УДК 519.2

S. Drin, A. Kriuchkova, V. Toloknova

https://doi.org/10.18523/2617-7080620236-13

PREDICTIVE MODEL FOR A PRODUCT WITHOUT HISTORY USING LIGHTGBM. PRICING MODEL FOR A NEW PRODUCT

The article focuses on developing a predictive product pricing model using LightGBM. Also, the goal was to adapt the LightGBM method for regression problems and, especially, in the problems of forecasting the price of a product without history, that is, with a cold start.

The article contains the necessary concepts to understand the working principles of the light gradient boosting machine, such as decision trees, boosting, random forests, gradient descent, GBM (Gradient Boosting Machine), GBDT (Gradient Boosting Decision Trees). The article provides detailed insights into the algorithms used for identifying split points, with a focus on the histogram-based approach.

LightGBM enhances the gradient boosting algorithm by introducing an automated feature selection mechanism and giving special attention to boosting instances characterized by more substantial gradients. This can lead to significantly faster training and improved prediction performance. The Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques used as enhancements to LightGBM are vividly described. The article presents the algorithms for both techniques and the complete LightGBM algorithm.

This work contains an experimental result. To test the lightGBM, a real dataset of one Japanese C2C marketplace from the Kaggle site was taken. In the practical part, a performance comparison between LightGBM and XGBoost (Extreme Gradient Boosting Machine) was performed. As a result, only a slight increase in estimation performance (RMSE, MAE, R-squard) was found by applying LightGBM over XGBoost, however, there exists a notable contrast in the training procedure's time efficiency. LightGBM exhibits an almost threefold increase in speed compared to XGBoost, making it a superior choice for handling extensive datasets.

This article is dedicated to the development and implementation of machine learning models for product pricing using LightGBM. The incorporation of automatic feature selection, a focus on high-gradient examples, and techniques like GOSS and EFB demonstrate the model's versatility and efficiency. Such predictive models will help companies improve their pricing models for a new product. The speed of obtaining a forecast for each element of the database is extremely relevant at a time of rapid data accumulation.

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

Intoduction

In today's rapidly evolving business landscape, companies are confronted with formidable challenges in effectively managing sales strategies and determining optimal pricing strategies for their products. A particularly intricate aspect of this endeavor is establishing the price for a new product lacking historical sales data.

The necessity to devise accurate and reliable pricing strategies for novel products has led to the emergence of sophisticated predictive models. These models serve as invaluable tools for companies aiming to ascertain optimal prices without sufficient past experience. Among these models, the LightGBM (Light Gradient Boosting Machine) algorithm stands out as a powerful tool for quickly finding the optimal forecast. The article uses the LightGBM method for a predictive pricing model for a new product, which is focused on the complex challenges of modern markets.

The relevance of this research is underscored by the critical role of pricing and decision-making, especially during periods of economic instability and the challenges associated with dynamic markets and the lack of historical sales data for new products.

The main goal of this study is to implement the LightGBM algorithm into the predictive pricing model for setting optimal prices for new products.

The paper is organized as follows. The second section delves into the theoretical underpinnings that underlie the chosen machine learning model for price forecasting. This section is based on the papers [?], [?], [1], [2], where essential concepts

are explained. Friedman's article [?] describes in great detail a general gradient-descent "boosting" paradigm. Specific algorithms are presented for least-squares, least-absolute-deviation, and Huber-M loss functions for regression, and multi-class logistic likelihood for classification. Many useful concepts and techniques can be found in Elements of Statistical Learning [?] by Hastie, Tibshirani, and Friedman. The authors introduce us to decision trees, bagging, boosting, random forest, gradient descent and much more. In [1] we can get acquainted with algorithms for identifying split points.

The third section is devoted to the basic model, namely to its distinguishing features and efficiency. The main concept of LightGBM can be found in source [2] as it is the original developer documentation. Using the original source a comprehensive explanation of the newly introduced algorithms, GOSS and EFB, is provided, accompanied by insights into their motivations and unique characteristics.

Forth section contains some numerical results for data, root mean squared errors (RMSE), mean absolute error (MAE) and R-squared, the benefits of which are clearly explained in [4]. The evaluation of the model is carried out using training data, and a comparative assessment is performed, juxtaposing the model's performance against that of the XGBoost model.

Preliminary theoretical base

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 [5].

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.

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 [?], [6], 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.

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 $F(x_0) \geq F(x_1) \geq \geq F(x_2) \geq \ldots$, 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.

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. [?]

The generic gradient tree-boosting algorithm for regression [?].

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

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

   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

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

   d Update

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

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.[2]

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 [2] 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.

LightGBM

LightGBM is one of the most recent types of Gradient Boosting Decision Trees (GBDT). This model was developed by a team of researchers at Microsoft in 2016. It was created as an improvement over one of the most popular models- XGBoost, which is known for its speed and reliability in multi-class classification projects.

The need to enhance XGBoost arose for a very obvious reason – to achieve even greater efficiency and faster implementation. As mentioned, the most computationally intensive task in GBDT is the search for optimal split points. This complexity is directly proportional to both the number of features and the number of instances. Consequently, when dealing with large datasets, we encounter speed-related issues. It was proposed to reduce the number of data instances and the

number of functions. This led to the introduction of two new techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). These innovations were aimed at mitigating the computational challenges associated with GBDT training on large datasets.

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

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 $O(\#data \times \times \#bundle)$, 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 [2]

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 [2], 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 $= \{(\chi 1, y1), (\chi 2, y2), \ldots, (\chi \mathrm{N}, y\mathrm{N})\}$, $\chi \mathrm{i} \in \chi$, $\chi \subseteq \mathrm{R}$, $y\mathrm{i} \in \{-1, +1\}$;

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 $\chi \mathrm{i}, \mathrm{i} = \{1, \ldots, \mathrm{N}\}$ 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:

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

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 $\tilde{\theta}(x) = \theta_M(x)$.

Practical implementation

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:

$$y = a_0 + a_1 X_1 + \cdots + a_n X_n + \varepsilon$$

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.

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.

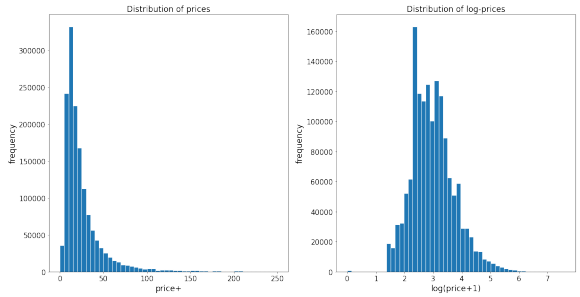

Figure 1. Distributions of prices

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

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

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.

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) [4], 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:

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

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:

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

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.

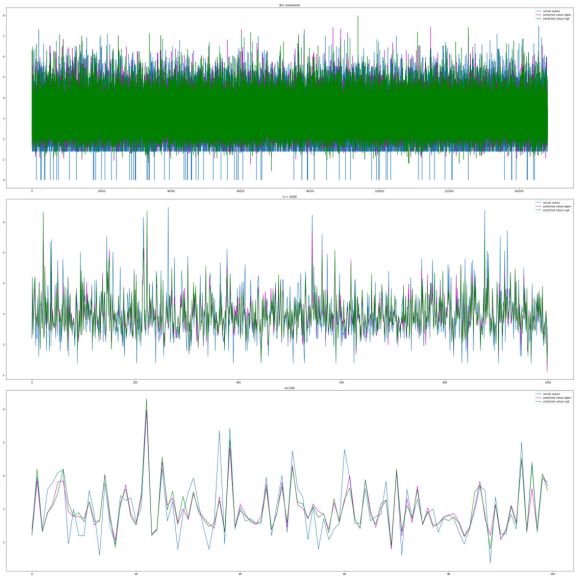

Figure 2. Distributions of prices

Where real prices are shown in blue, predicted by LightGBM in pink and predicted by XGBoost in green.

Conclusion

As a result of this work, we built a model for predicting the price of a product without a history based on its characteristics and data on nearby similar products using the LightGBM method on the example of a real data set from Mercari - one of the most popular C2C marketplaces on the Japanese market. We have shown what is the novelty of the LightGBM method and how these new algorithms work, pseudo-codes are provided. The obtained results showed that the model using the LightGBM method provides sufficiently high accuracy of forecasting the product price without history compared to other models, and also works many times faster. Thus, the study confirms the effectiveness of using the LightGBM method for price prediction. The results of the study may be useful to companies wishing to develop automated recommender systems for a historical commodity price forecasting model. As a further development in this research, several improvement ideas can be tried, such as explore ensemble methods by combining the predictions of multiple models, including LightGBM, to potentially achieve even higher accuracy. Techniques such as stacking or blending can be employed.

References

1. Chen Tianqi and Guestrin Carlos, "XGBoost: A Scalable Tree Boosting System," Association for Computing Machinery. **10**, 785–794 (2016).
2. Ke Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," NIPS (2017).
3. N. Dunbray, R. Rane, S. Nimje, J. Katade, and S. Mavale, "A Novel Prediction Model for Diabetes Detection Using Gridsearch and A Voting Classifier between Lightgbm and KNN," 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–7 (2021).
4. C. Davide, J. Warrens, and G. Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," PeerJ Computer Science. **7** (2021).
5. C. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal. **27** (3), 379–423 (1948).
6. Robert E. Schapire, and Yoav Freund, "Boosting: Foundations and algorithms, " Kybernetes. **42** (1), 164–166 (2013).

Крючкова А. О., Толокнова В. В., Дрінь С. С.

ПРОГНОСТИЧНА МОДЕЛЬ ДЛЯ ПРОДУКТУ БЕЗ ІСТОРІЇ З ВИКОРИСТАННЯМ LightGBM

Статтю присвячено розробці прогнозної моделі ціноутворення за допомогою LightGBM. Також метою було адаптування методу LightGBM для задач регресії та, особливо, задач прогнозування ціни продукту без історії, тобто з холодним стартом.

Стаття містить необхідні поняття для розуміння принципів роботи методів з посиленим градієнтом, таких як дерева рішень, бустинг, випадкові ліси, градієнтний спуск, GBM (машина посилення градієнта), GBDT (дерева рішень градієнтного підвищення). У статті наведено інформацію про алгоритми, які використовуються для пошуку точок розбиття, з акцентом на алгоритм на основі гістограм.

LightGBM покращує алгоритм градієнтних методів, запроваджуючи автоматичний механізм вибору функцій, приділяючи особливу увагу точкам посилення, що характеризуються більш вагомими градієнтами. Це може призвести до значно швидшого навчання та покращення ефективності передбачення. Описано методи односторонньої вибірки на основі градієнта (GOSS) і ексклюзивного пакетування функцій (EFB), які використовують для вдосконалення LightGBM.

Робота містить експериментальне дослідження. Щоб перевірити LightGBM, було взято реальний набір даних одного японського ринку C2C із сайту Kaggle. У практичній частині було проведено порівняння продуктивності LightGBM і XGBoost (Extreme Gradient Boosting Machine). У результаті було виявлено лише незначне підвищення в оцінках продуктивності (RMSE, MAE, R-squard) LightGBM порівняно з XGBoost, однак існує помітний контраст у часовій ефективності в процедурі навчання. LightGBM демонструє майже втричі більшу швидкість порівняно з XGBoost, що робить його кращим вибором для роботи з великими наборами даних.

Цю статтю присвячено розробці та впровадженню моделей машинного навчання для ціноутворення продуктів за допомогою LightGBM. Включення автоматичного вибору функцій, зосередженість на прикладах із високим градієнтом і таких методах, як GOSS і EFB, демонструють універсальність і ефективність моделі. Такі прогнозні моделі допоможуть компаніям покращити свої моделі ціноутворення на новий товар. Швидкість отримання відповідного прогнозу для кожного елемента бази є вкрай актуальною в час швидкого накопичення даних.

Ключові слова: GBM, GBDT, LightGBM, GOSS, EFB, прогнозна модель.

Матеріал надійшов 27.12.2023