WORKING PAPER 2/2024 (ECONOMICS AND STATISTICS)

Forecast model of the price of a product with a cold start

Svitlana Drin

ISSN 1403-0586

Örebro University School of Business
SE-701 82 Örebro, Sweden

Forecast model of the price of a product with a cold start

Svitlana Drin[1,2]

[1]School of Business, Örebro University, 70182 Örebro, Sweden
[2]Department of Mathematics, National University of Kyiv-Mohyla Academy, 04070 Kyiv, Ukraine

January 17, 2024

Abstract

This article presents a comprehensive study on developing a predictive product pricing model using LightGBM, a machine learning method optimized for regression challenges in situations with limited historical data. It begins by detailing the core principles of LightGBM, including decision trees, boosting, and gradient descent, and then delves into the method's unique features like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). The model's efficacy is demonstrated through a comparative analysis with XGBoost, highlighting LightGBM's enhanced efficiency and slight improvement in prediction accuracy. This research offers valuable insights into the application of LightGBM in developing fast and accurate product pricing models, crucial for businesses in the rapidly evolving data landscape.

Keywords: GBM, GBDT, LightGBM, GOSS, EFB, predictive model.

1 Introduction

In the current fast-changing business environment, companies face significant challenges in managing their sales strategies and determining the best pricing strategies for their products. One of the most complex aspects of this task is setting the price for a new product that lacks historical sales data.

Developing precise and dependable pricing strategies for new products has given rise to advanced predictive models. These models are essential for companies seeking to determine the best prices without prior experience. Of these models, the LightGBM (Light Gradient Boosting Machine) algorithm is particularly effective in rapidly identifying the optimal forecast. This article employs the LightGBM approach to construct a predictive pricing model for a novel product, designed to address the intricate obstacles of contemporary markets.

The focus of this study is to incorporate the LightGBM algorithm into a pricing model for predicting optimal prices for new products.

The paper is structured as follows. The second section provides an overview of the theoretical framework underlying the chosen machine learning model for price prediction. This section is based on the works of Friedman (2001), Hastie et al. (2009), Chen and Guestrin (2016), Ke et al. (2017) and explains the essential concepts. Friedman's article Friedman (2001) provides a detailed explanation of a general gradient-descent "boosting" paradigm. Specific algorithms are presented for regression using least-squares, least-absolute-deviation, and Huber-M loss functions, and for classification using multi-class logistic likelihood. Elements of Statistical Learning Hastie et al. (2009) by Hastie, Tibshirani, and Friedman introduces decision trees, bagging, boosting, random forest, gradient descent, and other useful concepts and techniques. In Chen and Guestrin (2016), algorithms for identifying split points are explained.

The third section describes the primary model, highlighting its distinctive features and efficiency. The primary concept of LightGBM is found in the original developer documentation Ke et al. (2017). The source provides a comprehensive explanation of newly introduced algorithms, GOSS, and EFB, along with insights into their motivations and unique characteristics.

The fourth section presents some numerical results for data, including root mean squared errors (RMSE), mean absolute error (MAE), and R-squared. The benefits of these results are explained in Chicco et al. (2021). The model is evaluated using training data, and a comparative assessment is performed, comparing the model's performance against that of the XGBoost model.

2 Preliminary theoretical base

2.1 Decision Trees

Decision Trees, a basic machine learning model, offer a structured approach to decision-making through a tree-like framework. One of the notable strengths of Decision Trees lies in their interpretability, enabling the identification of influential features that influence decision-making processes. The hierarchical structure of Decision Trees encompasses a root node, branches, internal nodes, and leaf nodes.

A crucial concept of comprehending Decision Tree algorithms is the notion of "impurity." There are different measures of impurity such as entropy and the Gini index. Impurity measures the impurity of a decision node in the tree. It aids in determining which attributes are best suited for dividing into two ranges for regression. The concept of information entropy was introduced by Claude Shannon in 1948 Shannon (1948).

Entropy is computed using the formula:

$$Entropy(S) = -\sum_{c \in C} p(c) \log_2(p(c))$$

where S represents the dataset under consideration, c denotes classes within dataset S, $p(c)$ signifies the fraction of data points belonging to class C relative to the total data points in dataset S. The entropy values range from 0 to 1. An entropy of 0 indicates that all instances in the dataset belong to a single class, while an entropy of 1 indicates maximum diversity.

2.2 Boosting

Boosting is a technique involving the sequential training of classifiers in an ensemble. Unlike Bagging, Boosting assigns greater attention to the mistakes of previous models.

While Bagging trains base learners on independently bootstrapped data subsets, allowing us to simultaneously train all base learners in a parallel environment, Boosting sequentially trains base learners-models are trained one after another. Therefore, training base learners in parallel is not possible in Boosting.

How Boosting Works:

1. Assign weights to each training example so that the sum of weights equals 1. Initially, all example weights are equal.
2. Train the first classifier and identify the examples on which it made mistakes.
3. Reallocate weights so that "error examples" from the previous step have greater weight (while the sum of weights remains 1).
4. Train the next classifier. Since classification quality is evaluated as a weighted sum of errors, the second classifier focuses on "smoothing out" the mistakes of the first classifier.
5. Repeat the process until all classifiers are trained.

Mathematically Friedman (2001), Schapire and Freund (2013), we have:

$$h(x) = \sum_{j=1}^{m} \rho_j h_j(x)$$

where ρ represents the weights of the j-th classifier.

By iteratively adjusting the weights of training examples and training weak models to correct the errors of the previous ones, Boosting creates a strong model capable of accurate data classification.

2.3 Gradient Descent

The gradient descent method is based on the idea that if the function of multiple variables $F(x)$ is defined and differentiable in the vicinity of point a, then $F(x)$ decreases fastest by moving from a in the direction of the negative gradient of F at a, denoted as $-\nabla F(a)$ (where ∇ represents the gradient, a vector of partial derivatives of the function). This implies that if $a_{n+1} = a_n - \gamma \nabla F(a_n)$ for a sufficiently small step size (or learning rate) $\gamma \in R_+$, then $F(a_n) \geq F(a_{n+1})$.

In other words, $\gamma \nabla F(a)$ is subtracted from a because we aim to move against the gradient towards a local minimum. With this understanding, we initiate with x_0 as an assumption for a local minimum of F, considering the sequence $x_0, x_1, x_2, \ldots$ such that $x_{n+1} = x_n - \gamma_n \nabla F(x_n)$, for $n \geq 0$.

As a result, we obtain $F(x_0) \geq F(x_1) \geq F(x_2) \geq \ldots$, and we expect the sequence x_n to converge towards a local minimum. It's worth noting that the step size γ can be adjusted at each iteration.

2.4 Gradient Boosting Machine

Gradient Boosting Machine, commonly referred to as GBM, is a machine learning method utilized for solving classification and regression tasks. It constructs a predictive model by combining multiple weak predictive models. GBM builds the model iteratively, similar to other boosting methods, but it is more versatile as it enables the optimization of any differentiable loss function.

GBM is typically used in conjunction with decision trees as base models, hence this combination is often referred to as Gradient Boosting Decision Trees (GBDT). Thus, we can assert that GBM is a variant of an ensemble method, while GBDT is a specific case where a tree is used as the estimator. Friedman (2001)

The generic gradient tree-boosting algorithm for regression Hastie et al. (2009).

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L\left(y_i, \gamma\right)$
2. For $m = 1$ to M :

a For $i = 1, 2, \ldots, N$ compute

$$r_{im} = -\left[\frac{\partial L\left(y_i, f\left(x_i\right)\right)}{\partial f\left(x_i\right)}\right]_{f=f_{m-1}}$$

b Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm}, j = 1, 2, \ldots, J_m$

c For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}\left(x_i\right) + \gamma\right)$$

d Update

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I\left(x \in R_{jm}\right)$$

3. Output $\hat{f}(x) = f_M(x)$.

At each iteration, GBDT learns a decision tree by fitting residual errors (errors up to the current iteration). This means that each subsequent learner aims to learn the difference between the actual outcome and the weighted sum of predictions from the previous iteration. Errors are minimized using the gradient method. The gradient indicates the steepest descent direction of the loss function, which GBDT employs to search for optimal splitting points to construct the tree.

Another popular algorithm is the Histogram-based algorithm. Instead of searching for split points among sorted feature values, the Histogram-based algorithm divides continuous feature values into discrete bins and utilizes these bins to construct feature histograms during training.Ke et al. (2017)

2.5 Histogram based algorithm

The fundamental idea of the Histogram-based algorithm is to discretize the sequential feature values into k integers and construct a histogram with a width of k. While traversing the data, the discretized value acts as an index for accumulating statistics in the histogram. After a single pass through the data, the histogram accumulates the necessary statistics and is subsequently traversed again to find the optimal split point. Since the histogram-based algorithm stores discrete bins rather than continuous feature values, a feature bundle can be built by allowing mutually exclusive features to occupy a specific range of bins. This can be achieved by increasing the shift of the initial feature value.

The histogram-based algorithm from Guolin et al. (2017) presented in Ke et al. (2017) as Algorithm 1: Histogram-based Algorithm.

As shown the histogram-based algorithm finds the best split points based on feature histograms. It costs $O(\#data \times \#feature)$ for histogram construction and $O(\#bin \times \#feature)$ for finding split points. Since $\#bin$ is typically much smaller than $\#data$, histogram construction will dominate the computational complexity. If we can reduce $\#data$ or $\#feature$, we can significantly accelerate GBDT training.

3 LightGBM

LightGBM is a type of Gradient Boosting Decision Trees (GBDT) that was developed by a team of researchers at Microsoft in 2016. This model was created to improve upon the popular XGBoost model, which is known for its speed and reliability in multi-class classification projects.

The reason for enhancing XGBoost was to achieve even greater efficiency and faster implementation. The most computationally intensive task in GBDT is the search for optimal split points, which is directly proportional to both the number of features and the number of instances. This leads to speed-related issues when dealing with large datasets. To address this, two new techniques were introduced: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). These techniques aimed to reduce the number of data instances and functions, thereby mitigating the computational challenges associated with GBDT training on large datasets.

3.1 Gradient-based One-Side Sampling (GOSS)

In GBDT, there is no individual weight for each data instance, but LightGBM that instances with different gradients have varying impacts on information gain calculations. Specifically, instances with higher gradients (less trained samples) exert a greater influence on calculating information gain.

To balance the effect of data distribution, GOSS introduces a constant multiplier for examples with smaller gradients, as presented in Algorithm 2: Gradient-based One-Side Sampling Ke et al. (2017), compensating for their contribution to the distribution. Initially, the algorithm sorts data by the absolute values of their gradients and selects the top $a \times 100\%$ of them. Then, a random selection of $b \times 100\%$ instances is made from the remaining data. During information gain computation, GOSS amplifies the selected data with lower gradients by a constant factor of $\frac{1-a}{b}$, paying more attention to less trained instances, without altering the original data distribution.

More theoretically, GBDT utilizes decision trees to learn functions from the input space χ^s to the gradient space G. Assuming a training dataset of n instances $\{x_1, x_2, \ldots, x_n\}$, where each x_i is a vector of dimension s in χ^s. During each gradient boosting iteration, we compute negative gradients of the loss function with respect to the model's predictions, denoted as $\{g_1, g_2, \ldots, g_n\}$.

For constructing decision tree models, each node is split based on the most informative feature. In the context of GBDT, information gain is typically quantified through post-split variance reduction, defined as follows.

Definition Let O be the training dataset on a fixed node of the decision tree. The variance gain of splitting feature j at point d for this node is defined as

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in O: x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in O: x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right),$$

where $n_O = \sum I[x_i \in O]$, $n^j_{l|O} = \sum I[x_i \in O : x_{ij} \le d]$, and $n^j_{r|O} = \sum I[x_i \in O : x_{ij} > d]$.

In GBDT, the decision tree algorithm chooses $d^*_j = \text{argmax}_d V_j(d)$ for feature j and computes the maximum gain $V_j(d^*_j)$. The data is then split based on feature j^* at point d^*_j into left and right child nodes.

In the novel GOSS algorithm a subset A is first formed by selecting the top $a \times 100\%$ of instances with higher gradients. A random subset B is then sampled from the remaining instances with lower gradients. The instances from subsets $A \cup B$ are split based on the estimated variance reduction $\widetilde{V}_j(d)$.

For a subset A_l, A_r, B_l, and B_r defined as described, the estimated variance reduction is given by:

$$\widetilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n^j_l(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n^j_r(d)}\right),$$

where a and b are constants, and the coefficient $\frac{1-a}{b}$ is used for normalization.

Additionally, the GOSS method is supported by the following theorem:

Theorem Let $\mathcal{E}(d) = \left|\widetilde{V}_j(d) - V_j(d)\right|$ represent the approximation error in GOSS. With probability at least $1 - \delta$, we have:

$$\mathcal{E}(d) \le C^2_{a,b} \ln\left(\frac{1}{\delta}\right) \cdot \max\left\{\frac{1}{n^j_l(d)}, \frac{1}{n^j_r(d)}\right\} + 2DC_{a,b}\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{n}},$$

where $C_{a,b} = \frac{1-a}{\sqrt{b}} \cdot \max_{x_i \in A^c} |g_i|$ and $D = \max\{\bar{g}^j_l(d), \bar{g}^j_r(d)\}$.

The theorem provides an upper bound on the approximation error $\mathcal{E}(d)$, which can be controlled. With a probability of at least $1 - \delta$, this error can be bounded by a value that depends on the size of the data subset and the maximum gradient value.

3.2 Exclusive Feature Bundling (EFB)

High-dimensional data often contain numerous features, which can lead to model overfitting. Sparsity in feature space is a common phenomenon.

Sparsity implies that many features are mutually exclusive, meaning they do not have non-zero values simultaneously. This allows us to group them into *"exclusive feature bundles"*. A scanning algorithm enables the construction of histograms for these bundles, instead of individual features, reducing the histogram construction complexity from $O(\#data \times \#feature)$ to $O(\#data \times \#bundle)$, where $\#bundle$ is significantly smaller than $\#feature$.

This facilitates accelerating GBDT training while preserving model accuracy. However, two issues arise: the first involves selecting features to be grouped into a bundle, and the second pertains to creating the bundle itself.

Since finding the optimal grouping strategy is an NP-hard problem, we can approximate it by reducing it to graph coloring, where nodes represent objects and edges indicate which objects can be grouped. A greedy algorithm can provide fairly accurate results for graph coloring with a constant approximation factor.

Algorithm 3: Greedy Bundling from Guolin et al. (2017) presented in Ke et al. (2017)

Randomly introducing noise to a fraction of feature values has a limited impact on training, as long as the maximum conflict frequency within each bundle is γ. The training accuracy will not decrease more than $O([(1-\gamma)n]^{\frac{-2}{3}})$, where γ is the total number of features.

Thus, opting for a small γ value maintains a balance between accuracy and efficiency. Based on this, we have an algorithm for exclusive feature bundling.

The EFB algorithm, which is introduced by Algorithm 4: Merge Exclusive Features in Ke et al. (2017), can consolidate numerous exclusive features into a significantly smaller set of dense features, enabling efficient avoidance of unnecessary computations for zero feature values.

The LightGBM algorithm

Input:

Training data:
D $= \{(\chi 1, y1), (\chi 2, y2), \ldots, (\chi \mathrm{N}, y\mathrm{N})\}$, χi $\in \chi$, $\chi \subseteq$ R, yi $\in \{-1, +1\}$;
loss function: $L(y, \theta(\chi))$;

Iterations: M;
Big gradient data sampling ratio: a;
slight gradient data sampling ratio: b;

1. Combine features that are mutually exclusive (i.e., features never simultaneously accept nonzero values) of χi, i $= \{1, \ldots, \mathrm{N}\}$ by the exclusive feature bundling (EFB) technique;
2. Set $\theta_0(\chi) = \arg\min_c \sum_i^N L(y_i, c)$;
3. For m = 1 to M do
4. Calculate gradient absolute values:

$$r_i = \left| \frac{\partial L(y_i, \theta(x_i))}{\partial \theta(x_i)} \right|_{\theta(x)=\theta_{m-1(x)}},$$

5. Resample data set using gradient-based one-side sampling (GOSS) process:
$topN = a \times len(D); randN = b \times len(D);$

$sorted = GetSortedIndices(abs(r))$;
$A = sorted[1 : topN]$;
$B = RandomPick(sorted[topN : len(D)], randN)$; $D' = A + B$;

6. Calculate information gains:

$$V_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} r_i + \frac{1-a}{b}\sum_{x_i \in B_l} r_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} r_i + \frac{1-a}{b}\sum_{x_i \in B_r} r_i\right)^2}{n_r^j(d)}\right);$$

7. Develop a new decision tree $\theta_m(x)'$ on set D';
8. Update $\theta_m(\chi) = \theta_{m-1}(\chi) + \theta_m(\chi)$;
9. End for;
10. Return $\tilde{\theta}(x) = \theta_M(x)$.

4 Practical implementation

4.1 EDA & Pre-Processing

The model was created as a way to solve classification problems. In this article, we will examine how the model works with the problem of forecasting commodity prices without history and for a dataset with non-numerical data.

The dataset is available on the Kaggle website under the name "Mercari Price Suggestion Challenge." Established in 2013, Mercari Inc. is a Japanese company that operates one of the most popular C2C marketplaces in the Japanese market.

The data is already divided into training and testing sets. The dataset comprises the following seven characteristics: "name", "item_condition_id", "brand_name", "category_name", "shipping", "item_description" and "price."

Our initial model looks like this:

$$y = a_0 + a_1X_1 + \cdots + a_nX_n + \varepsilon$$

where:

- y is the dependent variable (price),
- $X_1, \ldots, X_n$ are the independent variables (features),
- $a_0, a_1, \ldots, a_n$ are the coefficients associated with each independent variable,
- ε represents the error term, which accounts for the variability in y that is not explained by the model.

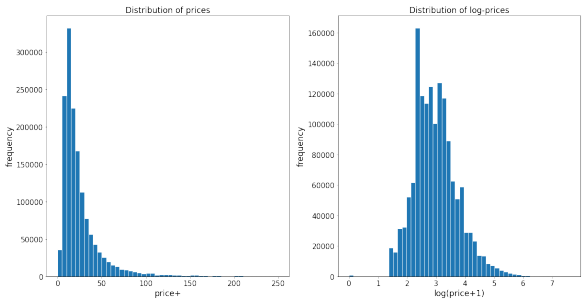

Figure 1: Distributions of prices

During the exploratory analysis and preparation of the data for work, it was chosen log(y) as the dependent variable, because after constructing the histograms of the distribution, it was found that the logarithmic distribution is the closest to the normal one. This is illustrated in the following figure.

Also, the general category was divided into three separate subcategories, missing values were processed. It's worth noting that since most of our characteristics are text, CountVectorizer, TfidfVectorizer were used to convert them to numeric values.

So, now we get a semi-logarithmic eight-factor model

$$log(y) = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + a_4X_4 + a_5X_5 + a_6X_6 + a_7X_7 + a_8X_8 + \varepsilon$$

4.2 Model training

In this section, we delve into the practical implementation of the LightGBM framework for price prediction.

The study was conducted in the `Python` environment using the `Params` function. For the 'objective' parameter, we designate 'regression' to align with our regression task. Additionally, we specify the boosting type as 'gbdt', which is the default setting. We include the 'data_sample_strategy' as our GOSS method, which is known for its effectiveness in dealing with large datasets. Furthermore, we activate the 'enable_bundle' option to indicate our utilization of the Exclusive Feature Bundling (EFB) technique. The chosen evaluation metric is 'RMSE', reflecting the Root Mean Squared Error.

This configuration enables us to leverage the strengths of LightGBM for accurate and efficient price prediction.

4.3 Evaluation

To evaluate the performance of the developed model, we will use the metrics R^2, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) Chicco et al. (2021), where the Mean Absolute Error (MAE) is calculated using the formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}|,$$

the Coefficient of Determination (R^2) is computed using the following formula:

Metric	LGBM	XGBoost
RMSE	0.47667	0.47778
MAE	0.35779	0.35897
R-squared	0.59461	0.59274
Time, s	810.24768	2116.68744

Table 1: Model Performance Comparison

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

and the Root Mean Squared Error (RMSE) is calculated using the formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

- n is the total number of observations in the dataset,
- $\hat{y}_i$ is the predicted price,
- y_i is the actual price.

For comparison, we also implemented the XGBoost model, which was mentioned in previous sections and has been further refined into the LightGBM framework.

It is important to note that in our experiment, we employed the classical implementations of both methods with identical parameters.

The obtained comparative table presents model evaluations based on training for 1000 iterations using the provided training data:

In order to show that the distribution of the real price and the model are quite similar, we visualized the entire sample, 1000 items and 100 items, which is the most representative. Where real prices are shown in blue, predicted by LightGBM in pink and predicted by XGBoost in green.

5 Conclusion

Through our work, we have successfully developed a model that can predict the price of a product without a history. Our model is based on the characteristics of the product and data on similar products found in the vicinity, and we utilized the LightGBM method. We used a real data set from Mercari, which is one of the most popular C2C marketplaces in Japan, to showcase the novelty of the LightGBM method. Pseudo-codes were provided to demonstrate how these new algorithms work.

Our findings showed that the LightGBM method provides highly accurate predictions for product prices without history, and is significantly faster compared to other models. This study confirms the efficacy of using the LightGBM method for price prediction. The results of our study can be valuable for companies looking to develop automated recommender systems for a historical commodity price forecasting model.

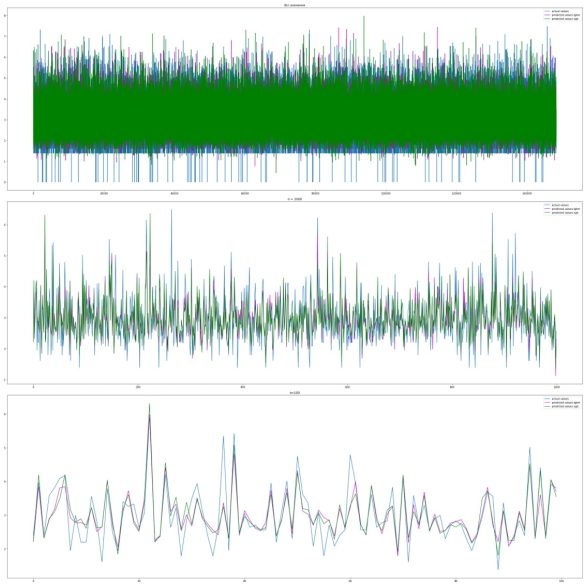

Figure 2: Distributions of prices

In our future research, we plan to explore ensemble methods to potentially achieve even higher accuracy. Techniques such as stacking or blending can be employed by combining the predictions of multiple models, including LightGBM.

Acknowledgement

Svitlana Drin acknowledges financial support from the Knowledge Foundation Grant (Dnr: 20220115).

References

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Chicco, D., Warrens, M. J., and Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, 7:e623.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2 edition.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.

Schapire, R. E. and Freund, Y. (2013). Boosting: Foundations and algorithms. *Kybernetes*, 42(1):164–166.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.