2023 4th International Conference on Computer, Big Data and Artificial Intelligence (ICCBD+AI) | 979-8-3503-7323-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCBD-AI62252.2023.00106

Price Prediction and Analysis Based on the LightGBM Regression Model

ShengYu.Huang*
School of Information Technology and Management
Hunan University of Finance and Economics
Changsha,China
202105620207@mails.hufe.edu.cn

***Abstract*—Price forecasting is widely used in many fields.It is of great significance for economic development to find out the model and method of predicting price accurately.Traditional methods of predicting prices include market comparison, income restoration, and cost estimation, but these methods require a significant amount of human and material resources and usually fail to predict housing prices accurately due to the influence of numerous factors. Therefore, this paper aims to use the LightGBM regression model and LGBMRegressor algorithm for the prediction and analysis of housing prices. We collected a large amount of data on second-hand housing prices and preprocessed the data to eliminate entries with missing values. Subsequently, we performed a general descriptive statistical analysis of the data and established feature data and label data based on different characteristics such as region, age of the house, nearby geographical conditions, cultural and traffic factors. Finally, we divided the dataset and made model predictions. The experimental results show that the predicted values of the regression model are basically consistent with the actual fluctuations in housing prices, allowing for relatively accurate predictions.**

Keywords- LightGBM, LGBMRegressor algorithm, regression model.

I. INTRODUCTION

Since the implementation of real estate market reform in 1998, there has been a steady increase in house prices in China, leading to various negative impacts on labor mobility, investment climate, resident spending patterns, and overall resource allocation efficiency.[1] Therefore, accurate prediction of housing prices has become crucial.

Machine learning and deep learning techniques offer potential solutions for automating the price prediction process.[2] In order to meet the high market demand, an efficient and high-performance algorithm is needed. This algorithm should be capable of handling large-scale data and making precise price predictions. While there are currently numerous predictive algorithms available, further exploration and research are necessary to determine which algorithm is the most practical for the market requirements.

Various techniques have been employed by experts for forecasting housing prices, such as the creation of a novel housing search index (HSI) derived from online search activities associated with the process of purchasing a house. Studies have discovered that HSI possesses a valuable predictive ability for future variations in housing prices, both within the sample and beyond, and even after adjusting for the influence of established predictors. These findings have been connected to search-based friction models. Consequently, these outcomes suggest that search data may be leveraged as an advanced signal of market trends. [3].

In order to streamline the analysis, we have explored various configurations including algorithms, delay factors, hidden neurons, and data splitting ratios. As a result, we have identified a basic neural network structure with three delays and three hidden neurons, which consistently achieves a stable performance with an average relative root mean square error of around 0.75% across the ten cities during the training, validation, and testing stages. These findings can be utilized independently or integrated with fundamental forecasts to generate insights into residential housing price trends and conduct policy analysis.[4]

Moreover, the convolutional neural network (CNN) primarily employs the convolution kernel in the convolution layer to select a feature set that is classified. The pioneering work by Kim (2014) introduced CNN into text classification tasks, demonstrating promising outcomes. In the domain of price prediction, Selvin et al. (2017) explored three distinct deep learning architectures, namely LSTM, RNN, and CNN, and compared their performances. The experimental results indicated that the CNN-sliding window model showcased superior performance with a lower percentage error. Subsequently, Lee and Soo (2017) proposed a recurrent convolutional neural network (RCN) that combined the strengths of convolutional sequence modeling, word embedding for stock price analysis, and information extraction from financial news [5].

In the construction domain, SVM, LSSVM, and PLS algorithms are commonly utilized to predict housing values. By leveraging various attributes, the housing value of Boston suburbs can be predicted [6]. In each market, out-of-sample forecasts are generated using a range of models such as vector autoregressive (VAR) and vector error-correction (VEC) models. Additionally, Bayesian, spatial, and causality versions of these models with different priors are also employed to improve the accuracy of the forecasts[7]. To enhance the accuracy of house price predictions, this study employs a novel approach called the Average Model (AM) algorithm before utilizing regression processing techniques. This technique calculates the average of multiple regression models in a unique manner. By evaluating essential models and achieving a 90% accuracy rate, the AM algorithm significantly improves the precision of house price predictions [8].

To address the heteroscedastic nature of housing data, we propose a hybrid algorithm, which combines linear regression, clustering analysis, nearest neighbor classification, and Support Vector Regression (SVR) methods. This algorithm aims to use one method's output

979-8-3503-7323-3/23/$31.00 ©2023 IEEE
DOI 10.1109/ICCBD-AI62252.2023.00106

as the input for another method in the process of predicting home prices[9]. A hybrid of genetic algorithm and support vector machines (G-SVM) approach is presented in housing price forecasting. Support vector machine (SVM) has been proven to be a robust and competent algorithm for both classification and regression in many applications [10].

The provincial house price data in China from 2000 to 2019 undergoes pre-processing such as statistical analysis, handling missing values, and selecting key features. Subsequently, the pre-processed data is fed into several models to gauge their predictive performance and the best one is identified, followed by hyper-parameter optimization for further enhancement[11]. In their study, Fathalla et al. introduced a deep neural network incorporating long short-term memory (LSTM) and convolutional neural networks (CNN) for price prediction. The experimental results demonstrated that the proposed model outperformed the support vector machine (SVM) in terms of average absolute error accuracy score.[12]

This paper uses LightGBM to analyze big data on housing prices, extract feature information [13], establish predictive models, and draw conclusions.

II. LIGHTGBM

GBDT, which stands for Gradient Boosting Decision Tree, has been widely used and recognized in the field of machine learning. It utilizes the iterative approach of training weak classifiers, or decision trees, to produce an optimal model that is resistant to overfitting while delivering favorable training outcomes. GBDT finds extensive application in various industries and scenarios such as click-through rate prediction, search ranking, and multi-classification tasks. Moreover, it shows tremendous potential in data mining competitions, with more than 50% of the champion solutions on Kaggle relying on this algorithm. LightGBM, on the other hand, is a framework that implements the GBDT algorithm, providing efficient parallel training and faster speed, lower memory consumption, and better accuracy, and the ability to handle massive data in a distributed manner. The core idea of the LightGBM algorithm is based on the gradient boosting tree learning framework, and it introduces optimization strategies to improve training speed and model performance. LightGBM uses the histogram algorithm to convert sample traversal into histogram traversal, significantly reducing time complexity. It also filters out samples with small gradients during the training process using a one-sided gradient algorithm, reducing a large amount of computation. Additionally, it constructs trees using a leaf-wise growth strategy, reduces unnecessary computation, and accelerates calculations using optimized feature parallelism and data parallelism methods. The algorithm also optimizes caching to increase cache hit rates.

A. Histogram algorithm

The fundamental concept behind the histogram algorithm is to convert continuous floating-point feature values into a set of K integers, simultaneously constructing a histogram of size K. During the data traversal process, cumulative statistics are accumulated in the histogram using the discretized values as indices. Once the data traversal is completed, the histogram will have accumulated the necessary statistics. Based on the discrete values within the histogram, the algorithm then searches for the optimal split point for further processing.

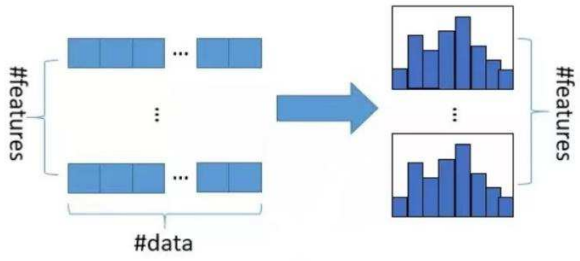

Figure 1. Histogram analysis

The histogram algorithm can be described in the following way: Initially, the algorithm calculates the required number of bins for each feature and assigns a unique integer to each bin. Next, it divides the range of floating-point numbers into a specified number of intervals, with the same number of intervals as the number of bins. Then, it updates the sample data by mapping each data point to the corresponding bin value within the intervals. Finally, the algorithm represents the data using a histogram, which is essentially a representation of the distribution of the data across the bins. While this process may appear complex, it is essentially a statistical technique that organizes large-scale data into a histogram format.

LightGBM implements an optimization technique called histogram subtraction to improve its performance. This optimization involves deriving the histogram of a leaf node by subtracting the histograms of its parent node and sibling node, effectively doubling the computation speed.

Traditionally, constructing a histogram necessitates traversing all the data points within a leaf node. However, with histogram subtraction, LightGBM only needs to traverse the 'k' buckets in the histogram, where 'k' represents the number of bins.During the tree construction process, LightGBM computes the histograms of smaller leaf nodes first. It then utilizes histogram subtraction to efficiently calculate the histograms of larger leaf nodes, minimizing the cost of obtaining the histograms of sibling leaves.By implementing this approach, LightGBM is able to construct decision trees more efficiently by reducing the computation required for histogram calculation, resulting in improved performance overall.

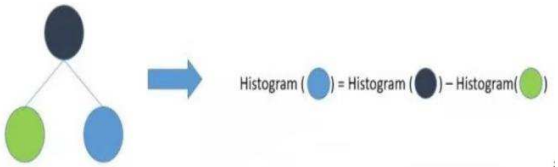

Figure 2. Leaf histogram

B. Leaf-wise

Building upon the Histogram algorithm, LightGBM further optimizes its approach. Firstly, it discards the

level-wise decision tree growth strategy commonly used by most GBDT tools, and instead adopts a leaf-wise algorithm with depth constraints. XGBoost adopts a level-wise growth strategy, which traverses the data once and simultaneously splits the leaves at the same level. This strategy is easy to optimize for multithreading and provides good control over model complexity, making it less prone to overfitting. However, in practice, the level-wise algorithm is inefficient because it treats all leaves at the same level indiscriminately. In reality, many leaves have low splitting gains and do not require searching and splitting, resulting in unnecessary computational overhead.

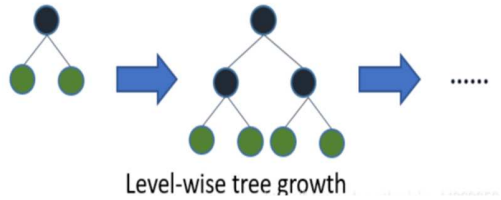

Figure 3. Level-wise tree growth

LightGBM utilizes the Leaf-wise growth strategy, which involves the iterative identification of the leaf node that provides the highest gain for splitting from all current nodes. This process is repeated in a loop. By adopting this approach, Leaf-wise offers certain advantages over the Level-wise growth strategy. For instance, with the same number of splits, Leaf-wise can effectively reduce more errors, thereby achieving higher accuracy. However, one potential drawback of Leaf-wise is that it may lead to the growth of decision trees that are relatively deep, thus increasing the risk of overfitting the training data.To address this concern, LightGBM introduces a maximum depth limit on top of the Leaf-wise growth strategy. This limit serves as a preventive measure against overfitting while ensuring efficient performance.

C. Goss

LightGBM utilizes the computed gradients to filter samples. Intuitively, larger gradients should indicate under-learning. If samples with large gradients can be predicted correctly, they will contribute more to the gain. Therefore, when splitting nodes, accurate division of samples with large gradients is desired, while smaller gradient samples can afford errors. Accordingly, when filtering samples, large gradient samples are retained, and only a portion of small gradient samples are removed. This is the essence of the GOSS approach, with the specific process as follows:

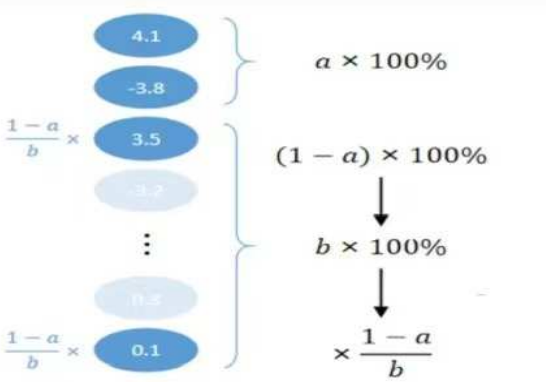

Figure 4. Goss operation

1) First, arrange the gradients in descending order of absolute values for ease of filtering.

2) Set a threshold ratio, denoted as "a." Samples in the top "a×100%" after sorting are considered large gradient samples and are all retained.

3) The remaining "(1-a)×100%" of the samples are considered small gradient samples. A random sampling is conducted, with a sampling ratio of "b×100%".

4) Since random sampling does not apply to all samples, it alters the original data distribution. To maintain the data distribution as much as possible, the sampled samples need to be multiplied by a coefficient to preserve their original distribution. This coefficient is:

1.

$$\frac{n(All\ small\ gradient\ samples)}{n(Small\ gradient\ sample\ obtained\ by\ sampling)} = \frac{(1-a)\times n(All\ samples)}{b\times(All\ samples)} = \frac{1-a}{b}$$

D. Exclusive Feature Bundling Algorithm

In high-dimensional data, sparsity is a common characteristic, leading to the development of a lossless approach for dimensionality reduction of features. Typically, the features that are grouped together are mutually exclusive, meaning they do not have non-zero values simultaneously (one-hot encoding). This guarantees that combining two features will not result in information loss. However, if two features are partially non-mutually exclusive (both features can have non-zero values in certain cases), a conflict ratio can be calculated to quantify the degree of non-exclusivity. When the conflict ratio is low, it is possible to bundle these partially non-mutually exclusive features without significantly impacting the final accuracy. The Exclusive Feature Bundling (EFB) algorithm suggests that by combining and bundling certain features, the overall number of features can be reduced. This reduction in feature count reduces the time complexity for constructing histograms from O(#data* #feature) to O(#data* #bundle), where #bundle represents the number of feature bundles, which is significantly smaller than the total number of features (#feature).

III. Experiment

Source and Introduction of the Data: The LightGBM model is capable of effectively handling large-scale data. Therefore, based on this, the LGBMRegressor algorithm is used to establish a regression model. Specifically, a large amount of second-hand housing data is analyzed, and various feature factors are processed using the algorithm model to predict the prices of second-hand houses. The data for this modeling comes from the internet.

A. Data Preprocessing

Real-world data may contain a large number of missing values, noisy data, or outliers caused by human input errors, which are extremely detrimental to the training of algorithm models. The result of data cleaning is the processing of various dirty data in a corresponding manner to obtain standardized, clean, and continuous data, which can be used for data statistics, data mining, and other purposes. Data preprocessing typically includes methods such as data cleaning, reduction, aggregation,

transformation, and sampling. The quality of data preprocessing determines the accuracy and generalization value of subsequent data analysis, mining, and modeling work. The following briefly introduces the main preprocessing methods in data preprocessing work and uses the head() method of the Pandas tool to view the first five rows of data:

TABLE I. PANDAS DATA TABLE

ID	MSSubClass	MSZoning	SaleType	SaleCoundition	SalePrice
1	60	RL	WD	Normal	208500
2	20	RL	WD	Normal	181500
3	60	RL	WD	Normal	223500
4	70	RL	WD	Normal	140000
5	60	RL	WD	Normal	250000

From the above figure, it can be seen that there are a total of 81 fields: 79 feature variables, where "Id" is just an index and should be removed during modeling. Additionally, this dataset contains a large number of missing data.

Calculating the percentage of missing values for each feature:

Using the isna() and mean() methods of the Pandas tool to calculate the percentage of missing data for each feature: calculating the feature variables with missing data exceeding 50%.

TABLE II. FEATURE MISSING VALUE TABLE

Column_name	percentage
PoolQC	99.520548
MiscFeature	96.301370
Alley	93.767123
Fence	80.753425

- From the above figure, it can be seen that a total of 5 feature variables have missing data exceeding 80%. In the next step of this project, these feature variables will be removed and will not be involved in the subsequent modeling.
- Experimental Data Analysis
- It is found that there are 1459 records classified as

Model name	Name of index	Targeted value
Test set		
Lightgbm regression model	Interpretable variance value	0.89
	The average absolute error	15732.75
	Error of mean square	750967526.17
	R*R	0.89

AllPub and only 1 record classified as NoSeWa, indicating a significant data imbalance. The distribution analysis of the Street, Condition2, RoofMatl, and Heating features also shows significant data imbalances. Subsequently, the age of the houses is constructed using YrSold and YearBuilt, and descriptive statistical analysis is performed using the describe() method of the Pandas tool.

TABLE III. DESCRIPTIVE STATISTICAL ANALYSIS

count	1460.000000
mean	36.547345
std	30.250152
min	0.000000
25%	8.000000
50%	35.000000
75%	54.000000
max	136.000000
Name:Age House,dtype:float64	

- From the figure above, it can be observed that the age of the houses ranges from less than 1 year to a maximum of 136 years, with an average age of around 36 years.

Next, we construct the TotalBsmtBath, TotalBath, and TotalSA features, and process the categorical features. Feature processing is divided into two categories: one is for ordinal features, for which direct numerical mapping is performed; the other is for encoding categorical features, for which one-hot encoding is applied. Subsequently, the feature data and label data are established, where SalePrice is the label data, and the features exclude SalePrice. The dataset is then split into a training set and a validation set, with 70% for training and 30% for validation. The main algorithm used is LGBMRegressor, which is used for target regression. The model parameters are as follows:

TABLE IV. MODEL PARAMETER

Number	Model name	Parameter
1	Lightgbm regression model	objective='regression'
2		num_leaves=4
3		learning_rate=0.01
4		n_estimators=12000
5		max_bin=200
6		bagging_fraction=0.75
7		bagging_freq=5
8		bagging_seed=7
9		feature_fraction=0.4

The evaluation metrics mainly include explained variance score, mean absolute error, mean squared error, R-squared value, and so on.

TABLE V. EVALUATION INDICATORS AND RESULTS

From the table above, it can be seen that the R-squared value is 89% and the explained variance score is 89%. This indicates that the lightgbm regression model is excellent, and the performance is very good. Due to the large number of features, the top 20 features are selected for visualization.

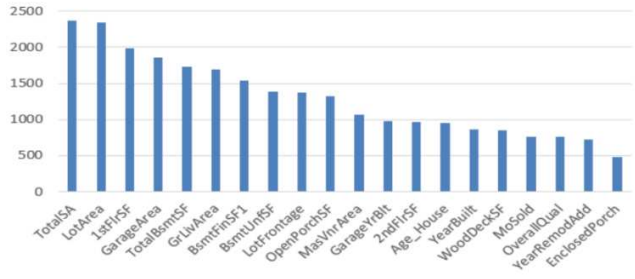

Figure 5. Model feature importance diagram

The importance of the feature variables for this model can be seen from the above figure, in the following order: TotalSA, LotArea, 1stFlrSF, GarageArea, TotalBsmtSF, GrLivArea, BsmtFinSF1, BsmtUnfSF, LotFrontage, OpenPorchSF, MasVnrArea, GarageYrBlt, and so on. Below is the comparison chart of actual values versus predicted values.

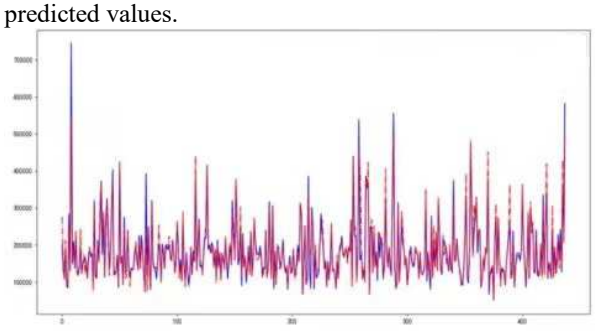

Figure 6. Comparison diagram between real value and test value

The experiment starts with our most common second-hand housing data, and through a large amount of data training, a regression model is established and predictions are made. From the above figure, it can be seen that the fluctuations in actual values and predicted values are basically consistent, indicating that the model has an excellent fit. This ultimately proves that the model we proposed has performed well.

IV. Discussion and Future Work

The study proposed in this article focuses on the prediction and analysis of house prices using the LightGBM algorithm, aiming to achieve more accurate price predictions and discard traditional forecasting methods. This method can provide fast and accurate reference for consumers, developers, governments, and other relevant parties. Scientific, rapid, and effective [14] prediction methods like this should be widely promoted in the real estate industry to drive better development and lay a solid foundation for the national economy. The model discussed in this article, when trained on a large amount of data with a 70% training set and a 30% validation set, has validated the relationship between actual and predicted prices, demonstrating the feasibility of the LightGBM algorithm's regression model. In the future of the real estate industry, the LightGBM algorithm will be used not only for price prediction but also for site selection, customer promotion in real estate sales, price changes, market trends, and other aspects to enhance investor security, increase consumer satisfaction in home purchases, and improve developer profits. This will promote the healthy development of the real estate industry and drive it towards a path of rapid intelligence and technological advancement.

References

[1] Jingyun Zhang , Guangping Liu , Xueyuan Li , Han Jiang PMID: 32109247PMCID: PMC7048277DOI: 10.1371/journal.pone.0228598

[2] Baoyang Cui,Zhonglin Ye,Haixing Zhao,Zhuome Renqing,Lei Meng, andYanlin YangElectronics 2022, 11(18), 2932; https://doi.org/10.3390/electronics11182932

[3] Stig Vinther Møller , Thomas Pedersen , Erik Christian Montes Schütte , AllanTimmermann ,Published Online:9 Fe2023https://doi.org/10.1287/mnsc.2023.4672

[4] Yun Zhang Published: 12 May 2022 Volume 34, pages 14763–14776, (2022)

[5] Shengting Wu,Yuling Liu,Ziran Zou&Tien-Hsiung Weng Pages 44-62 | Received 30 Jan 2021, Accepted 03 Jun 2021, Published online: 14 Jun 2021

[6] JingyiMu,1FangWu,2and AihuaZhangVolume 2014 | Article ID 648047 | https://doi.org/10.1155/2014/648047

[7] Rangan Gupta & Stephen M. MillerPublished: 19 November 2010 Volume 48, pages 763–782, (2012)

[8] Fu Jintao;Zhou Yong;Qiu Qian;Xu Guangwei;Wan Neng10.32604/JQC.2022.038358

[9] Özöğür Akyüz Süreyya;Eygi Erdogan Birsen;Yıldız Özlem;Karadayı Ataş Pınar10.1007/S10614-022-10298-8

[10] JirongGu;MingcangZhu;LiuguangyanJiang10.1016/j.eswa.2010.08.123

[11] Ding Xiafei;Wang Weiya;Zhang Yiqian;Zhong Xiaoyuqian10.2991/AEBMR.K.220307.035

[12] Fathalla, A.; Salah, A.; Li, K.; Li, K.; Francesco, P. Deep end-to-end learning for price prediction of second-hand items. Knowl. Inf. Syst. 2020, 62, 4541–4568.

[13] ZengXiangruiAbdullahNibras;LiangBaixue10.1016/J.HELIYON.2023.E23071