Linear Regression



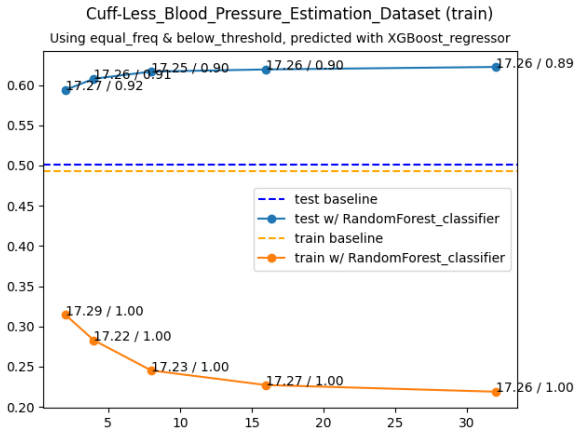
Decision Tree Regressor



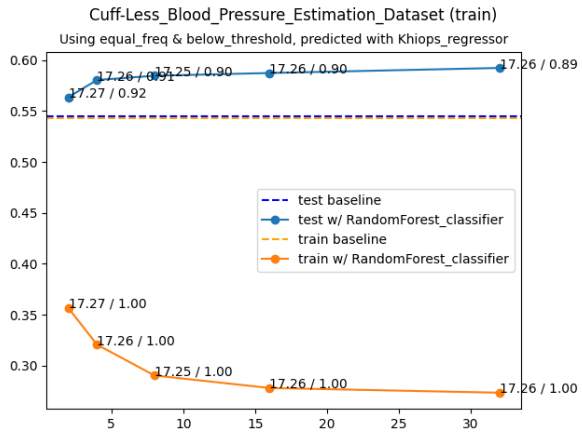
Random Forest Regressor



XGBoost Regressor

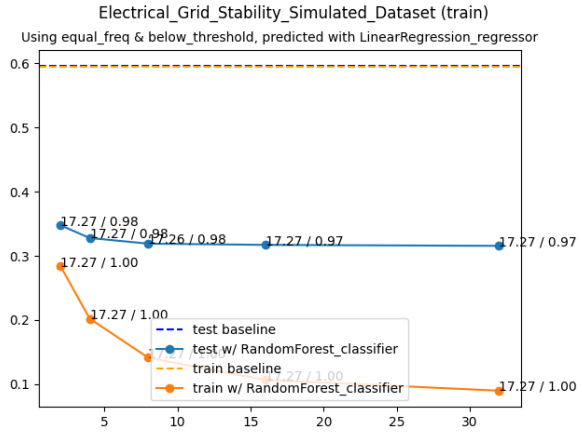


Khiops Regressor

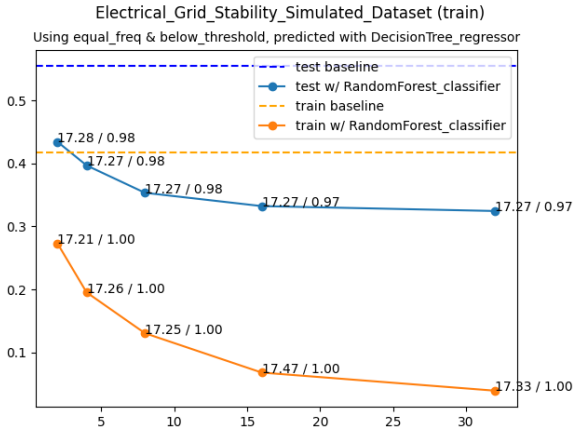


A.16 Electrical Grid Stability Simulated Dataset

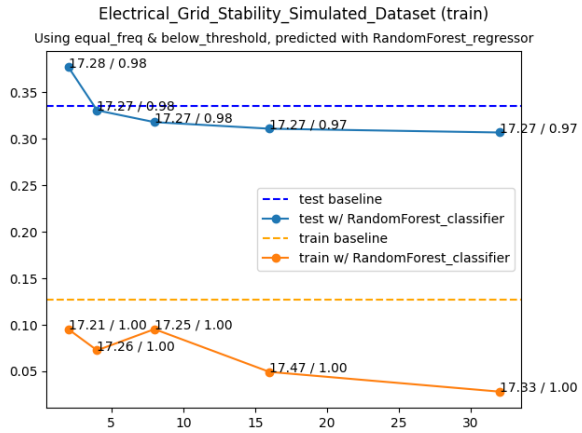
Linear Regression



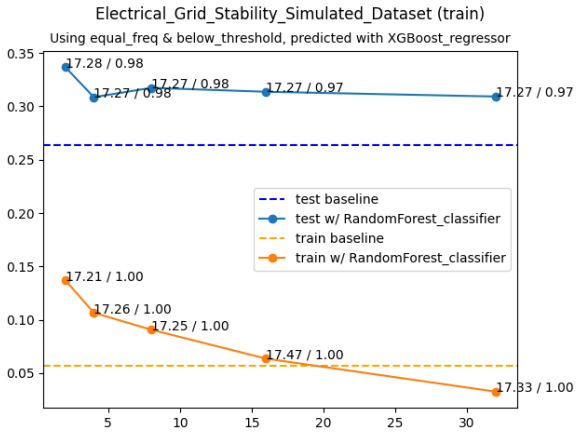
Decision Tree Regressor



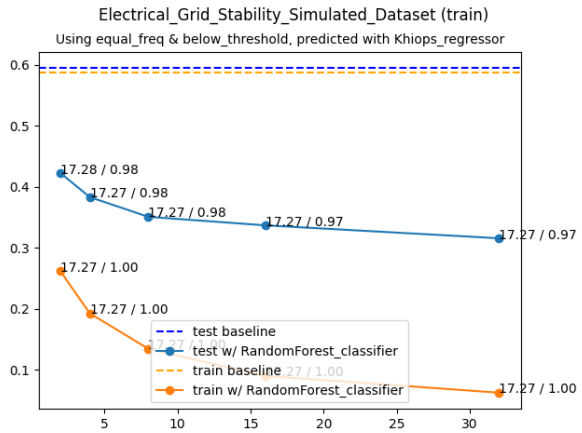
Random Forest Regressor



XGBoost Regressor

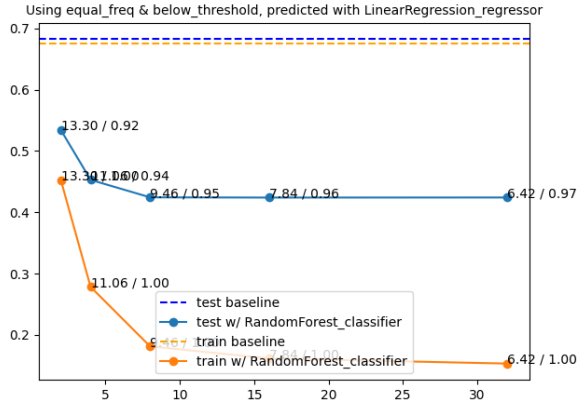


Khiops Regressor

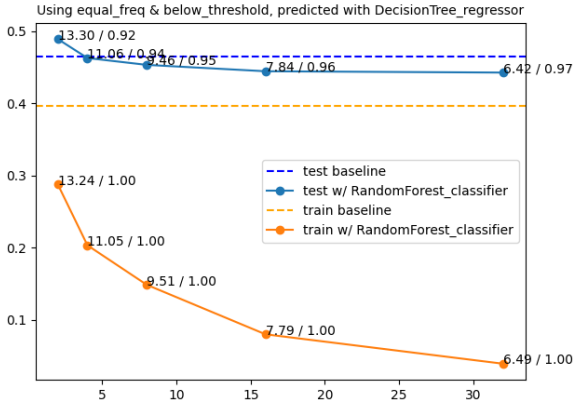


A.17 Facebook comment volume Dataset

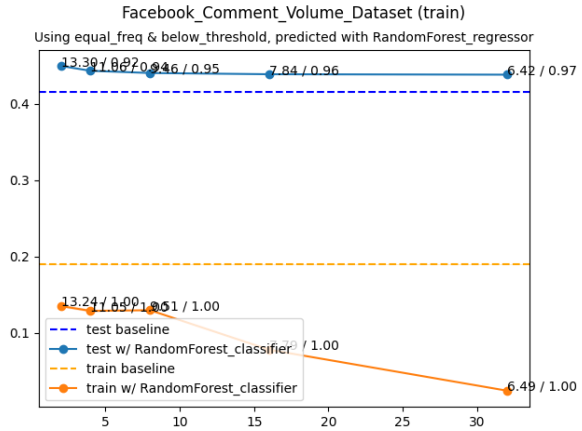
Linear Regression



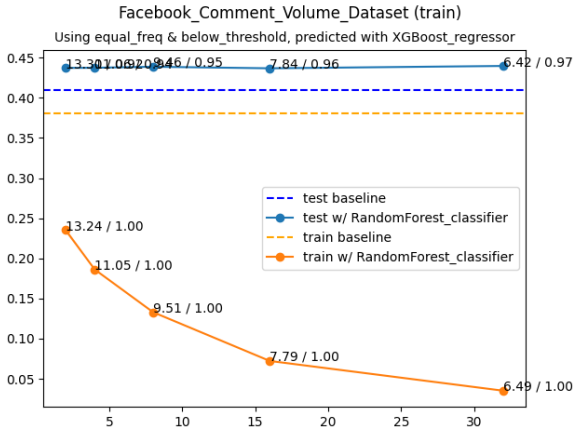
Decision Tree Regressor



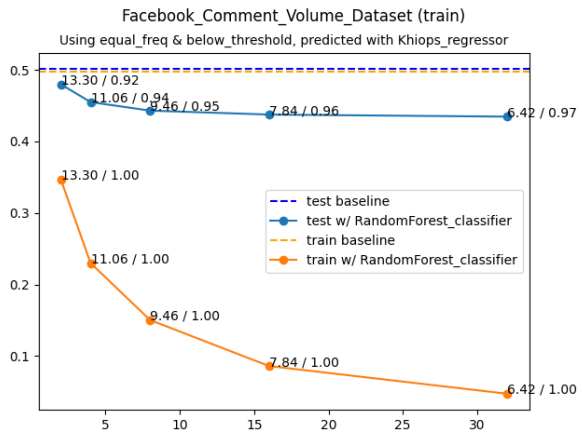
Random Forest Regressor



XGBoost Regressor

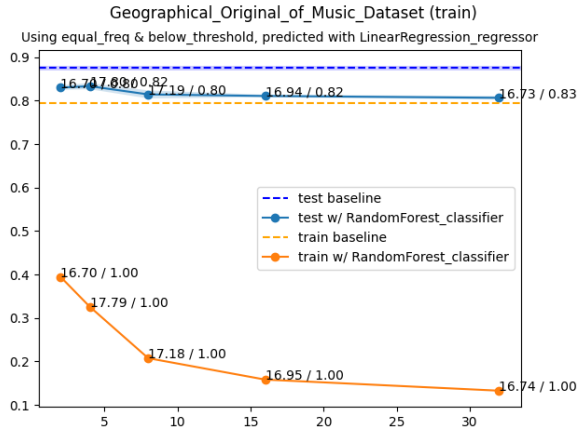


Khiops Regressor

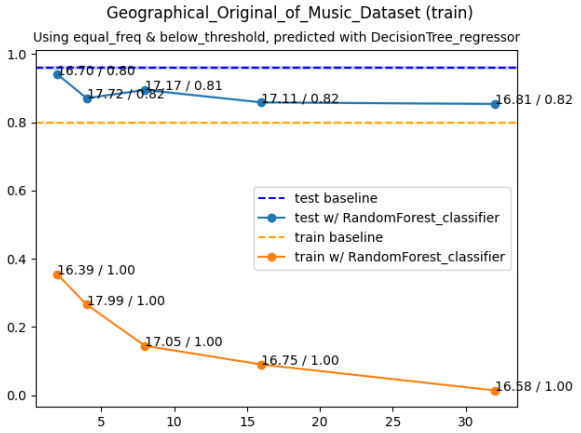


A.18 Geographical original of music Dataset

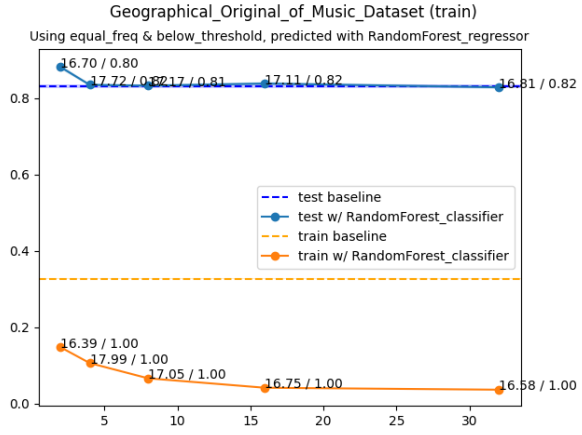
Linear Regression



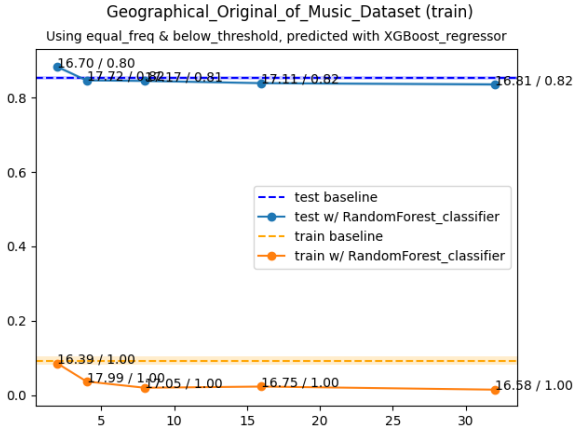
Decision Tree Regressor



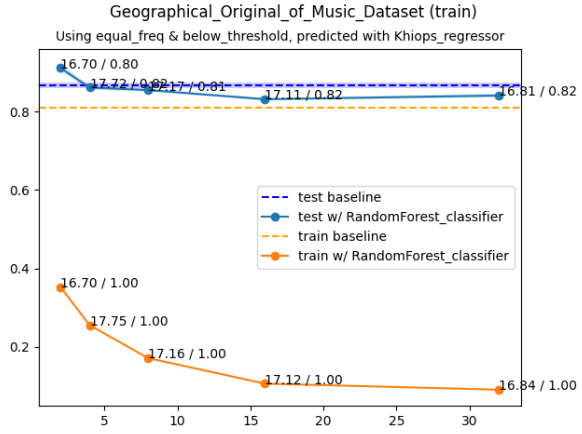
Random Forest Regressor



XGBoost Regressor

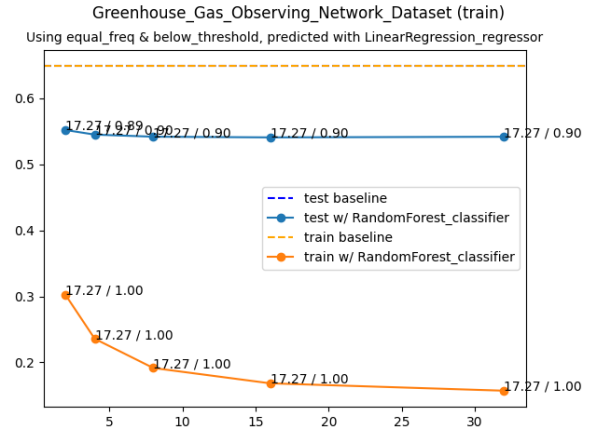


Khiops Regressor

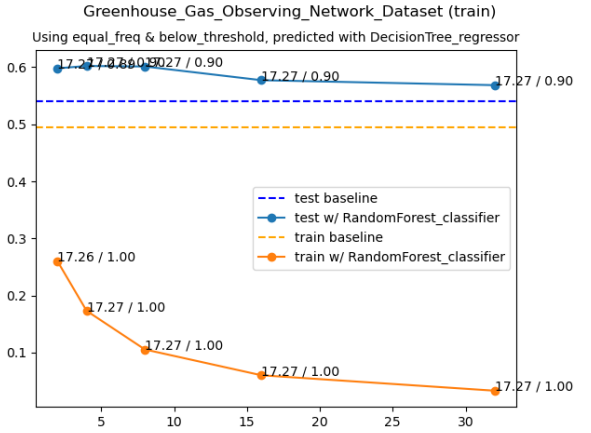


A.19 Greenhouse gas observing network Dataset

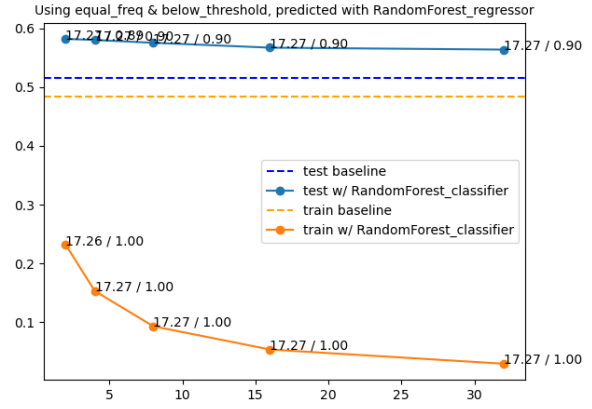
Linear Regression



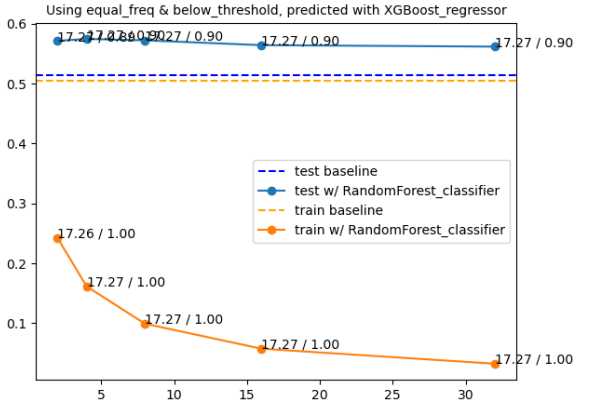
Decision Tree Regressor



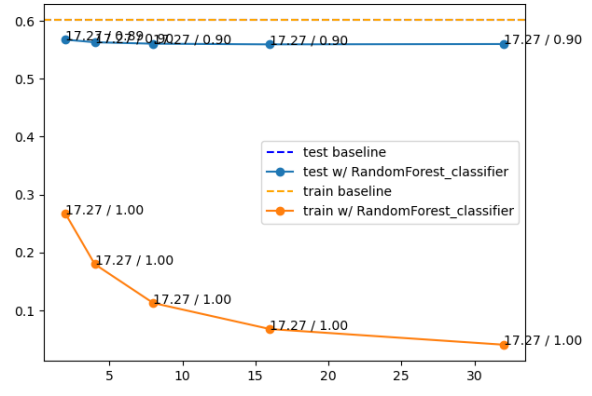
Random Forest Regressor



XGBoost Regressor

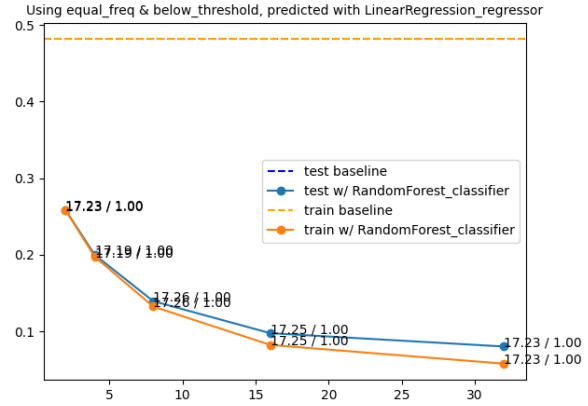


Khiops Regressor

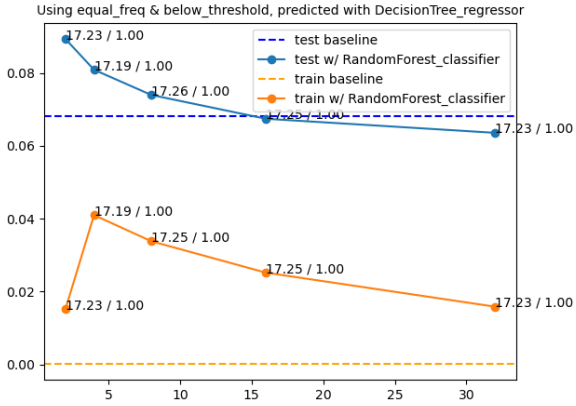


A.20 Individual household electric power consumption Dataset

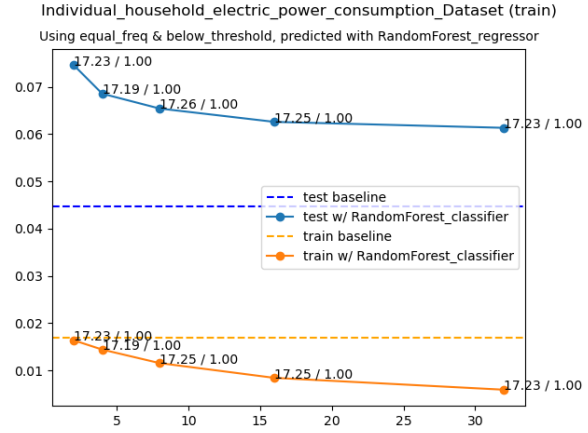
Linear Regression



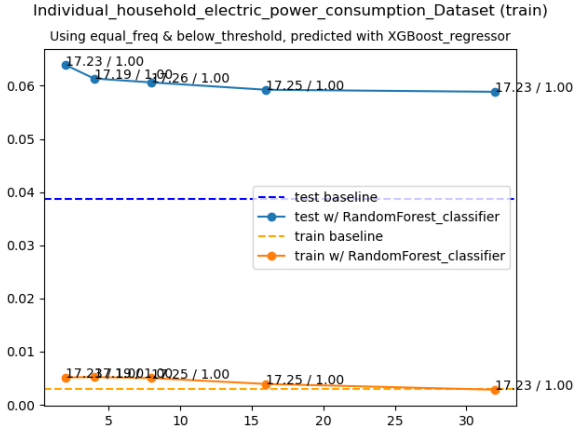
Decision Tree Regressor



Random Forest Regressor



XGBoost Regressor

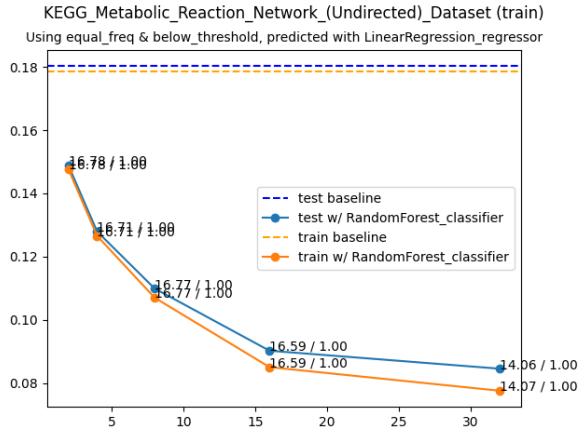


Khiops Regressor

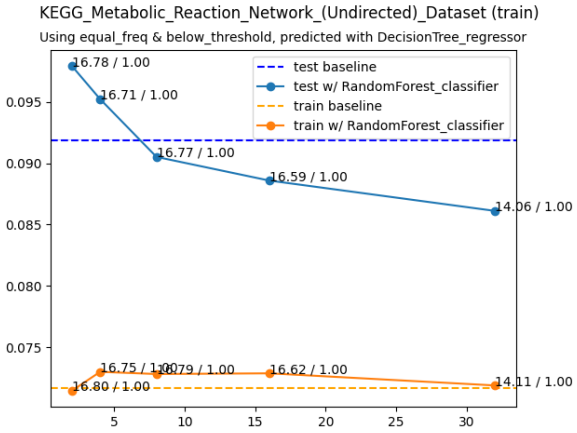
n/a

A.21 KEGG metabolic reaction network (undirected) Dataset

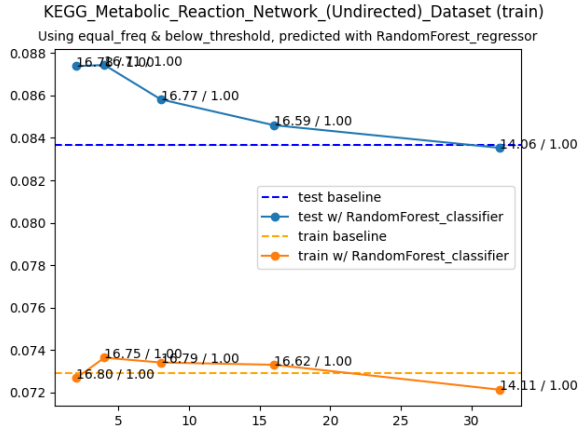
Linear Regression



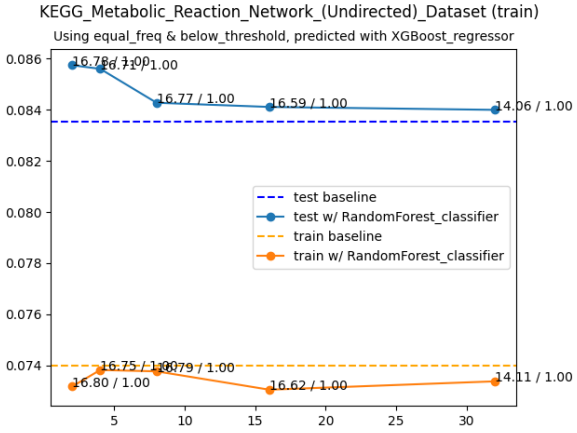
Decision Tree Regressor



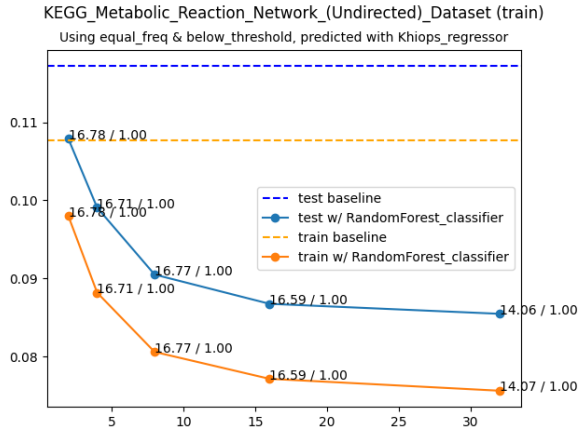
Random Forest Regressor



XGBoost Regressor

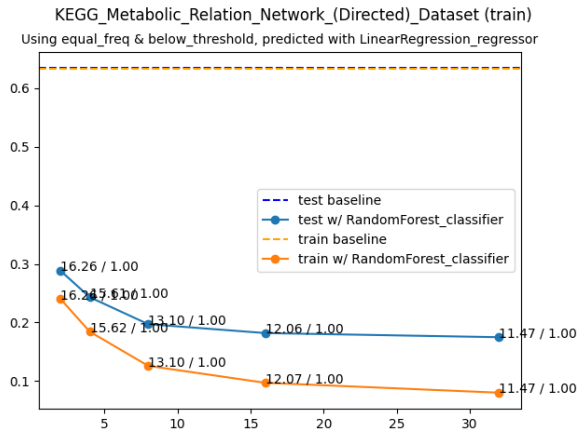


Khiops Regressor

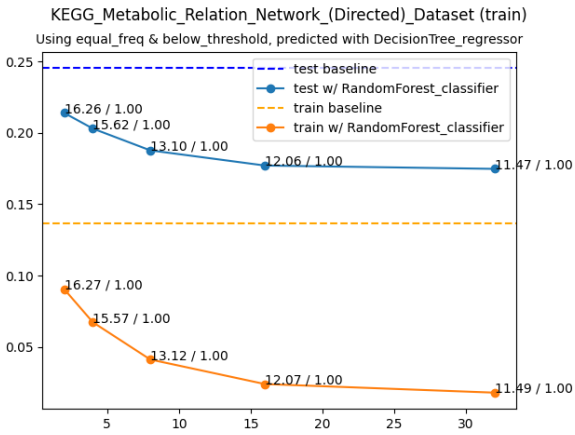


A.22 KEGG metabolic relation network (directed) Dataset

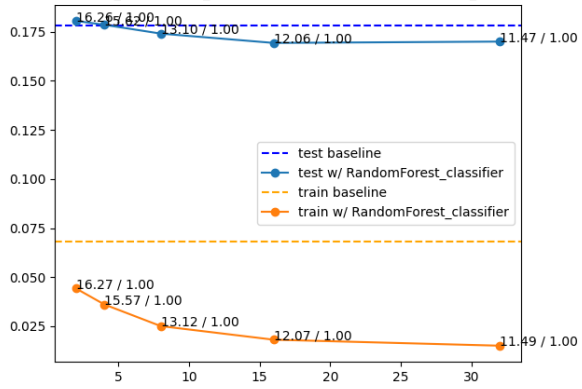
Linear Regression



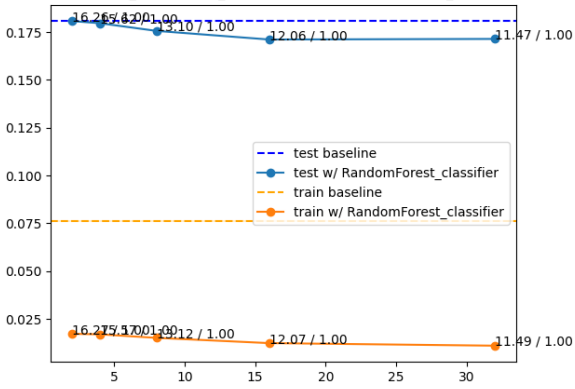
Decision Tree Regressor



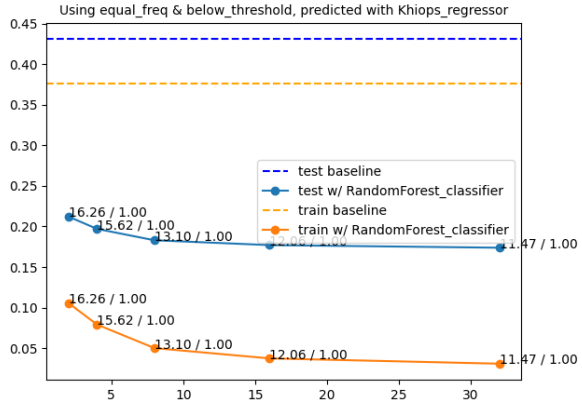
Random Forest Regressor



XGBoost Regressor

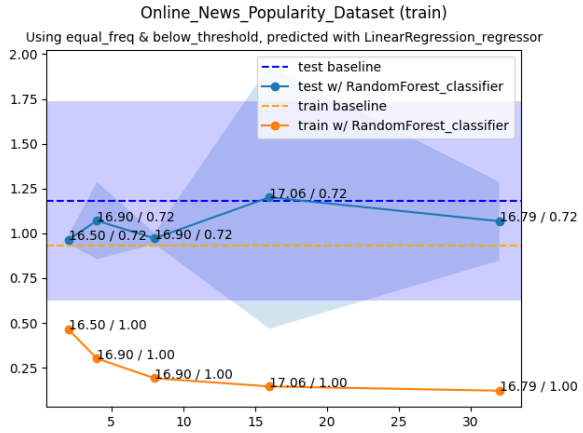


Khiops Regressor

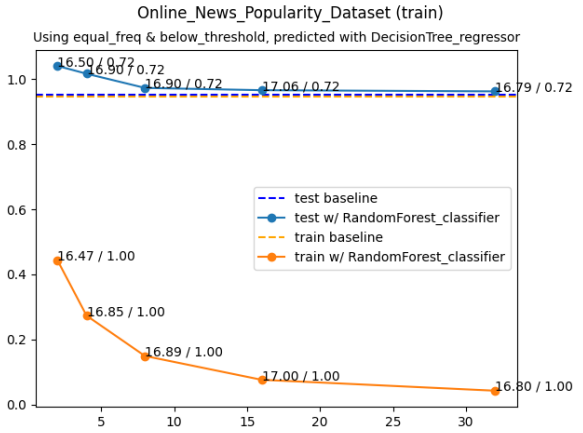


A.23 Online news popularity Dataset

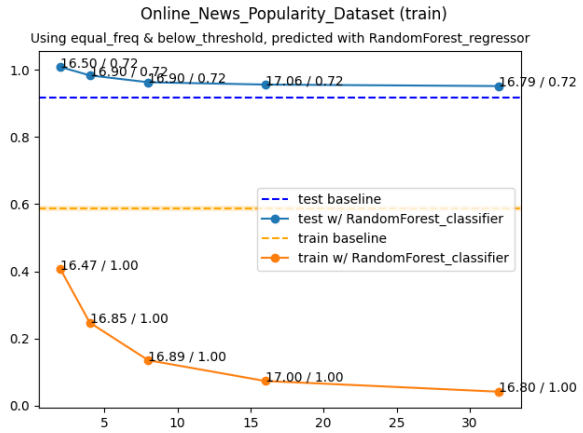
Linear Regression



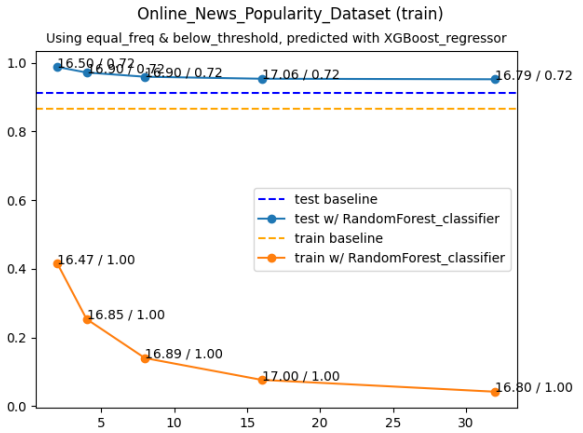
Decision Tree Regressor



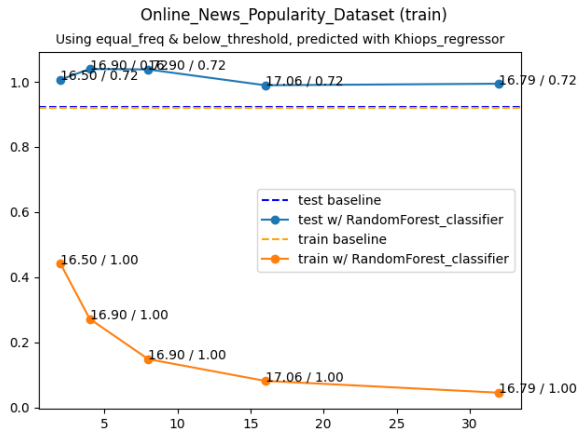
Random Forest Regressor



XGBoost Regressor

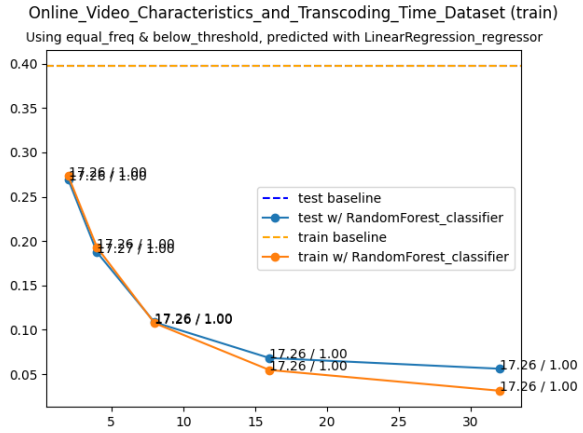


Khiops Regressor

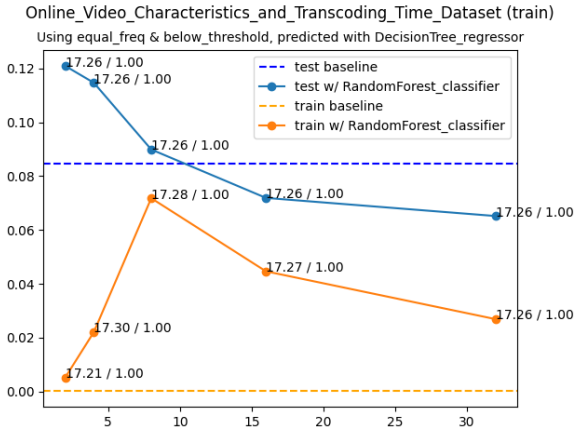


A.24 Online video characteristics and transcoding time dataset Dataset

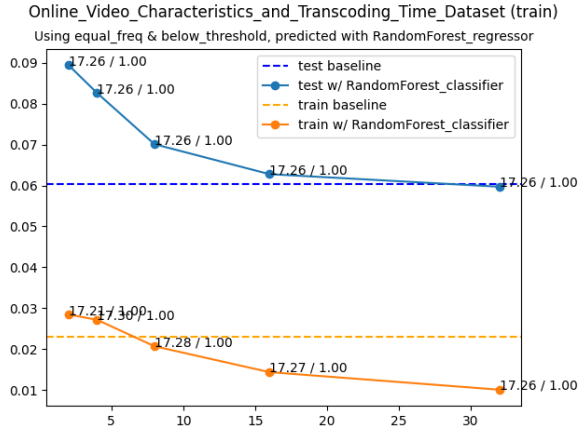
Linear Regression



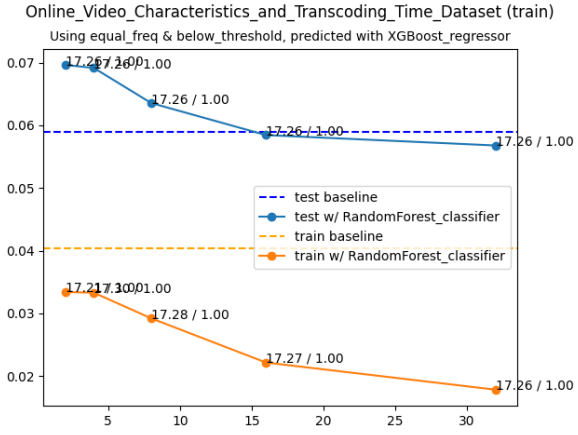
Decision Tree Regressor



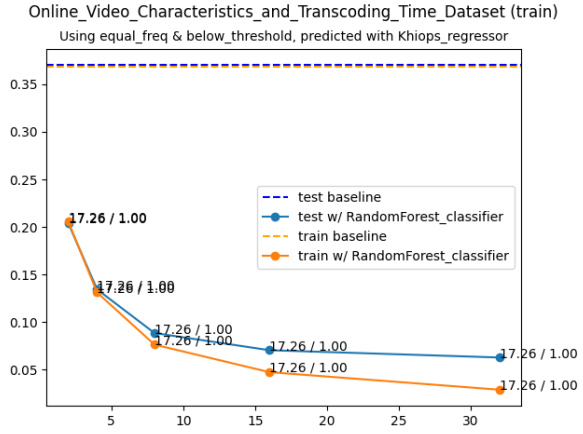
Random Forest Regressor



XGBoost Regressor

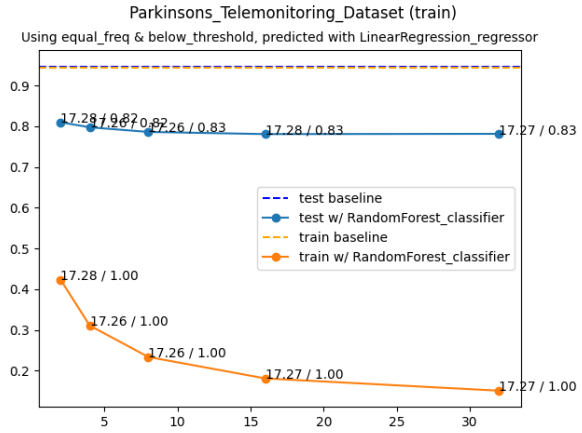


Khiops Regressor

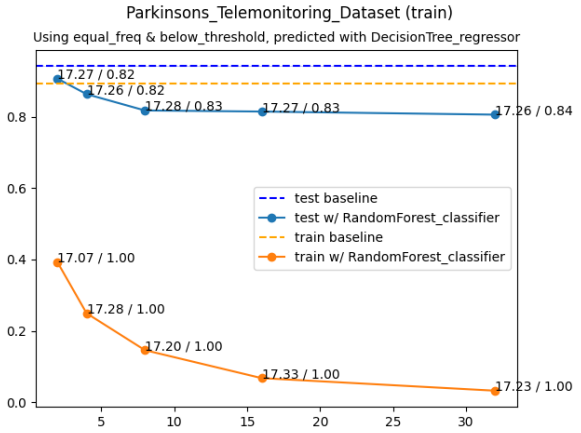


A.25 Parkinsons telemonitoring Dataset

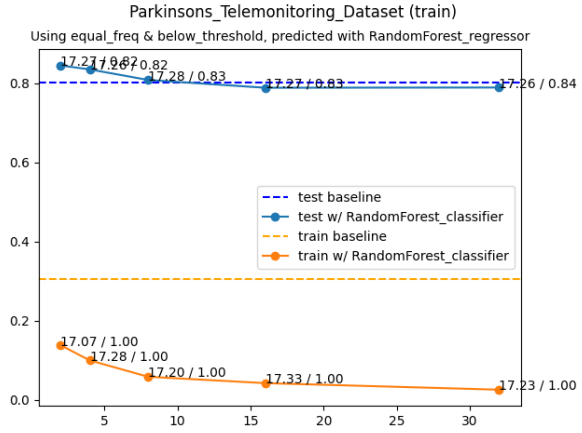
Linear Regression



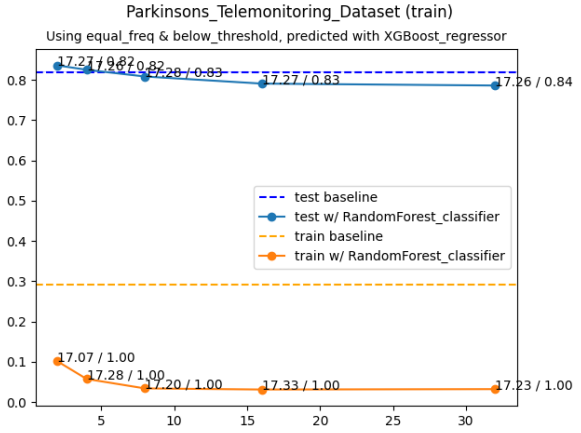
Decision Tree Regressor



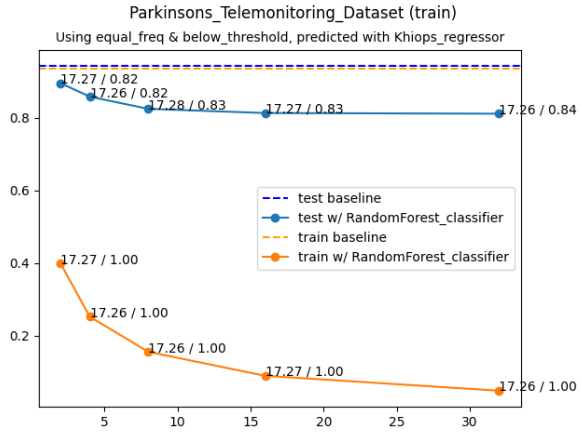
Random Forest Regressor



XGBoost Regressor

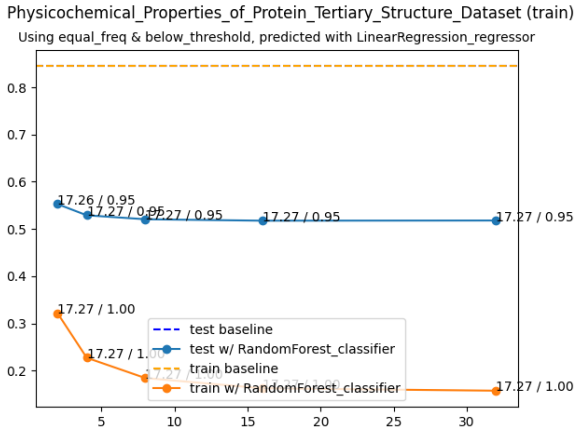


Khiops Regressor

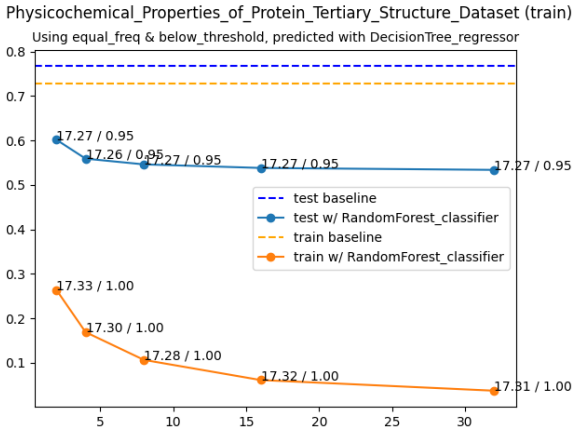


A.26 Physicochemical properties of protein tertiary structure Dataset

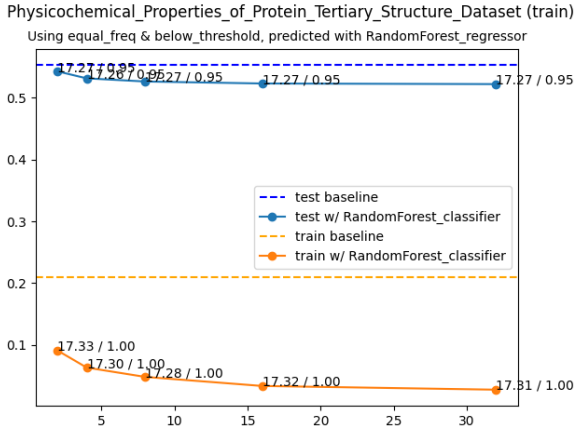
Linear Regression



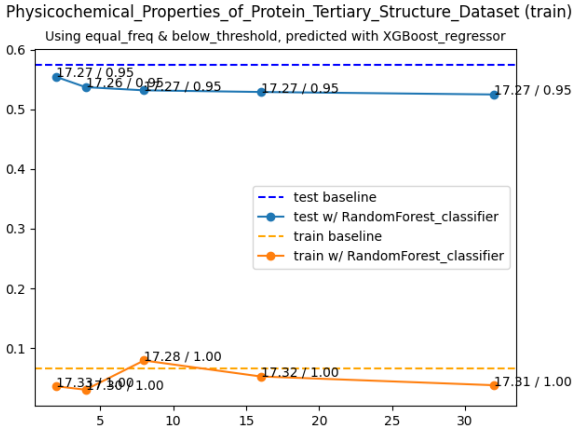
Decision Tree Regressor



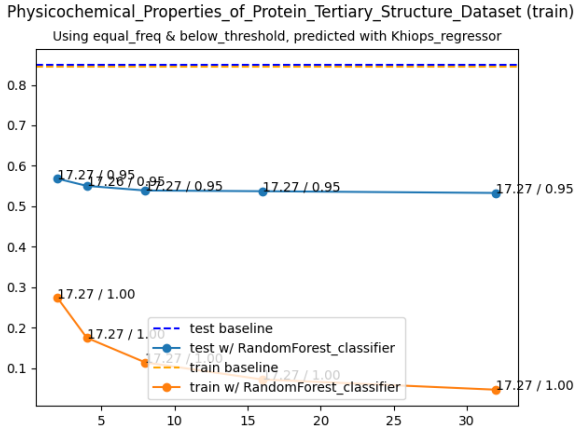
Random Forest Regressor



XGBoost Regressor

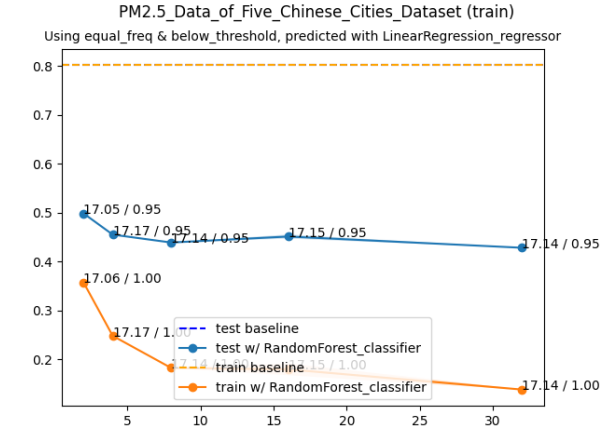


Khiops Regressor

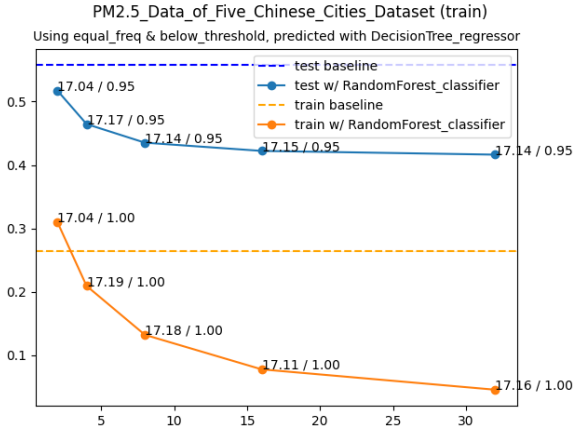


A.27 PM2.5 Data 5 Chinese Cities Dataset

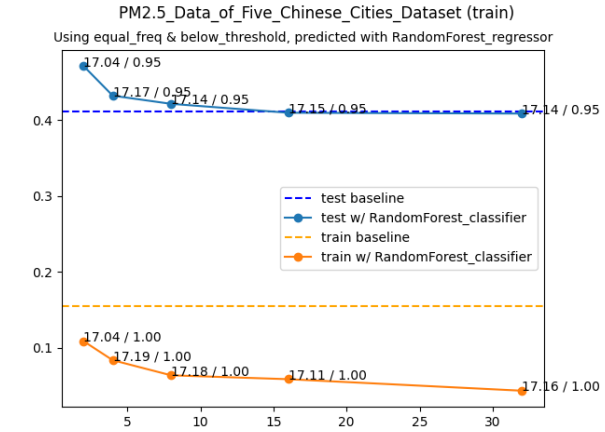
Linear Regression



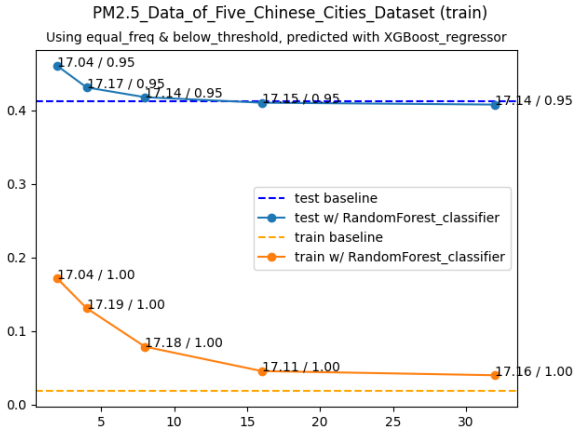
Decision Tree Regressor



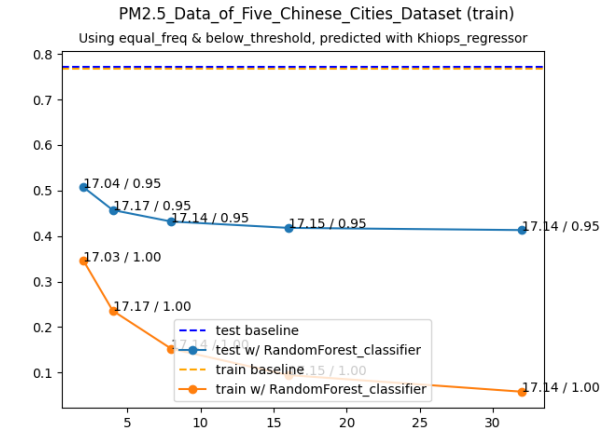
Random Forest Regressor



XGBoost Regressor

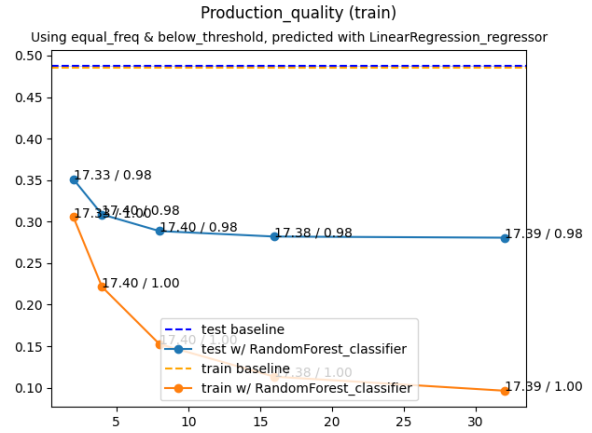


Khiops Regressor

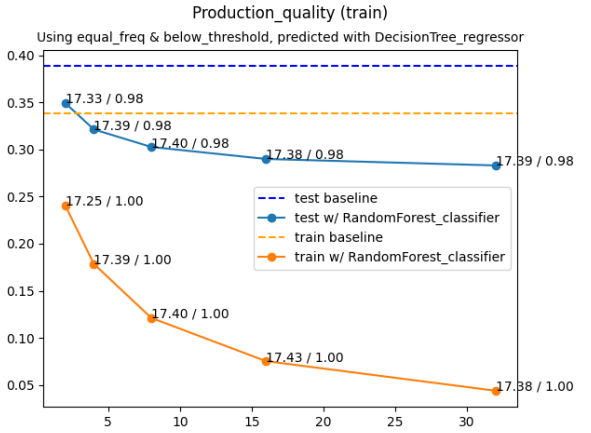


A.28 Production quality Dataset

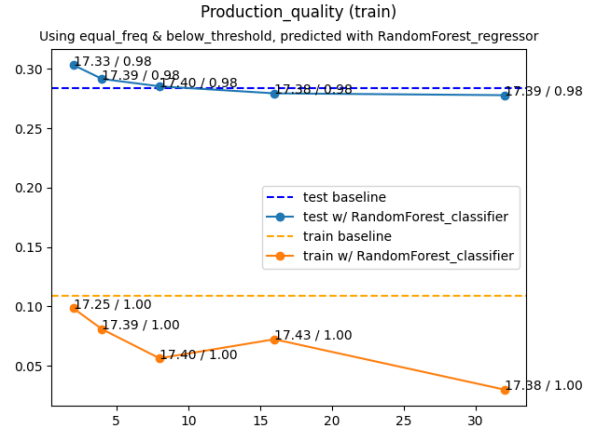
Linear Regression



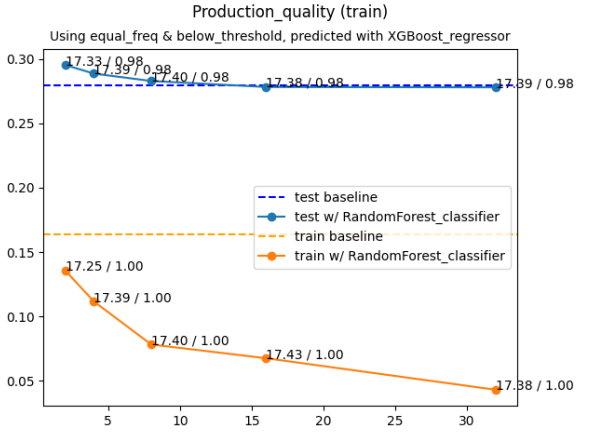
Decision Tree Regressor



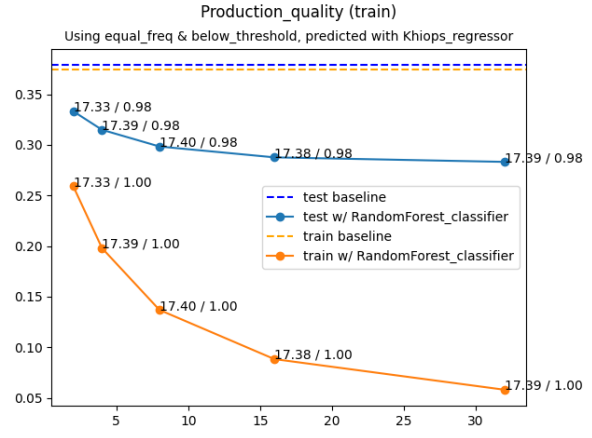
Random Forest Regressor



XGBoost Regressor

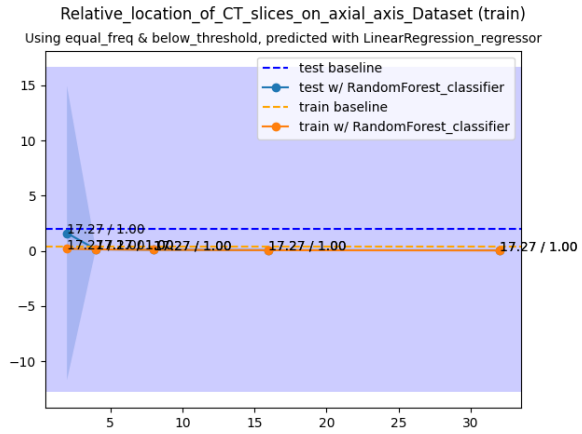


Khiops Regressor

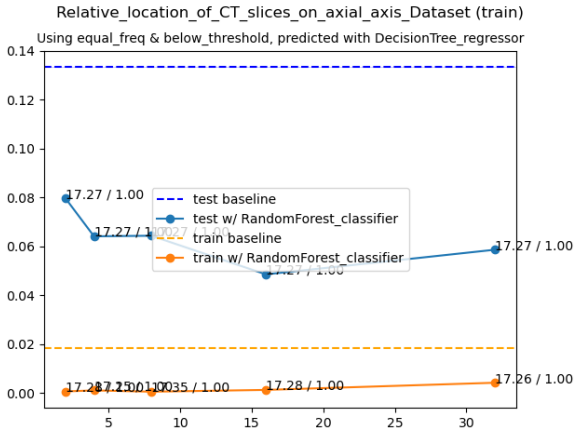


A.29 Relative location of CT slices on axial axis Dataset

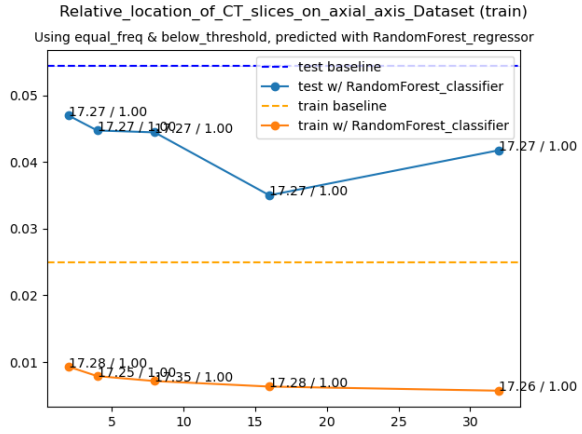
Linear Regression



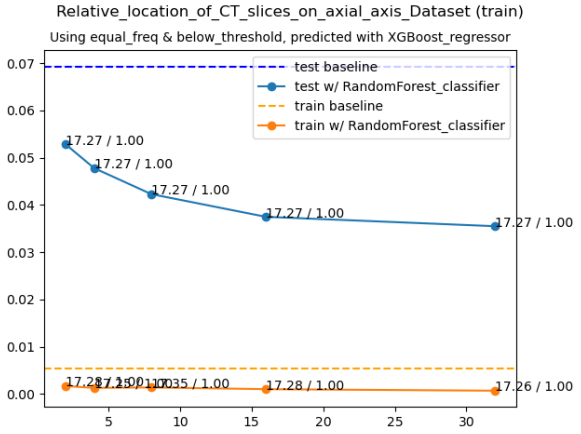
Decision Tree Regressor



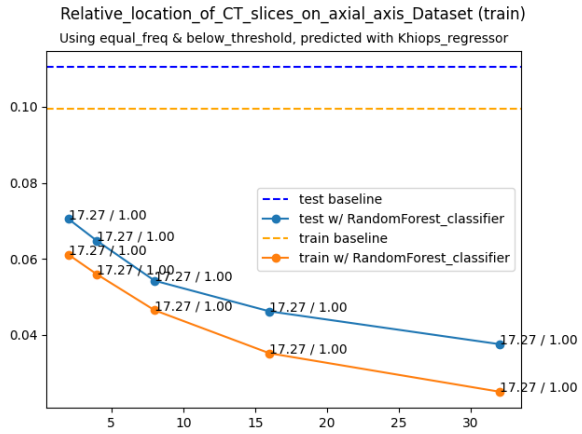
Random Forest Regressor



XGBoost Regressor

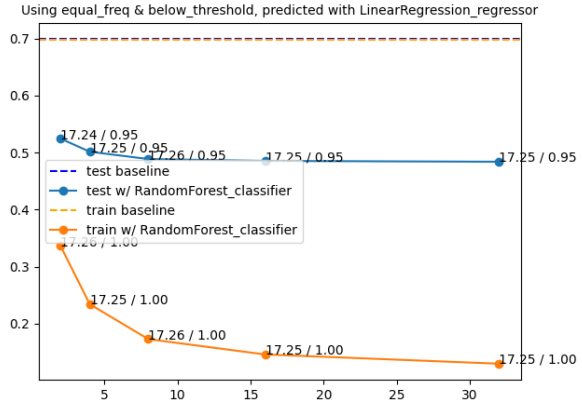


Khiops Regressor

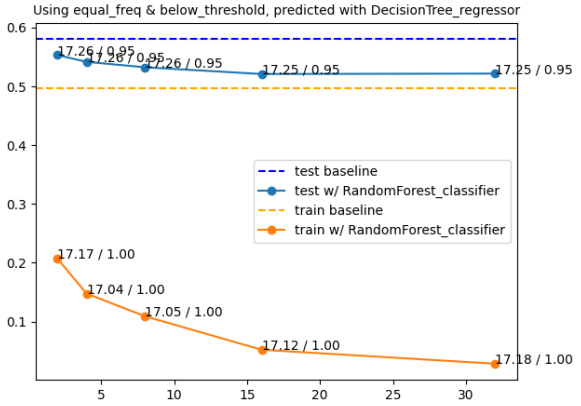


A.30 Seoul Bike Sharing Demand Dataset

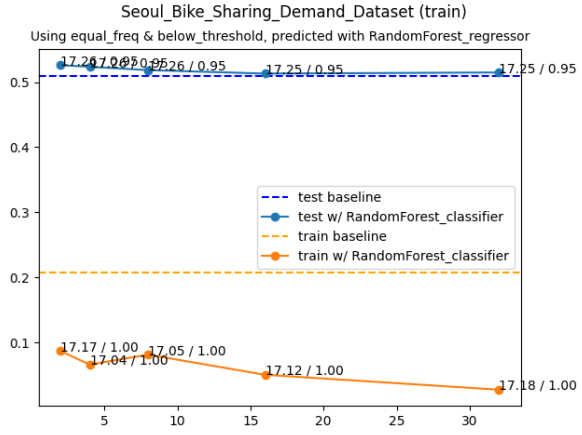
Linear Regression



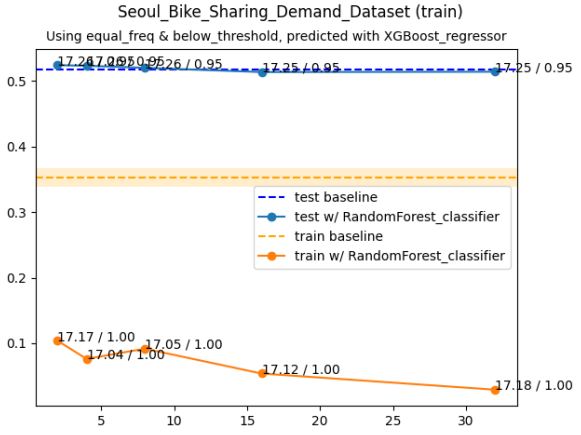
Decision Tree Regressor



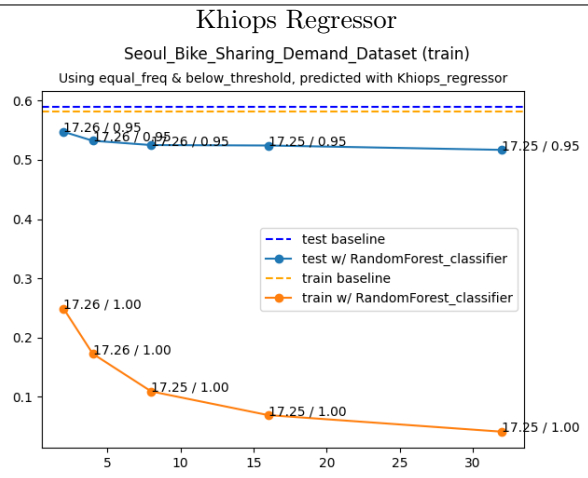
Random Forest Regressor



XGBoost Regressor

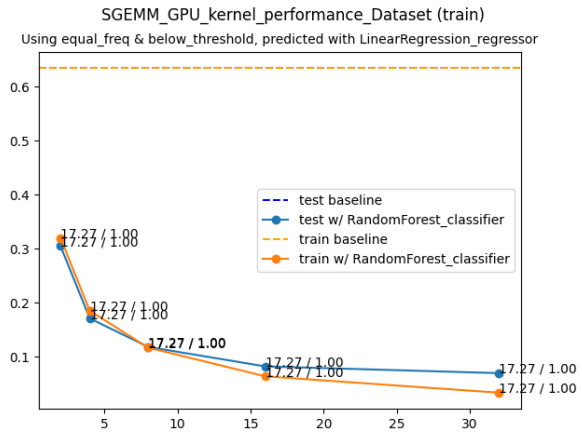


Khiops Regressor

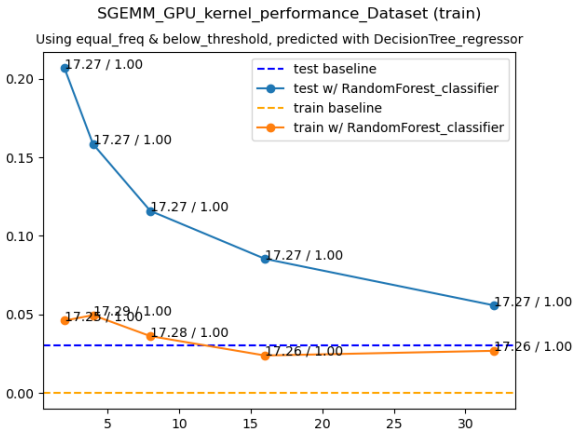


A.31 SGEMM GPU kernel performance Dataset

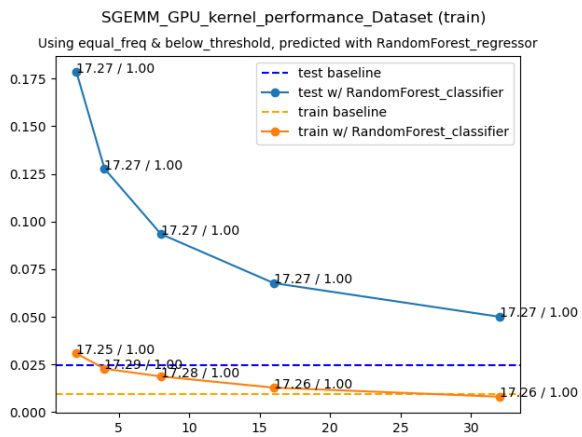
Linear Regression



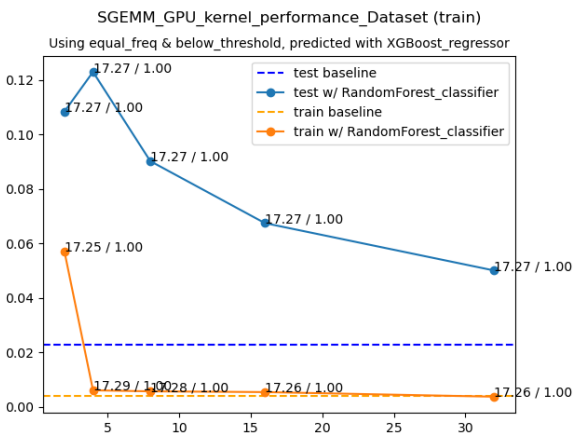
Decision Tree Regressor



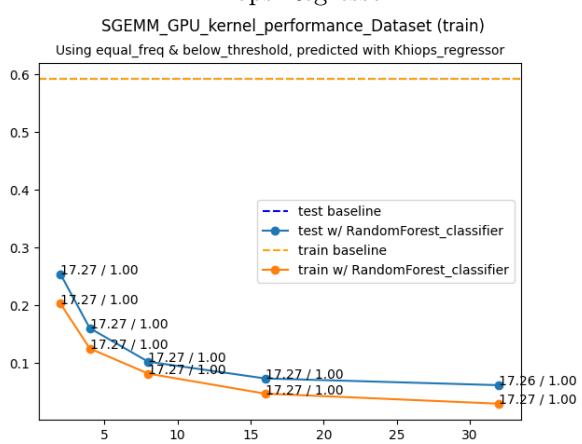
Random Forest Regressor



XGBoost Regressor

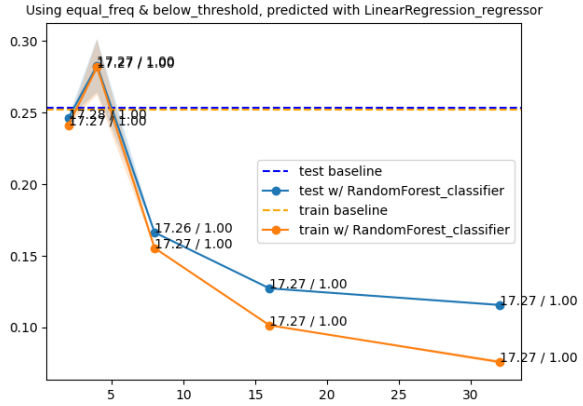


Khiops Regressor

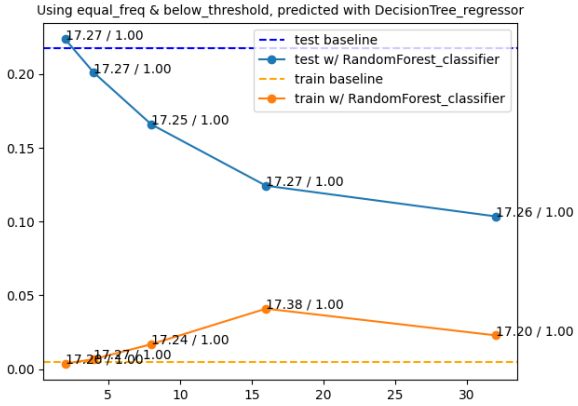


A.32 SML2010 Dataset

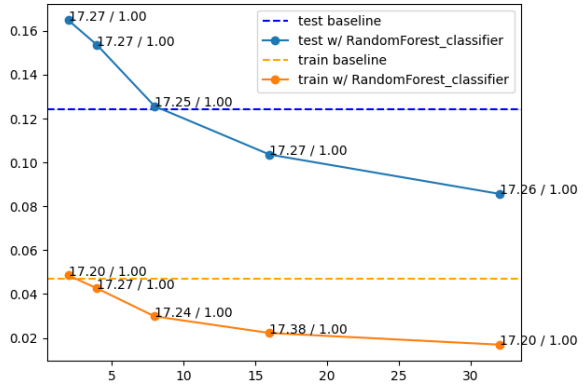
Linear Regression



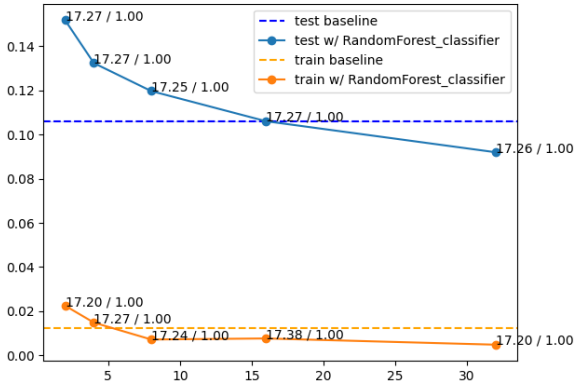
Decision Tree Regressor



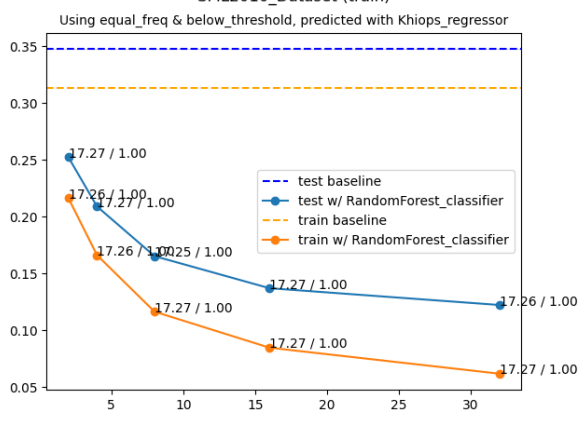
Random Forest Regressor



XGBoost Regressor

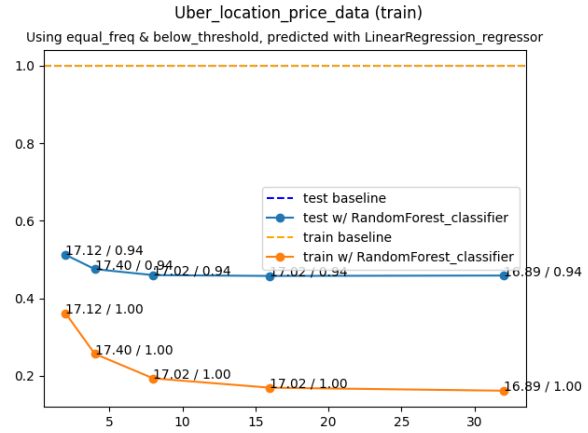


Khiops Regressor

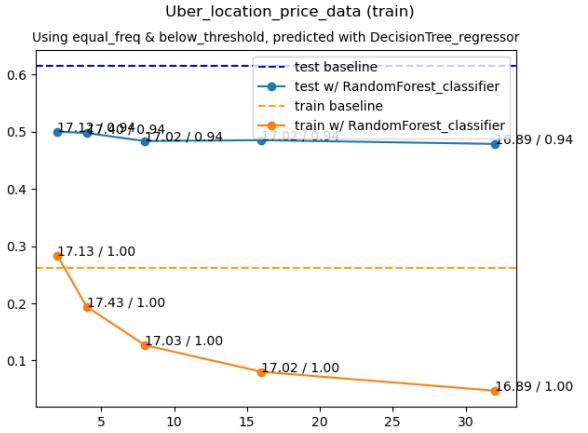


A.33 Uber location price Dataset

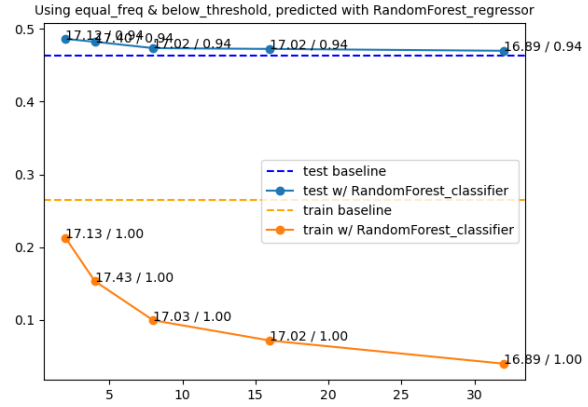
Linear Regression



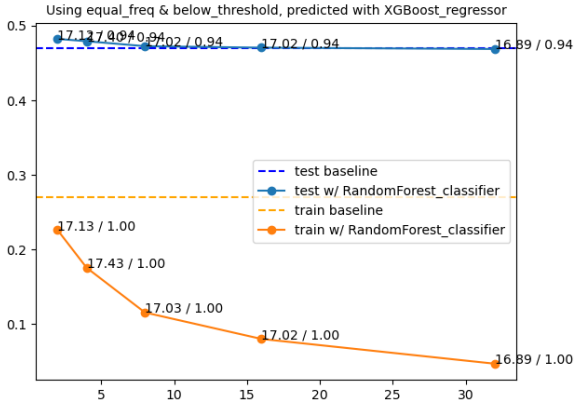
Decision Tree Regressor



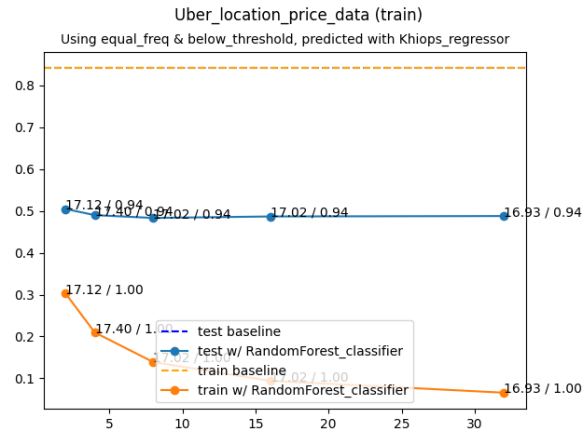
Random Forest Regressor



XGBoost Regressor

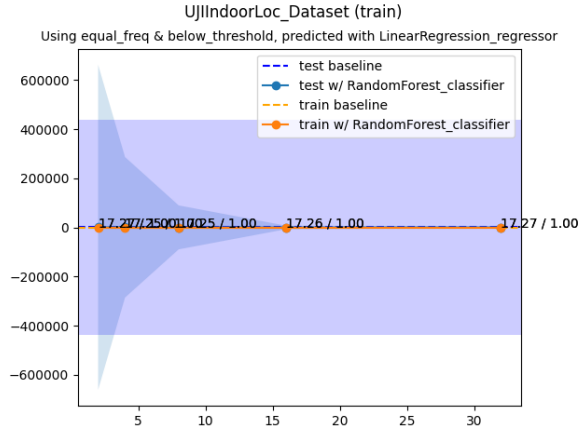


Khiops Regressor

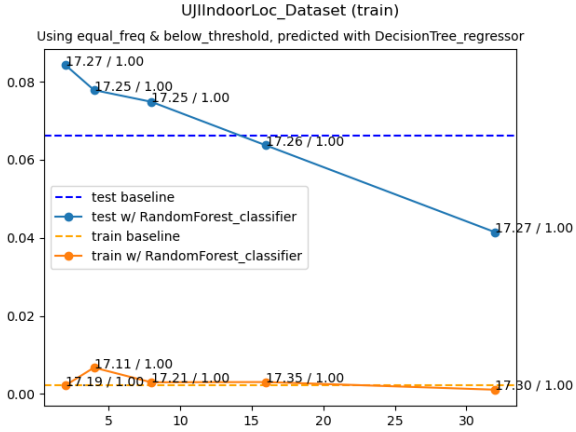


A.34 UJIIndoorLoc Dataset

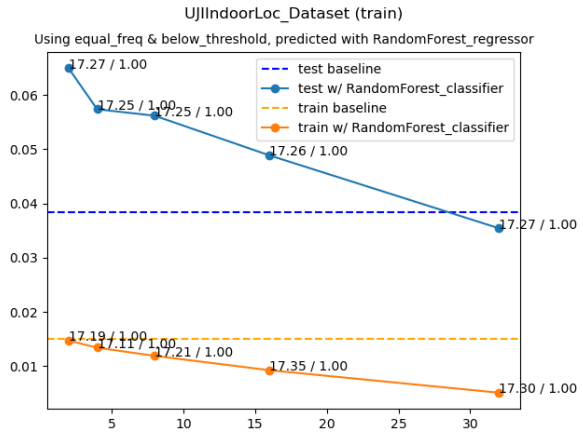
Linear Regression



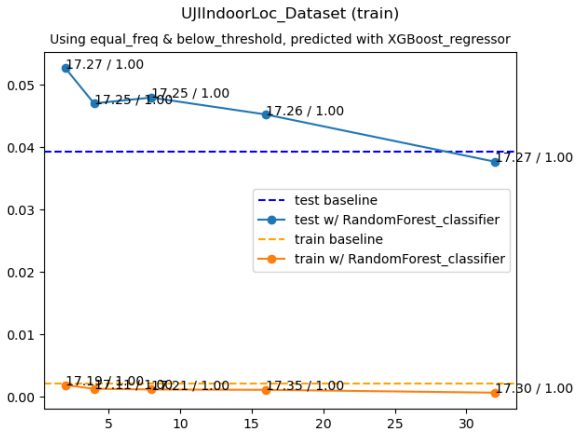
Decision Tree Regressor



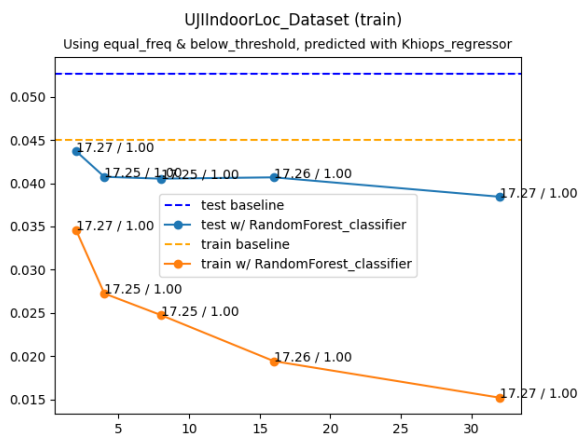
Random Forest Regressor



XGBoost Regressor

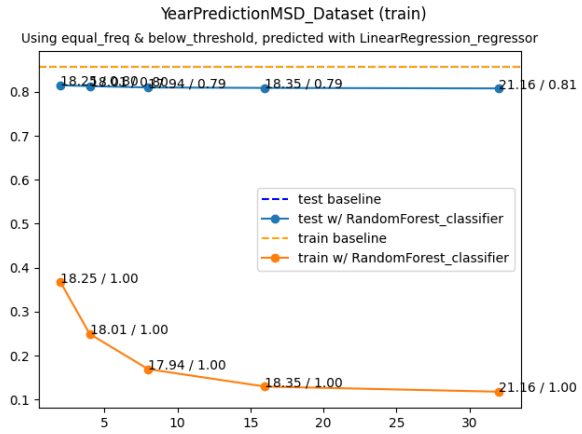


Khiops Regressor

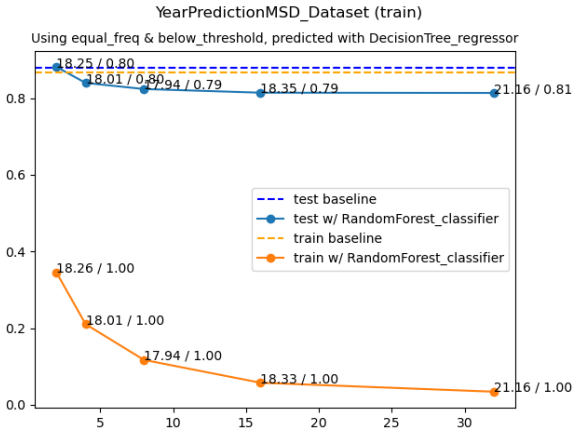


A.35 YearPredictionMSD Dataset

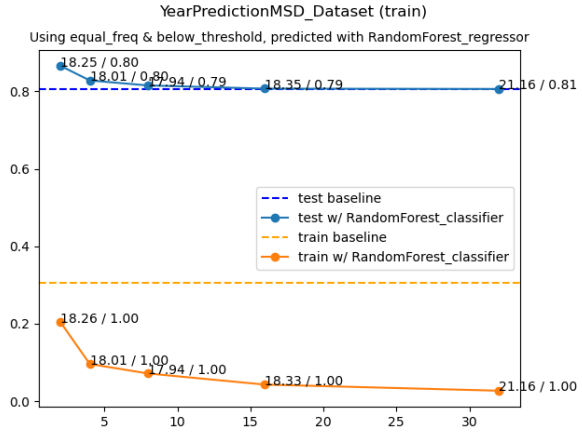
Linear Regression



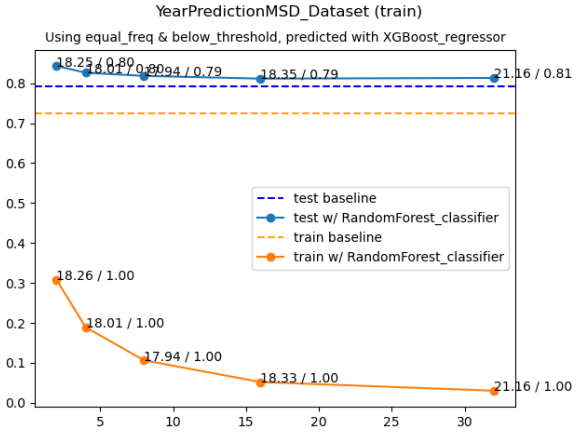
Decision Tree Regressor



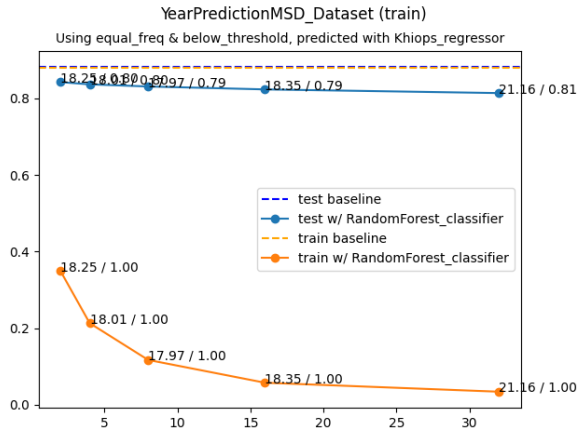
Random Forest Regressor



XGBoost Regressor



Khiops Regressor



B Results in tabular form

B.1 Linear Regression with Random Forest classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,9867	0,3282	0,1994	0,1151	0,0689	0,0488	0,9867	0,3220	0,2028	0,133	0,1027	0,093
air-quality-CO	0,3267	0,2799	0,2482	0,2039	0,1629	0,1255	0,3269	0,3087	0,3004	0,288	0,2777	0,2672
airfoil	0,7158	0,3375	0,2358	0,1577	0,1147	0,0913	0,7176	0,3654	0,2992	0,2569	0,2402	0,2351
appliances-energy	0,8246	0,4130	0,3166	0,2033	0,1923	0,1525	0,8260	0,5579	0,5242	0,4978	0,4943	0,4865
beijing-pm25	0,7830	0,3109	0,2104	0,1508	0,1264	0,1183	0,7833	0,4725	0,4381	0,4244	0,421	0,418
temp-forecast-bias	0,4735	0,3154	0,2386	0,1693	0,1235	0,0960	0,4749	0,3732	0,3311	0,3048	0,2916	0,2847
bike-hour	0,7157	0,2831	0,1827	0,1226	0,0958	0,0862	0,7163	0,3369	0,2827	0,2647	0,2556	0,2547
blog-feedback	0,8300	0,4925	0,4179	0,2769	0,2436	0,2309	0,8307	0,6881	0,6708	0,6486	0,645	0,6434
buzz-twitter	0,7167	0,3554	0,1960	0,1240	0,0929	0,0845	0,7248	0,3648	0,2481	0,2198	0,2168	0,2183
combined-cycle	0,2702	0,2277	0,1778	0,1218	0,0833	0,0652	0,2702	0,2503	0,2265	0,2085	0,1995	0,1967
com-crime	0,5230	0,2965	0,2079	0,1391	0,1004	0,0816	0,6312	0,6377	0,5542	0,5745	0,5488	0,5485
com-crime-unnorm	0,5989	0,3347	0,2311	0,1561	0,1146	0,0959	0,6751	0,6297	0,6076	0,6078	0,6047	0,6031
compress-stren	0,6250	0,3353	0,2372	0,1649	0,1226	0,1036	0,6331	0,3788	0,3138	0,2853	0,2792	0,2732
cond-turbine	0,3001	0,2209	0,1535	0,0852	0,0559	0,0403	0,3002	0,2162	0,1557	0,1161	0,1124	0,1096
cuff-less	0,7996	0,3574	0,3208	0,2883	0,2774	0,2741	0,7996	0,5902	0,5837	0,5786	0,5777	0,5791
electrical-grid-stab	0,5954	0,2839	0,2021	0,1413	0,1066	0,0899	0,5961	0,3476	0,3279	0,3191	0,3172	0,3156
facebook-comment	0,6756	0,4524	0,2782	0,1812	0,1615	0,1531	0,6828	0,5342	0,4530	0,4244	0,4241	0,4242
geo-lat	0,7953	0,3947	0,3260	0,2076	0,1583	0,1328	0,8752	0,8300	0,8333	0,8144	0,8108	0,8066
greenhouse-net	0,6491	0,3023	0,2358	0,1916	0,1684	0,1572	0,6492	0,5520	0,5451	0,5419	0,5409	0,5418
household-consume	0,4820	0,2581	0,1973	0,1323	0,0822	0,0576	0,4821	0,2585	0,1999	0,1394	0,0975	0,0801
KEGG-reaction	0,1787	0,1477	0,1265	0,1070	0,0850	0,0776	0,1803	0,1488	0,1279	0,1099	0,0902	0,0845
KEGG-relation	0,6329	0,2402	0,1845	0,1261	0,0972	0,0804	0,6342	0,2883	0,2432	0,1973	0,182	0,175
online-news	0,9333	0,4646	0,3015	0,1918	0,1459	0,1222	1,1822	0,9649	1,0709	0,9717	1,2014	1,0677
video-transcode	0,3972	0,2733	0,1928	0,1075	0,0546	0,0313	0,3975	0,2692	0,1873	0,1085	0,0682	0,0561
pm25-chengdu-us	0,8014	0,3575	0,2482	0,1826	0,1798	0,1385	0,8022	0,4986	0,4552	0,4391	0,4513	0,4283
park-total-UPDRS	0,9438	0,4237	0,3108	0,2338	0,1810	0,1509	0,9472	0,8100	0,7977	0,7863	0,781	0,7816
physico-protein	0,8451	0,3216	0,2271	0,1842	0,1624	0,1573	0,8453	0,5526	0,5289	0,5206	0,5175	0,5179
production-quality	0,4856	0,3058	0,2214	0,1525	0,1133	0,0964	0,4872	0,3512	0,3086	0,2886	0,282	0,2807
CT-slices	0,3554	0,1940	0,1394	0,0941	0,0583	0,0350	1,9533	1,6078	0,1387	0,0975	0,0673	0,0504
gpu-kernel-perf	0,6346	0,3200	0,1851	0,1168	0,0638	0,0334	0,6347	0,3049	0,1710	0,1181	0,0821	0,0696
SML2010	0,2519	0,2413	0,2818	0,1555	0,1015	0,0759	0,2536	0,2462	0,2828	0,1664	0,1273	0,1157
seoul-bike-sharing	0,6979	0,3363	0,2341	0,1729	0,1459	0,1298	0,6990	0,5244	0,5014	0,4886	0,4853	0,4839
UJ-lat	0,1816	0,1559	0,1380	0,0968	0,0624	0,0384	~~631,5~~	~~646,3~~	~~545,5~~	~~296,1~~	~~66,97~~	~~31,22~~
uber-location-price	0,9998	0,3619	0,2571	0,1937	0,1695	0,1618	0,9998	0,5125	0,4755	0,46	0,4579	0,4589
year-prediction	0,8567	0,3669	0,2491	0,1691	0,1294	0,1178	0,8569	0,8150	0,8131	0,8100	0,809	0,8079
Average :	0,6383	0,3216	0,2345	0,1624	0,1276	0,1086	0,7016	0,4973	0,4177	0,3885	0,3841	0,3752

B.2 Decision Tree regressor with Random Forest classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,0099	0,0169	0,0148	0,0676	0,0378	0,0327	0,1213	0,1918	0,1571	0,1197	0,0947	0,0831
air-quality-CO	0,2680	0,2079	0,1738	0,1214	0,0828	0,0504	0,3255	0,3146	0,3103	0,2939	0,2831	0,2727
airfoil	0,0142	0,1716	0,1625	0,1315	0,0665	0,0173	0,4373	0,3849	0,3530	0,3166	0,3361	0,3194
appliances-energy	0,3083	0,3930	0,2950	0,1570	0,1405	0,0694	0,7431	0,5973	0,5529	0,5324	0,5237	0,5267
beijing-pm25	0,2926	0,2811	0,1812	0,1083	0,0612	0,0336	0,6013	0,4944	0,4484	0,4153	0,4047	0,3948
temp-forecast-bias	0,3630	0,2243	0,2026	0,1130	0,0687	0,0407	0,4668	0,3899	0,3485	0,3144	0,2942	0,2848
bike-hour	0,0870	0,1920	0,1454	0,0957	0,0551	0,0305	0,3294	0,3249	0,2960	0,2846	0,2651	0,2564
blog-feedback	0,6305	0,4364	0,3694	0,2210	0,1693	0,1468	0,6741	0,7093	0,7028	0,6732	0,6750	0,6694
buzz-twitter	0,2237	0,1910	0,1463	0,1075	0,0672	0,0375	0,2299	0,2327	0,2355	0,2381	0,2293	0,2265
combined-cycle	0,2088	0,1789	0,1410	0,0960	0,0584	0,0326	0,2647	0,2517	0,2372	0,2213	0,2066	0,2014
com-crime	0,5654	0,2897	0,1862	0,1092	0,0619	0,0108	0,6418	0,6207	0,5961	0,5701	0,5564	0,5552
com-crime-unnorm	0,6292	0,3131	0,2147	0,1262	0,0759	0,0183	0,7036	0,6993	0,6691	0,6469	0,6356	0,6251
compress-stren	0,1136	0,1648	0,1940	0,1545	0,0845	0,0215	0,4519	0,3996	0,3614	0,3283	0,3254	0,3173
cond-turbine	0,0050	0,0030	0,0056	0,0431	0,0224	0,0001	0,1764	0,1879	0,1584	0,1226	0,1123	0,1094
cuff-less	0,4953	0,3142	0,3054	0,2677	0,2575	0,2544	0,5165	0,6049	0,6220	0,6237	0,6247	0,6255
electrical-grid-stab	0,4180	0,2733	0,1955	0,1307	0,0682	0,0395	0,5545	0,4344	0,3968	0,3534	0,3323	0,3245
facebook-comment	0,3968	0,2881	0,2035	0,1488	0,0802	0,0395	0,4655	0,4893	0,4630	0,4535	0,4447	0,4429
geo-lat	0,7981	0,3545	0,2655	0,1447	0,0897	0,0138	0,9594	0,9408	0,8708	0,8949	0,8583	0,8538
greenhouse-net	0,4952	0,2600	0,1730	0,1053	0,0601	0,0331	0,5397	0,5981	0,6022	0,6013	0,5773	0,5686
household-consume	0,0002	0,0151	0,0410	0,0338	0,0252	0,0159	0,0680	0,0893	0,0809	0,0740	0,0674	0,0636
KEGG-reaction	0,0717	0,0715	0,0730	0,0728	0,0729	0,0719	0,0919	0,0979	0,0952	0,0905	0,0886	0,0861
KEGG-relation	0,1368	0,0906	0,0674	0,0414	0,0242	0,0182	0,2454	0,2139	0,2030	0,1877	0,1771	0,1748
online-news	0,9459	0,4438	0,2732	0,1494	0,0762	0,0427	0,9516	1,0404	1,0164	0,9728	0,9658	0,9622
video-transcode	0,0003	0,0051	0,0221	0,0718	0,0446	0,0269	0,0846	0,1210	0,1147	0,0899	0,0719	0,0652
pm25-chengdu-us	0,2635	0,3100	0,2093	0,1321	0,0777	0,0456	0,5578	0,5174	0,4646	0,4352	0,4222	0,4164
park-total-UPDRS	0,8939	0,3927	0,2493	0,1459	0,0676	0,0324	0,9416	0,9067	0,8631	0,8174	0,8144	0,8057
physico-protein	0,7286	0,2641	0,1690	0,1066	0,0613	0,0374	0,7676	0,6016	0,5590	0,5462	0,5382	0,5338
production-quality	0,3386	0,2407	0,1785	0,1213	0,0754	0,0440	0,3886	0,3493	0,3212	0,3026	0,2899	0,2830
CT-slices	0,0185	0,0007	0,0011	0,0006	0,0013	0,0043	0,1332	0,0797	0,0641	0,0644	0,0486	0,0586
gpu-kernel-perf	0,0002	0,0460	0,0495	0,0362	0,0238	0,0268	0,0300	0,2069	0,1584	0,1159	0,0854	0,0557
SML2010	0,0046	0,0037	0,0068	0,0169	0,0410	0,0229	0,2179	0,2236	0,2010	0,1659	0,1243	0,1036
seoul-bike-sharing	0,4965	0,2072	0,1476	0,1091	0,0521	0,0284	0,5801	0,5532	0,5416	0,5324	0,5210	0,5217
UJ-lat	0,0022	0,0022	0,0068	0,0030	0,0030	0,0011	0,0662	0,0843	0,0779	0,0749	0,0637	0,0414
uber-location-price	0,2621	0,2834	0,1941	0,1269	0,0803	0,0471	0,6153	0,5003	0,4979	0,4839	0,4854	0,4789
year-prediction	0,8670	0,3449	0,2101	0,1169	0,0573	0,0339	0,8783	0,8816	0,8397	0,8239	0,8143	0,8137
Average :	0,3244	0,2079	0,1564	0,1067	0,0683	0,0406	0,4520	0,4381	0,4137	0,3938	0,3816	0,3749

B.3 Random Forest regressor with Random Forest classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,0353	0,0459	0,0374	0,0262	0,0166	0,0101	0,0908	0,1506	0,1245	0,1000	0,0852	0,0792
air-quality-CO	0,1079	0,0942	0,0754	0,0834	0,0626	0,0354	0,2667	0,2786	0,2750	0,2704	0,2645	0,2642
airfoil	0,1171	0,1005	0,0839	0,0525	0,0383	0,0245	0,2987	0,3146	0,2937	0,2765	0,2956	0,2865
appliances-energy	0,2041	0,1634	0,2341	0,1443	0,1363	0,0543	0,5456	0,5427	0,5326	0,5224	0,5194	0,5164
beijing-pm25	0,1577	0,0983	0,0697	0,0471	0,0596	0,0328	0,4168	0,4545	0,4227	0,4049	0,3961	0,3874
temp-forecast-bias	0,1252	0,0889	0,0719	0,0491	0,0460	0,0277	0,3237	0,3194	0,2934	0,2862	0,2780	0,2771
bike-hour	0,0964	0,0738	0,0601	0,0416	0,0532	0,0269	0,2415	0,2916	0,2718	0,2668	0,2537	0,2499
blog-feedback	0,6074	0,3933	0,3274	0,2085	0,1668	0,1439	0,6481	0,6754	0,6762	0,6670	0,6662	0,6618
buzz-twitter	0,1471	0,1292	0,1035	0,0778	0,0617	0,0373	0,2170	0,2215	0,2263	0,2253	0,2231	0,2200
combined-cycle	0,0841	0,0698	0,0567	0,0505	0,0557	0,0314	0,2079	0,2167	0,2143	0,2067	0,1996	0,1952
com-crime	0,2572	0,2449	0,1801	0,0999	0,0535	0,0135	0,5525	0,5836	0,5688	0,5577	0,5524	0,5505
com-crime-unnorm	0,3214	0,1218	0,1461	0,0823	0,0634	0,0162	0,6163	0,6523	0,6383	0,6305	0,6251	0,6157
compress-stren	0,1327	0,1146	0,0849	0,0847	0,0699	0,0348	0,3137	0,3188	0,3102	0,2933	0,2908	0,2896
cond-turbine	0,0419	0,0416	0,0334	0,0363	0,0143	0,0002	0,1117	0,1489	0,1275	0,1123	0,1067	0,1064
cuff-less	0,4880	0,3090	0,2811	0,2587	0,2442	0,2371	0,5025	0,5981	0,6072	0,6137	0,6132	0,6105
electrical-grid-stab	0,1269	0,0953	0,0726	0,0954	0,0494	0,0282	0,3352	0,3773	0,3306	0,3179	0,3109	0,3068
facebook-comment	0,1897	0,1350	0,1287	0,1296	0,0779	0,0242	0,4162	0,4499	0,4435	0,4406	0,4390	0,4384
geo-lat	0,3275	0,1477	0,1062	0,0662	0,0415	0,0364	0,8321	0,8818	0,8351	0,8329	0,8386	0,8287
greenhouse-net	0,4832	0,2320	0,1530	0,0931	0,0536	0,0294	0,5158	0,5820	0,5802	0,5754	0,5671	0,5638
household-consume	0,0170	0,0164	0,0144	0,0115	0,0084	0,0059	0,0447	0,0746	0,0686	0,0654	0,0626	0,0613
KEGG-reaction	0,0729	0,0727	0,0736	0,0734	0,0733	0,0721	0,0837	0,0874	0,0874	0,0858	0,0846	0,0835
KEGG-relation	0,0681	0,0442	0,0361	0,0251	0,0182	0,0151	0,1782	0,1805	0,1786	0,1741	0,1693	0,1700
online-news	0,5876	0,4059	0,2471	0,1349	0,0730	0,0410	0,9177	1,1988	0,9840	0,9635	0,9566	0,9521
video-transcode	0,0229	0,0285	0,0272	0,0207	0,0144	0,0101	0,0604	0,0895	0,0826	0,0701	0,0628	0,0597
pm25-chengdu-us	0,1549	0,1089	0,0838	0,0639	0,0588	0,0437	0,4111	0,4710	0,4319	0,4212	0,4094	0,4084
park-total-UPDRS	0,3045	0,1385	0,1001	0,0587	0,0426	0,0261	0,8020	0,8449	0,8350	0,8083	0,7887	0,7893
physico-protein	0,2096	0,0912	0,0636	0,0486	0,0339	0,0279	0,5528	0,5424	0,5311	0,5260	0,5228	0,5218
production-quality	0,1093	0,0984	0,0808	0,0563	0,0723	0,0299	0,2836	0,3034	0,2916	0,2854	0,2795	0,2779
CT-slices	0,0249	0,0093	0,0079	0,0072	0,0063	0,0057	0,0544	0,0470	0,0447	0,0444	0,0350	0,0418
gpu-kernel-perf	0,0093	0,0308	0,0228	0,0187	0,0129	0,0081	0,0246	0,1785	0,1276	0,0934	0,0677	0,0501
SML2010	0,0468	0,0485	0,0426	0,0298	0,0223	0,0169	0,1241	0,1648	0,1535	0,1257	0,1036	0,0857
seoul-bike-sharing	0,2070	0,0866	0,0659	0,0812	0,0502	0,0273	0,5101	0,5263	0,5237	0,5186	0,5132	0,5149
UJ-lat	0,0151	0,0147	0,0134	0,0119	0,0093	0,0051	0,0383	0,0650	0,0574	0,0562	0,0489	0,0354
uber-location-price	0,2650	0,2127	0,1535	0,0997	0,0719	0,0401	0,4635	0,4863	0,4823	0,4736	0,4725	0,4699
year-prediction	0,3050	0,2035	0,0959	0,0713	0,0427	0,0269	0,8059	0,8658	0,8283	0,8153	0,8071	0,8062
Average :	0,1849	0,1231	0,0981	0,0726	0,0564	0,0356	0,3659	0,4053	0,3851	0,3751	0,3688	0,3650

B.4 XGBoost regressor with Random Forest classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,0023	0,0066	0,0062	0,0041	0,0029	0,0016	0,0935	0,1406	0,1171	0,0951	0,0829	0,0781
air-quality-CO	0,1409	0,1397	0,1483	0,1083	0,0722	0,0253	0,2682	0,2720	0,2702	0,2646	0,2600	0,2633
airfoil	0,0444	0,0789	0,0748	0,0657	0,0457	0,0237	0,2814	0,2930	0,2714	0,2670	0,2905	0,2752
appliances-energy	0,1416	0,2259	0,1684	0,1385	0,1220	0,0538	0,5790	0,5412	0,5280	0,5221	0,5183	0,5203
beijing-pm25	0,2359	0,1344	0,1330	0,0823	0,0465	0,0317	0,4149	0,4402	0,4163	0,4010	0,3964	0,3885
temp-forecast-bias	0,0869	0,0660	0,0503	0,1006	0,0522	0,0377	0,3085	0,3193	0,2942	0,2848	0,2771	0,2773
bike-hour	0,1372	0,0986	0,0800	0,0760	0,0555	0,0308	0,2208	0,2713	0,2591	0,2574	0,2517	0,2500
blog-feedback	0,5841	0,4029	0,3374	0,2016	0,1550	0,1339	0,6444	0,6692	0,6721	0,6661	0,6703	0,6652
buzz-twitter	0,1919	0,1613	0,1279	0,0957	0,0612	0,0376	0,2162	0,2205	0,2258	0,2256	0,2232	0,2205
combined-cycle	0,0961	0,0919	0,0791	0,0552	0,0459	0,0322	0,2041	0,2068	0,2082	0,2030	0,1983	0,1950
com-crime	0,2691	0,1590	0,1065	0,0641	0,0351	0,0171	0,5665	0,5783	0,5629	0,5546	0,5535	0,5503
com-crime-unnorm	0,2842	0,1726	0,1199	0,0711	0,0406	0,0225	0,6186	0,6412	0,6233	0,6299	0,6196	0,6124
compress-stren	0,1065	0,0949	0,0957	0,0855	0,0728	0,0254	0,2790	0,2991	0,2974	0,2875	0,2858	0,2955
cond-turbine	0,0089	0,0170	0,0044	0,0257	0,0099	0,0000	0,1062	0,1428	0,1242	0,1098	0,1085	0,1094
cuff-less	0,4925	0,3148	0,2828	0,2453	0,2272	0,2189	0,5012	0,5938	0,6079	0,6167	0,6194	0,6225
electrical-grid-stab	0,0572	0,1372	0,1066	0,0907	0,0636	0,0327	0,2634	0,3368	0,3086	0,3173	0,3136	0,3092
facebook-comment	0,3808	0,2354	0,1861	0,1330	0,0722	0,0352	0,4095	0,4373	0,4374	0,4387	0,4368	0,4397
geo-lat	0,0924	0,0856	0,0371	0,0202	0,0235	0,0151	0,8540	0,8826	0,8462	0,8448	0,8388	0,8353
greenhouse-net	0,5049	0,2424	0,1615	0,0992	0,0574	0,0324	0,5137	0,5713	0,5747	0,5722	0,5642	0,5618
household-consume	0,0029	0,0051	0,0052	0,0050	0,0039	0,0028	0,0388	0,0639	0,0614	0,0606	0,0592	0,0589
KEGG-reaction	0,0740	0,0732	0,0738	0,0738	0,0731	0,0734	0,0835	0,0857	0,0856	0,0843	0,0841	0,0840
KEGG-relation	0,0761	0,0172	0,0169	0,0151	0,0123	0,0110	0,1809	0,1808	0,1797	0,1757	0,1712	0,1715
online-news	0,8664	0,4151	0,2537	0,1401	0,0766	0,0422	0,9124	0,9880	0,9717	0,9594	0,9536	0,9521
video-transcode	0,0403	0,0334	0,0333	0,0292	0,0222	0,0178	0,0590	0,0697	0,0692	0,0636	0,0585	0,0568
pm25-chengdu-us	0,0191	0,1714	0,1313	0,0789	0,0458	0,0401	0,4125	0,4604	0,4312	0,4179	0,4105	0,4079
park-total-UPDRS	0,2913	0,1013	0,0575	0,0340	0,0312	0,0322	0,8196	0,8360	0,8251	0,8084	0,7908	0,7860
physico-protein	0,0658	0,0360	0,0304	0,0793	0,0525	0,0379	0,5744	0,5543	0,5372	0,5318	0,5291	0,5249
production-quality	0,1636	0,1357	0,1117	0,0782	0,0675	0,0430	0,2794	0,2954	0,2887	0,2830	0,2783	0,2781
CT-slices	0,0054	0,0017	0,0013	0,0014	0,0010	0,0007	0,0693	0,0529	0,0478	0,0423	0,0375	0,0355
gpu-kernel-perf	0,0039	0,0571	0,0060	0,0056	0,0053	0,0036	0,0228	0,1083	0,1230	0,0902	0,0674	0,0500
SML2010	0,0124	0,0225	0,0149	0,0072	0,0077	0,0049	0,1058	0,1517	0,1323	0,1197	0,1060	0,0920
seoul-bike-sharing	0,3527	0,1039	0,0761	0,0917	0,0538	0,0292	0,5171	0,5239	0,5230	0,5199	0,5135	0,5139
UJ-lat	0,0021	0,0019	0,0013	0,0012	0,0011	0,0007	0,0393	0,0527	0,0470	0,0479	0,0452	0,0377
uber-location-price	0,2700	0,2262	0,1756	0,1156	0,0801	0,0469	0,4705	0,4824	0,4792	0,4726	0,4706	0,4688
year-prediction	0,7236	0,3083	0,1892	0,1066	0,0521	0,0304	0,7929	0,8433	0,8262	0,8183	0,8114	0,8131
Average :	0,1951	0,1307	0,0995	0,0750	0,0528	0,0350	0,3636	0,3999	0,3792	0,3730	0,3685	0,3657

B.5 Khiops regressor

B.5.1 Khiops regressor with Random Forest classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,5492	0,2125	0,1411	0,0891	0,0558	0,0357	0,5580	0,2271	0,1599	0,1164	0,0929	0,0827
air-quality-CO	0,3392	0,2768	0,2401	0,1974	0,1481	0,1134	0,3428	0,3211	0,3191	0,3026	0,2889	0,2768
airfoil	0,7518	0,3107	0,2259	0,1578	0,1107	0,0747	0,7709	0,3872	0,3471	0,3176	0,3105	0,2963
appliances-energy	0,8249	0,3985	0,2938	0,1600	0,1424	0,0711	0,8354	0,5759	0,5491	0,5258	0,5224	0,5206
beijing-pm25	0,8060	0,2964	0,1928	0,1156	0,0678	0,0383	0,8082	0,4839	0,4355	0,4113	0,4016	0,3931
temp-forecast-bias	0,4669	0,2797	0,2148	0,1569	0,1122	0,0786	0,4782	0,3704	0,3374	0,3177	0,3006	0,2907
bike-hour	0,4839	0,2427	0,1661	0,1090	0,0668	0,0435	0,4858	0,3241	0,2865	0,2702	0,2587	0,2534
blog-feedback	0,6996	0,4686	0,3976	0,2518	0,2006	0,1802	0,7013	0,6873	0,6730	0,6575	0,6582	0,6601
buzz-twitter	0,2235	0,1962	0,1513	0,1119	0,0711	0,0410	0,2238	0,2235	0,2272	0,2258	0,2222	0,2191
combined-cycle	0,2510	0,2069	0,1679	0,1185	0,0755	0,0489	0,2570	0,2431	0,2315	0,2178	0,2062	0,1995
com-crime	0,5418	0,3006	0,2019	0,1176	0,0649	0,0377	0,5612	0,5924	0,5866	0,5687	0,5610	0,5523
com-crime-unnorm	0,6165	0,3280	0,2224	0,1343	0,0809	0,0475	0,6453	0,6798	0,6599	0,6423	0,6395	0,6340
compress-stren	0,5674	0,3297	0,2540	0,1720	0,1232	0,0922	0,5962	0,4196	0,3783	0,3360	0,3041	0,2959
cond-turbine	0,4294	0,2022	0,1322	0,0598	0,0330	0,0134	0,4481	0,2213	0,1669	0,1261	0,1166	0,1096
cuff-less	0,5430	0,3567	0,3207	0,2904	0,2782	0,2735	0,5450	0,5635	0,5804	0,5845	0,5872	0,5922
electrical-grid-stab	0,5874	0,2615	0,1926	0,1346	0,0899	0,0624	0,5951	0,4226	0,3833	0,3509	0,3369	0,3157
facebook-comment	0,4983	0,3466	0,2298	0,1502	0,0859	0,0472	0,5014	0,4799	0,4551	0,4431	0,4376	0,4347
geo-lat	0,8091	0,3526	0,2551	0,1711	0,1060	0,0906	0,8668	0,9109	0,8612	0,8547	0,8313	0,8408
greenhouse-net	0,6012	0,2666	0,1799	0,1129	0,0680	0,0410	0,6016	0,5675	0,5630	0,5605	0,5593	0,5599
household-consume	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
KEGG-reaction	0,1076	0,0980	0,0882	0,0806	0,0772	0,0756	0,1172	0,1079	0,0991	0,0905	0,0868	0,0855
KEGG-relation	0,3768	0,1057	0,0790	0,0499	0,0372	0,0307	0,4314	0,2123	0,1970	0,1830	0,1771	0,1738
online-news	0,9193	0,4413	0,2722	0,1488	0,0819	0,0456	0,9228	1,1969	1,0396	1,0381	0,9895	0,9942
video-transcode	0,3687	0,2058	0,1311	0,0762	0,0473	0,0288	0,3704	0,2041	0,1344	0,0884	0,0703	0,0627
pm25-chengdu-us	0,7677	0,3457	0,2359	0,1524	0,0945	0,0579	0,7716	0,5071	0,4571	0,4319	0,4180	0,4131
park-total-UPDRS	0,9364	0,3997	0,2529	0,1562	0,0900	0,0490	0,9428	0,8959	0,8587	0,8251	0,8136	0,8117
physico-protein	0,8444	0,2736	0,1753	0,1143	0,0722	0,0469	0,8494	0,5682	0,5505	0,5391	0,5373	0,5330
production-quality	0,3749	0,2590	0,1979	0,1368	0,0882	0,0580	0,3790	0,3332	0,3149	0,2984	0,2877	0,2832
CT-slices	0,0994	0,0611	0,0559	0,0465	0,0352	0,0251	0,1104	0,0705	0,0647	0,0543	0,0462	0,0376
gpu-kernel-perf	0,5924	0,2031	0,1256	0,0820	0,0473	0,0299	0,5927	0,2543	0,1608	0,1027	0,0737	0,062
SML2010	0,3134	0,2164	0,1660	0,1163	0,0845	0,0616	0,3473	0,2524	0,2088	0,1652	0,1369	0,1221
seoul-bike-sharing	0,5812	0,2490	0,1727	0,1091	0,0691	0,0411	0,5891	0,5475	0,5323	0,5250	0,5243	0,5169
UJ-lat	0,0450	0,0346	0,0272	0,0248	0,0194	0,0152	0,0527	0,0437	0,0407	0,0405	0,0407	0,0384
uber-location-price	0,8410	0,3030	0,2098	0,1390	0,0941	0,0657	0,8419	0,5048	0,4901	0,4833	0,4869	0,4879
year-prediction	0,8808	0,3492	0,2130	0,1171	0,0575	0,0341	0,8822	0,8424	0,8366	0,8311	0,8234	0,8138
Average :	0,5482	0,2700	0,1936	0,1283	0,0875	0,0617	0,5595	0,4483	0,4172	0,3978	0,3867	0,3813

B.5.2 Khiops regressor with Khiops classifier

Dataset name	Base Train RMSE	S = 2 Train RMSE	S = 4 Train RMSE	S = 8 Train RMSE	S = 16 Train RMSE	S = 32 Train RMSE	Base Test RMSE	S = 2 Test RMSE	S = 4 Test RMSE	S = 8 Test RMSE	S = 16 Test RMSE	S = 32 Test RMSE
3Droad	0,5520	0,3270	0,2892	0,2635	0,2366	0,2153	0,5594	0,3432	0,3081	0,2863	0,2626	0,2438
air-quality-CO	0,3379	0,3313	0,3292	0,3206	0,3141	0,3071	0,3432	0,3376	0,3385	0,3300	0,3235	0,3165
airfoil	0,7540	0,5748	0,5400	0,5043	0,4657	0,4517	0,7690	0,6056	0,5621	0,5421	0,5067	0,4929
appliances-energy	0,8196	0,6906	0,6734	0,6593	0,6619	0,6593	0,8332	0,7205	0,7059	0,6943	0,6944	0,6932
beijing-pm25	0,8066	0,5901	0,5901	0,5666	0,5603	0,5469	0,8091	0,6072	0,6058	0,5836	0,5791	0,5642
temp-forecast-bias	0,4629	0,4179	0,4117	0,4060	0,3973	0,3915	0,4750	0,4427	0,4348	0,4303	0,4238	0,4183
bike-hour	0,4844	0,3587	0,3289	0,3133	0,3071	0,2931	0,4865	0,3672	0,3375	0,3210	0,3179	0,3052
blog-feedback	0,6987	0,6673	0,6675	0,6604	0,6579	0,6554	0,7007	0,6731	0,6722	0,6668	0,6650	0,6624
buzz-twitter	0,2236	0,2237	0,2236	0,2232	n/a	0,2230	0,2239	0,2241	0,2239	0,2236	n/a	0,2236
combined-cycle	0,2527	0,2468	0,2422	0,2403	0,2379	0,2359	0,2579	0,2525	0,2478	0,2476	0,2450	0,2414
com-crime	0,5457	0,5368	0,5336	0,5314	0,5284	0,5276	0,5686	0,5672	0,5592	0,5604	0,5594	0,5654
com-crime-unnorm	0,6216	0,6144	0,6079	0,6094	0,6074	0,6065	0,6454	0,6436	0,6372	0,6364	0,6373	0,6426
compress-stren	0,5508	0,4027	0,3860	0,3722	0,3572	0,3557	0,5804	0,4401	0,4345	0,4109	0,4063	0,4148
cond-turbine	0,4185	n/a	0,2307	0,2055	0,1893	0,1700	0,4379	n/a	0,2573	0,2320	0,2204	0,2038
cuff-less	0,5428	0,5095	0,5050	0,5026	0,5012	0,5004	0,5447	0,5139	0,5103	0,5081	0,5074	0,5062
electrical-grid-stab	0,5851	0,3919	0,3744	0,3607	0,3466	0,3474	0,5941	0,4162	0,3964	0,3845	0,3679	0,3697
facebook-comment	0,5001	0,4667	0,4542	0,4412	0,4366	0,4357	0,5037	0,4736	0,4631	0,4495	0,4458	0,4445
geo-lat	0,8023	0,7849	0,7799	0,7844	0,7611	0,7750	0,8661	0,8682	0,8615	0,8827	0,8386	0,8640
greenhouse-net	0,6016	0,5787	0,5771	0,5781	0,5791	0,5756	0,6019	0,5791	0,5775	0,5784	0,5795	0,5762
household-consume	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
KEGG-reaction	0,1072	0,0964	0,0931	0,0908	0,0889	0,0884	0,1173	0,1056	0,1004	0,0960	0,0929	0,0919
KEGG-relation	0,3788	0,2750	0,2609	0,2467	0,2363	0,2315	0,4360	0,3442	0,3330	0,3189	0,3089	0,3009
online-news	0,9188	0,9136	0,9132	0,9131	0,9128	0,9126	0,9236	0,9200	0,9188	0,9193	0,9190	0,9185
video-transcode	0,3684	0,2203	0,1811	0,1566	0,1403	0,1279	0,3704	0,2240	0,1857	0,1628	0,1467	0,1352
pm25-chengdu-us	0,7681	0,6611	0,6610	0,6592	0,6578	0,6400	0,7720	0,6734	0,6738	0,6703	0,6700	0,6540
park-total-UPDRS	0,9330	0,8807	0,8676	0,8537	0,8373	0,8302	0,9419	0,9022	0,8932	0,8856	0,8701	0,8750
physico-protein	0,8441	0,6585	0,6416	0,6362	0,6355	0,6352	0,8493	0,6827	0,6689	0,6658	0,6681	0,6729
production-quality	0,3729	0,3235	0,3147	0,3061	0,3013	0,2985	0,3775	0,3323	0,3236	0,3162	0,3124	0,3081
CT-slices	0,1001	0,0756	0,0706	0,0615	0,0584	0,0560	0,1104	0,0825	0,0798	0,0721	0,0670	0,0665
gpu-kernel-perf	0,5939	0,1929	0,1225	0,1097	0,0941	0,0801	0,5942	0,1940	0,1240	0,1116	0,0966	0,0830
SML2010	0,3198	0,2427	0,2066	0,1816	0,1669	0,1580	0,3474	0,2693	0,2333	0,2100	0,1958	0,1887
seoul-bike-sharing	0,5847	0,5332	0,5240	0,5175	0,5126	0,5112	0,5907	0,5460	0,5375	0,5355	0,5346	0,5335
UJ-lat	0,0460	0,0362	0,0312	0,0290	0,0262	0,0251	0,0533	0,0441	0,0397	0,0353	0,0320	0,0316
uber-location-price	0,8413	0,4952	0,4840	0,4777	0,4735	0,4721	0,8422	0,5031	0,4915	0,4853	0,4816	0,4799
year-prediction	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Average :	0,5375	0,4475	0,4278	0,4176	0,4152	0,4042	0,5493	0,4656	0,4466	0,4380	0,43676	0,4269

C Hyperparameters influence on performance for Random Forest regressor

Dataset name	Train RMSE	Test RMSE	Optimized Train RMSE	Optimized Test RMSE
3Droad	0,0282	0,0748	n/a	n/a
air-quality-CO	0,0983	0,2621	0,1079	0,2667
airfoil	0,0969	0,2615	0,1171	0,2987
appliances-energy	0,1924	0,5115	0,2041	0,5456
beijing-pm25	0,1418	0,3776	0,1577	0,4168
temp-forecast-bias	0,1172	0,3119	0,1252	0,3237
bike-hour	0,0867	0,2291	0,0964	0,2415
blog-feedback	0,2615	0,6373	0,6074	0,6481
buzz-twitter	0,0818	0,2182	n/a	n/a
combined-cycle	0,0759	0,2021	0,0841	0,2079
com-crime	0,2076	0,5535	0,2572	0,5525
com-crime-unnorm	0,2280	0,6089	0,3214	0,6163
compress-stren	0,1171	0,2861	0,1327	0,3137
cond-turbine	0,0345	0,0896	0,0419	0,1117
cuff-less	0,2761	0,5634	0,488	0,5025
electrical-grid-stab	0,1200	0,3184	0,1269	0,3352
facebook-comment	0,1555	0,3967	0,1897	0,4162
geo-lat	0,3011	0,8113	0,3275	0,8321
greenhouse-net	0,1982	0,5300	n/a	n/a
household-consume	0,0161	0,0426	n/a	n/a
KEGG-reaction	0,0742	0,0832	0,0729	0,0837
KEGG-relation	0,0627	0,1626	0,0681	0,1782
online-news	0,3442	0,9202	0,5876	0,9177
video-transcode	0,0206	0,0544	0,0229	0,0604
pm25-chengdu-us	0,1381	0,3652	0,1549	0,4111
park-total-UPDRS	0,2928	0,7848	0,3045	0,802
physico-protein	0,2010	0,5332	0,2096	0,5528
production-quality	0,1047	0,2786	0,1093	0,2836
CT-slices	0,0219	0,0573	n/a	n/a
gpu-kernel-perf	0,0068	0,0180	n/a	n/a
SML2010	0,0386	0,1000	0,0468	0,1241
seoul-bike-sharing	0,1866	0,4990	0,207	0,5101
UJ-lat	0,0126	0,0326	n/a	n/a
uber-location-price	0,1744	0,4636	n/a	n/a
year-prediction	0,3006	0,8025	n/a	n/a

Table 8: Difference of the performance before and after the optimization of the hyper-parameters of the Random Forest regressor on the original datasets (not extended using classifiers). The results are the mean of a 10 fold cross-validation.

D Wilcoxon signed rank test & Box plots

Please refer to the figure 17 for the box plot of the Linear Regression model.

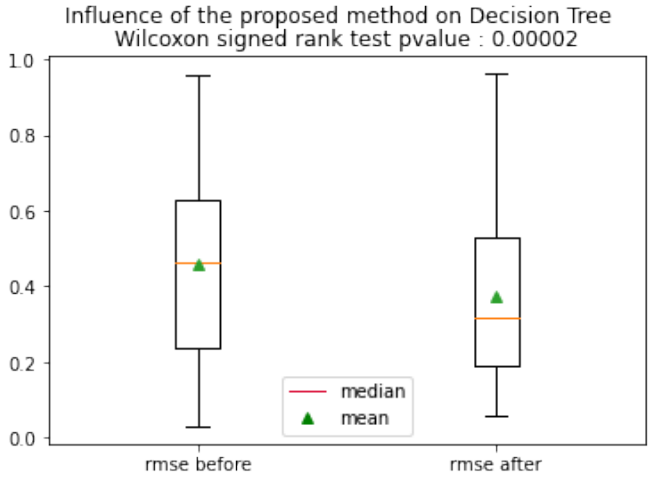


Figure 26: Box plot of the RMSE of the **Decision Tree regressor** on the 35 datasets without (label 'rmse before') and with (label 'rmse after') our method for 32 thresholds to extract features with the Random Forest classifier.

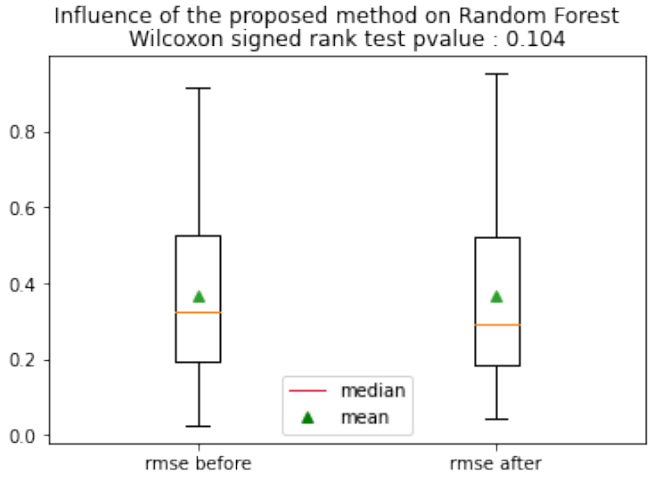


Figure 27: Box plot of the RMSE of the **Random Forest regressor** on the 35 datasets without (label 'rmse before') and with (label 'rmse after') our method for 32 thresholds to extract features with the Random Forest classifier.

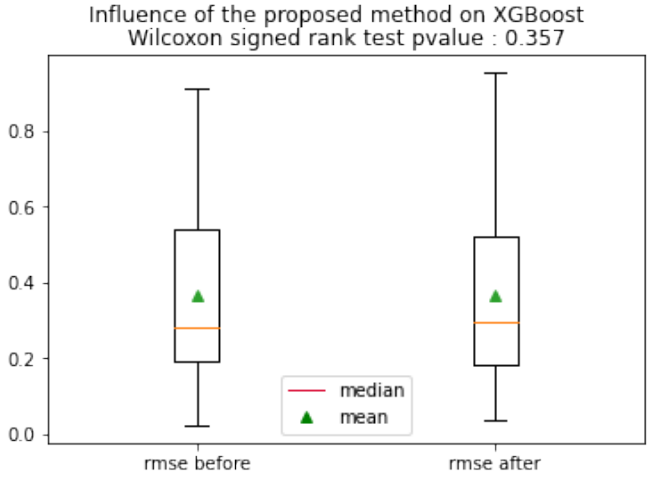


Figure 28: Box plot of the RMSE of the **XGBoost regressor** on the 35 datasets without (label 'rmse before') and with (label 'rmse after') our method for 32 thresholds to extract features with the Random Forest classifier.

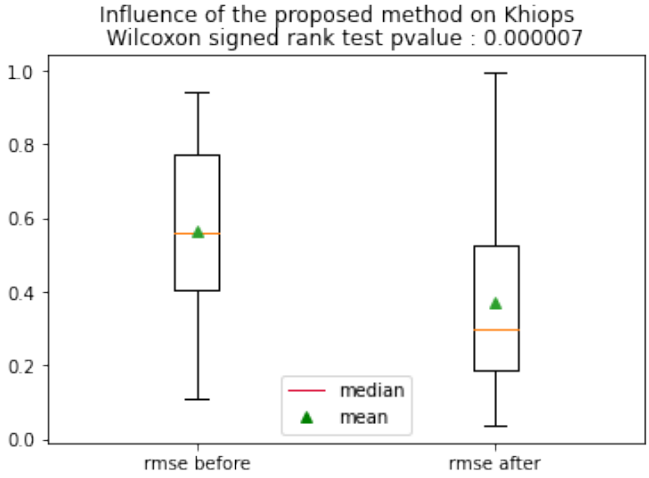


Figure 29: Box plot of the RMSE of the **Khiops regressor** on the 35 datasets without (label 'rmse before') and with (label 'rmse after') our method for 32 thresholds to extract features with the Random Forest classifier.

E Performance impact of collinear extracted features

	Linear Regression		Decision Tree		Random Forest		Khiops	
Dataset name	RMSE	New RMSE	RMSE	New RMSE	RMSE	New RMSE	RMSE	New RMSE
air-quality-CO	0,2680	0,2678	0,2712	0,2712	0,2639	0,2631	0,3148	0,3189
airfoil	0,2373	0,2372	0,3159	0,3187	0,2871	0,2881	0,4916	0,4930
appliances-energy	0,4871	0,4868	0,5269	0,5270	0,5165	0,5160	0,6938	0,6939
beijing-pm25	0,4495	0,4177	0,3949	0,3946	0,3876	0,3873	0,5641	0,5640
temp-forecast-bias	0,2841	0,2843	0,2863	0,2853	0,2774	0,2774	0,4181	0,4191
bike-hour	0,2554	0,2548	0,2561	0,2565	0,2498	0,2497	0,3037	0,3067
blog-feedback	0,6438	0,6437	0,6701	0,6690	0,6619	0,6626	0,6631	0,6623
combined-cycle	0,1964	0,1959	0,2011	0,2014	0,1954	0,1953	0,2398	0,2421
com-crime	0,5512	0,5509	0,5556	0,5564	0,5504	0,5507	0,5640	0,5624
com-crime-unnorm	0,6034	0,6032	0,6264	0,6256	0,6175	0,6172	0,6419	0,6430
compress-stren	0,2764	0,2784	0,3181	0,3172	0,2901	0,2901	0,4114	0,4129
cond-turbine	0,1057	0,1076	0,1112	0,1096	0,1061	0,1062	0,2086	0,2071
cuff-less	0,5798	0,5772	0,6251	0,6253	0,6105	0,6143	0,5065	0,5062
electrical-grid-stab	0,3142	0,3142	0,3256	0,3249	0,3086	0,3081	0,3694	0,3698
facebook-comment	0,4246	0,4177	0,4430	0,4431	0,4389	0,4389	0,4449	0,4448
geo-lat	0,8093	0,8113	0,8532	0,8461	0,8351	0,8407	0,8620	0,8611
KEGG-reaction	0,0847	0,0844	0,0862	0,0867	0,0835	0,0836	0,0901	0,0920
KEGG-relation	0,1772	0,1733	0,1748	0,1740	0,1700	0,1697	0,2758	0,2813
online-news	0,9955	1,1312	0,9622	0,9623	0,9508	0,9519	0,9187	0,9184
video-transcode	0,0559	0,0564	0,0652	0,0652	0,0598	0,0596	0,1352	0,1498
Average:	0,4094	0,4150	0,4128	0,4123	0,4023	0,4026	0,4840	0,4855
Average difference:	-0,0056		0,0005		-0,0003		-0,0015	

Table 9: Effect of the presence of collinear extracted features for the Linear Regression, Decision Tree, Random Forest and Khiops regressors. The columns *Test RMSE* displays the test performance of the regressors when the datasets were extended using a Random Forest classifier with 32 thresholds with our method, and the features $P(C_0|X)$ and $P(C_1|X)$ were added. The columns *New Test RMSE* show the performance when only the feature $P(C_0|X)$ was used.

F Models parameters

Model name	Parameters	
Linear Regression	fit_intercept = True normalize = False	copy_X = True positive = False
Logistic Regression	penalty = l2 dual = False tol = 1e-4 C = 1.0 fit_intercept = True intercept_scaling = 1 class_weight = None random_state = None	solver = lbfgs max_iter = 100 multi_class = auto verbose = 0 warm_start = False n_jobs = -1 l1_ratio = None
RandomForest Regressor	n_estimators = 100 criterion = mse max_depth = None min_samples_split = 2 min_samples_leaf = 1 min_weight_fraction_leaf = 0.0 max_features = auto (n_feat.) max_leaf_nodes = None min_impurity_decrease = 0.0	min_impurity_split = None bootstrap = True oob_score = False random_state = None verbose = 0 warm_start = False ccp_alpha = 0.0 max_samples = None base_estimator_ = DecisionTree
RandomForest Classifier	n_estimators = 100 criterion = gini max_depth = None min_samples_split = 2 min_samples_leaf = 1 min_weight_fraction_leaf = 0.0 max_features = sqrt(n_feat.) max_leaf_nodes = None min_impurity_decrease = 0.0 min_impurity_split = None	bootstrap = True oob_score = False n_jobs = -1 random_state = None verbose = 0 warm_start = False class_weight = None ccp_alpha = 0.0 max_samples = None base_estimator_ = DecisionTreer
XGBoost Regressor & XGBoost Classifier	n_estimators = 100 max_depth = 6 learning_rate = 0.3 verbosity = 1 objective = "reg:squarederror" for regressor and "binary:logistic" for classifier booster = gbtree tree_method = auto n_jobs = -1 gamma = 0 min_child_weight = 1 max_delta_step = 0 subsample = 1	colsample_bytree = 1 colsample_bylevel = 1 colsample_bynode = 1 reg_alpha = 0 reg_lambda = 1 scale_pos_weight = 1 base_score = 0.5 random_state = None missing = np.nan num_parallel_tree = 1 monotone_constraints = ? interaction_constraints = ? importance_type = "gain" validate_parameters = False
Decision Tree Regressor & Decsision Tree Classifier	criterion = "gini" splitter = "best" max_depth = None min_samples_split = 2 min_samples_leaf = 1 min_weight_fraction_leaf = 0.0 max_features = None	random_state = None max_leaf_nodes = None min_impurity_decrease = 0.0 min_impurity_split = 0 class_weight = None (decision tree classifier only) ccp_alpha = 0.0

Table 10: Parameters used for the models used in the project. The version of scikit-learn is 0.24.2 and the version of xgboost is 1.4.1.

model	**parameters**	
XGBoost	**max_depth**	**learning_rate**
	[4, 8, 16, 32]	[0.01, 0.1, 0.3]
Random Forest	**max_depth**	**max_features**
	[4, 8, 16, 32]	4 values between 2 and the number of features, plus the square root of the number of features
Decision Tree	**max_depth**	**max_features**
	[4, 8, 16, 32]	4 values between 2 and the number of features, plus the square root of the number of features

Table 11: The parameters explored for each regressor

References

[1] Amir Ahmad, S. M. Halawani, and I. Albidewi. Novel ensemble methods for regression via classification problems. *Expert Syst. Appl.*, 39:6396–6401, 2012.

[2] Amir Ahmad, Shehroz S Khan, and Ajay Kumar. Learning regression problems by using classifiers. *Journal of Intelligent & Fuzzy Systems*, 35(1):945–955, 2018.

[3] A. Atkinson. The box-cox transformation: review and extensions. *LSE Research Online Documents on Economics*, 2020.

[4] Alexei Botchkarev. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14:45–79, 01 2019.

[5] M. Boullé. MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165, 2006.

[6] M. Boullé. Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685, 2007.

[7] L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[8] J. Catlett. On changing continuous attributes into ordered discrete attributes. In Yves Kodratoff, editor, *Machine Learning — EWSL-91*, pages 164–178, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *the 22nd ACM SIGKDD International Conference*, pages 785–794, 08 2016.

[10] Antoine Cornuéjols and Laurent Miclet. *Apprentissage artificiel : Concepts et algorithmes*. Eyrolles, June 2010.

[11] Ahiale Darlington. Car evaluation data set. Accessed: 2021-10-05.

[12] Carsten F. Dormann, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R. García Marquéz, Bernd Gruber, Bruno Lafourcade, Pedro J. Leitão, Tamara Münkemüller, Colin McClean, Patrick E. Osborne, Björn Reineking, Boris Schröder, Andrew K. Skidmore, Damaris Zurell, and Sven Lautenbach. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 36(1):27–46, 2013.

[13] James Dougherty, Ron Kohavi, and Mehran Sahami. *Supervised and Unsupervised Discretization of Continuous Features*. Morgan Kaufmann, 1995.

[14] M. Fernández-Delgado, M.S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande. An extensive experimental survey of regression methods. *Neural Networks*, 111:11–34, 2019.

[15] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[16] Carine Hue and Marc Boullé. A new probabilistic approach in rank regression with optimal bayesian partitioning. *Journal of Machine Learning Research*, 8(90):2727–2754, 2007.

[17] Frederik Janssen and Johannes Fürnkranz. Separate-and-conquer regression. Technical Report TUD-KE-2010-01, TU Darmstadt, Knowledge Engineering Group, 2010.

[18] Frederik Janssen and Johannes Fürnkranz. Heuristic rule-based regression via dynamic reduction to classification. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 1330–1335, 2011.

[19] Kaggle. Categorical feature encoding challenge. Accessed: 2021-10-05.

[20] Roger Koenker. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

[21] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, 01 2013.

[22] Max Kuhn and Kjell Johnson. *Feature Engineering and Selection*. CRC Press, 07 2019.

[23] UCI Machine Learning. Adult census income. Accessed: 2021-10-05.

[24] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:14 – 23, 01 2011.

[25] S. A. Memon, W. Zhao, B. Raj, and R. Singh. Neural regression trees. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[26] Florian Pargent. A benchmark experiment on how to encode categorical features in predictive modeling. Master's thesis, Ludwig-Maximilians-Universität München, 03 2019.

[27] Foster Provost. Well-trained pets: Improving probability estimation trees, 11 2000.

[28] R. Rodriguez. Five things you should know about quantile regression. In *SAS Global Forum*, 2017. accessed april 2021.

[29] Damien Soukhavong. Visiting: Categorical features and encoding in decision trees. Accessed: 2021-30-04.

[30] Luís Torgo and João Gama. Regression using classification algorithms. *Intelligent Data Analysis*, 1(1):275–292, 1997.

[31] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, 2018.

Feature engineering for regression using classification

Résumé :

Ce rapport propose une nouvelle méthode pour améliorer les performances de régression de tout modèle en utilisant des classifieurs. Pour utiliser des modèles de classification sur un problème de régression, cette méthode s'appuie sur un processus de discrétisation et d'encodage de classes. Une étude des modèles de régression de l'état de l'art qui ont été utilisés dans ce rapport a été réalisée, ainsi que les multiples métriques utilisées pour évaluer les performances de la régression. Une approche scientifique a été utilisée pour mesurer l'impact sur les performances de cette méthode.

Mots-clés :

Régression, classification, apprentissage machine, protocole experimental, scikit-learn, RMSE, discrétisation.

Abstract :

This report provides a new method to improve the regression performance of any model using classifiers. To use classification models on a regression problem, this method is based on a process of discretization and encoding of classes. A study of the advanced regression models that were used in this report was performed, as well as the multiple metrics used to assess the performance of the regression. A scientific approach was used to measure the impact on the performance of this method.

Keywords :

Regression, classification, machine learning, experimental protocol, scikit-learn, RMSE, discretization.