The Performance of Hotel Recommendation System Based on Different ML Methods

Group 77- Yulu Wang(2727657) Yunshan Wang(2721621) Siqi Wei(2724831)

Vrije Universiteit, Amsterdam, Netherlands

1 Introduction

1.1 Task description

In the past, traditional portal websites, Internet products, and other fields had information push based on manual configuration push. However, this method is inefficient, and the recommendation effect is still open to question. With the explosive growth of data volume and the further application of big data and algorithm models on the Internet, algorithms gradually replace manual recommendations to achieve the effect of personalized recommendations. Recommender system (RS) mainly refers to the application of collaborative intelligence to recommendation technology, which aims to help users solve the problem of unclear purpose in massive information, quickly obtain information, and actively screen information pain points, thereby improving and further converting users [1]. The personalized recommendation system can effectively solve the problem of information overload and provide an ordered recommendation list according to the user's historical preferences, product characteristics, and environmental factors such as user and product transactions (such as time, season, location, etc.) [2]. This can enhance and improve the user experience of exciting items. At the same time, it helps enterprises maximize the attraction of users, retain users, increase user stickiness, and improve user conversion rates [3]. Recommendation systems can be roughly divided into three categories: collaborative filtering systems (collecting information about user interactions with items to generate recommendations), content-based systems (the user may like items similar to what he once wanted), and hybrid recommendation models (using interaction information, user and product metadata) [4]. At present, collaborative filtering technology has become the most widely used recommendation technology and has been commonly used in many famous commercial systems such as Amazon, Netflix, Taobao, etc. Amazon introduced the item-based collaborative filtering algorithm (ItemCF algorithm) in 1998 [5]. The advantages of this algorithm are mainly these: simple, scalable, often surprising, and practical recommendations; can be updated immediately based on new information recommended by users; highly interpretable. These companies using recommender systems increase sales by personalizing the customer experience [6].

But these models also have limitations, such as long-tail effects, cold starts of users and items, etc. If a recommendation system is in the long tail effect for a long time, it will cause recommendation fatigue, and its recommendation effect

will be weakened. The long tail effect is that the more referrals, the more clicks, and the more user trajectories formed, the hotter it becomes [**?**]. Cold start is in the early stage of recommender systems. There is no intersection information between users and items. That is, there is no user behavior trajectory, and it cannot be recommended through similar collaboration or user preferences [8]. If the prediction is not accurate and timely, it will also cause the problem of information redundancy and even lead to the customer's psychological reaction [9].

With the increasingly fierce competition in the global travel market, personalized recommendation services have become very important for the online travel industry. Companies hope to free users from numerous travel choices through recommendation and matching and guide users to quickly find items of interest, thereby simplifying users' travel purchases [10]. Originating from a former Kaggle competition, this research hopes to create a hotel recommendation system based on machine learning methods to help booking companies like Expedia organize the search results for a user. The problem is defined as a multi-classification problem; after feature engineering, multiple machine learning models are constructed to predict the hotels that users are likely to book, given detailed probabilities of potential choices.

1.2 Dataset description

Observed by reading the data, training data has many features and samples (54 columns and 4958347 rows), while testing data has 4959183 samples and 50 features. Many variables are discrete, and some continuous variables are left-skewed or right-skewed. The distribution of each variable in training data is shown in Fig.1.

2 Methods

2.1 Workflow

The process model of the standard recommendation domain is shown in Fig.2. Since this task provides a data source, we start directly by reading the original data, constructing the final feature through multiple EDA, generating features, and comparing the recommendation model and algorithm to select the recommendation model to complete the final prediction.

2.2 Data preprocessing

Type conversion The data shows that date_time, click_bool, booking_bool are read as object types and converted to date type and bool type, respectively.

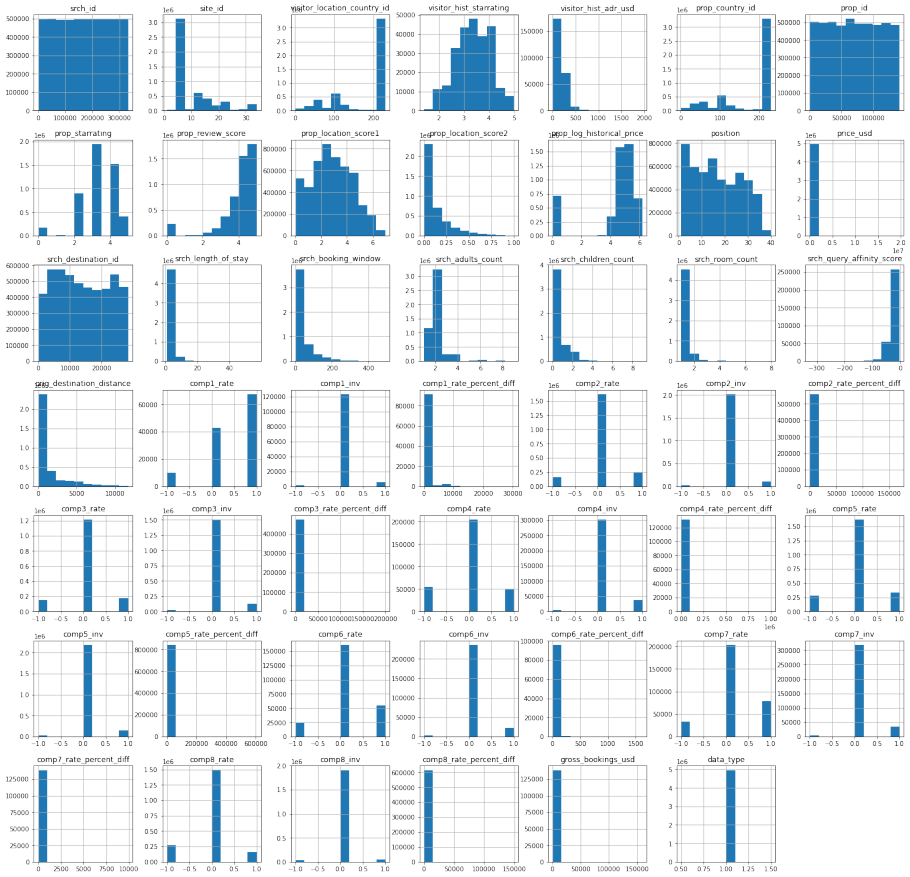

Fig. 1. Histogram of all variables

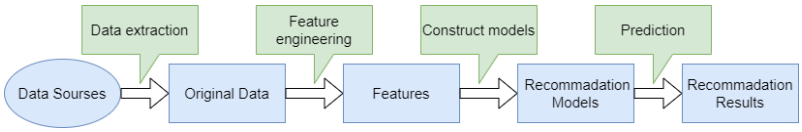

Fig. 2. Workflow for data mining in recommendation domain

Missing value By observing the info information, you can see that there are NaN values in many columns of the data. NaN values cannot be saved for int columns, so convert them to Int type when processing, process, or selecting a prediction model that can handle null values during feature engineering. Since the null values may have a particular significance for the final prediction, we keep the null values to avoid the impact of null value filling on the model. For example, the values in the price comparison between the recommended hotel and eight competitive hotels are 1, lower than the competitive hotel; 0, the same as the competitive hotel; -1, higher than the competitive hotel; and the missing

value, if any of the three categories is used Either type of padding will cause the model to predict incorrectly, so it needs to be retained. Another example is the missing value of visitor_hist_starrating and visitor_hist_adr_usd, which may be caused by the user being a new user, so the missing value is valuable, and the value of the lost value needs to be fully considered and retained.

2.3 Exploratory data analysis

Correlation analysis Fig.3 below shows the price comparison between the booked hotel and the eight competing hotels comp1_rate ...comp8_rate, while the availability of the hotel in question compared to the competing hotels has a strong correlation between comp1_inv.... .comp8_inv.

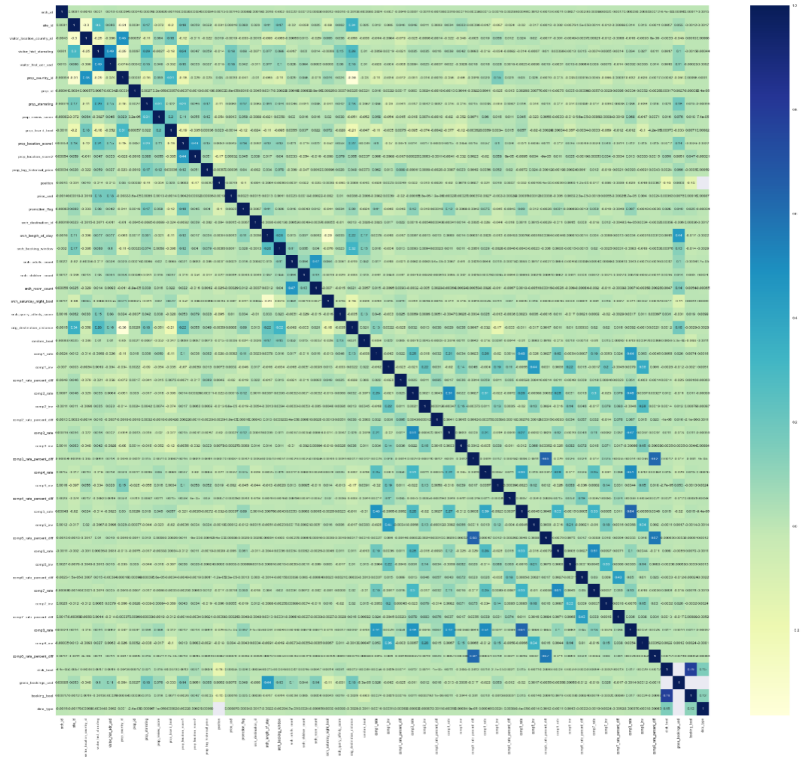

Fig. 3. Correlation heatmap before feature engineering

2.4 Feature engineering

The recommendation system is a process of mining users' interests and preferences based on the user's historical behavior and recommending the user's

favorite hotels to the user in a specific scenario. The features in hotel recommendations can be divided into the following categories: components of the user dimension, which refer to some features related to the user itself; features of the hotel dimension; and the characteristics of the query dimension, which refer to the characteristics of the user in the query and reservation process, etc. To find the most compelling features among the features, we can obtain features with more obvious physical or statistical significance by transforming the features.

Establishment of competitive hotel information features Through the correlation analysis, it can be seen that there is a strong correlation between the price comparison between the query hotel and the eight competing hotels and the availability of rooms between the query hotel and the eight competing hotels. Therefore, we merge them to reduce the redundancy of features, and it can be observed that comp1_rate...comp8_rate and comp1_inv... comp8_inv all take values of 1,0,-1 and NaN. Where comp1-8_inv is -1 may represent a situation where Expedia has no vacancies while competitors have vacancies. This should not be present in user search results, so it could be addressed by temporarily filling it with NaN. As shown in Fig.4 and Fig.5, the rate of hotel reservations with the same price is higher than the eight competitive hotels. More reservations with lower prices than higher prices indicate that price is an essential factor for inquirers to book and browse. And its correlation is vital, so according to its high, low, same, and NaN classification statistics. In the same way, through the comparison between the browsed hotels and eight competing hotels, it can be seen that in most cases, when the competing hotels have no rooms, the success rate of the hotel being booked and browsed will increase. Therefore, the statistics are classified according to all rooms, no rooms, and NaN. Since the null value may have a particular significance for the final prediction, to avoid the influence of deleting the NaN value on the forecast, the number of null values is counted to retain the invalid value data.

Therefore, new features columns comp_rate_high_count, comp_rate_low_count, comp_rate_same_count, comp_rate_nan_count were statistically generated based on high price, low price, same price, and no data on the comparison data of the queried hotel and eight other competitive hotels respectively. The data is classified and counted according to all rooms, no rooms in competitive hotels, and no data to generate new feature columns comp_inv_nan_count, comp_inv_yes_count, comp_inv_no_count.

Establishment of clicking and booking rate features The click_bool and booking_bool of the original data describe whether the hotel in each query is clicked and booked, but it cannot express the relationship between the hotel's query and booking among all users. Therefore, we group each hotel to count the number of clicks and reservations and divide it by the number of hotel queries to get the hotel's click-through rate and reservation rate.

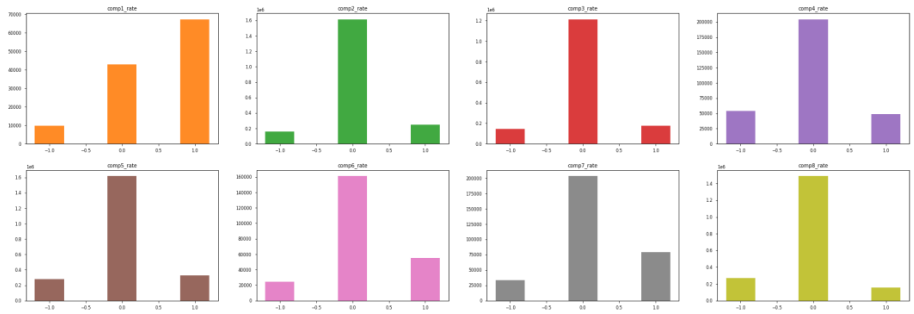

Fig. 4. Histogram of price comparison between hotel reservations and competitive hotels and classification of reservations

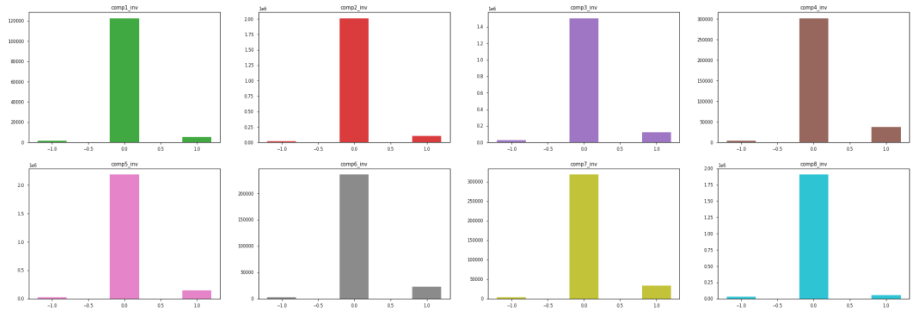

Fig. 5. Histogram of whether the browsed hotel has a room compared with the competitive hotel and the click classification statistics

Establishment of reservation and time features There are peak seasons and low seasons for hotel reservations, which is related to whether it is a weekend or not. The original attribute data save the scheduled date date_time and the number of days from the search date to the check-in srch_booking_window, the arrival date can be calculated, so the planned booking_month, arrival_month can be extracted, and according to the original srch_length_of_stay to calculate whether weekends are included in.

Establishment of differences in user hotel reservations features There is a specific correlation between the star rating and price of the hotel that the user has booked and the hotel star rating and price queried. Therefore, we calculate the star rating difference between visitor_hist_starrating and prop_starratin and the price difference between visitor_hist_adr_usd and price_usd, to obtain the historical difference feature column of user hotel reservations starrating_diff, usd_diff. A new comp1-8_rate_percent_diff feature column is calculated by comparing the price of 8 competitive hotels with comp1-8_rate and the difference comp1-8_rate_percent_diff to combine the price difference. The hotel location scores prop_location_score1 and prop_location_score2 are merged into prop_location_score to reduce homogeneous features.

Establishment of evaluation indexes The original training set has four more attributes than the test set: click_bool, booking_bool, position, gross_booking_usd, click_bool, and booking_bool are the most intuitive responses of users to hotel reservations. In contrast, the position reflects sorting order, and gross_booking_usd is the response to price. Therefore, we successively designed three kinds of labels as evaluation indexes.

Index1 Label: $Label = booking\_bool * 4 + click\_bool$

Index2 Labelmd: $Labelmd = (booking\_bool * 4 + click\_bool) * 100 + position$

Index3 Labelall: $Labelall = (booking\_bool * 4 + click\_bool) * 1000 + round(log(gross\_bookings\_usd)) * 100 + position$

2.5 Second Exploratory data analysis

From Fig.6 below, it can be found that the correlation between label all and each feature value is significantly better than that of a label, which is also proved from the later model training results.

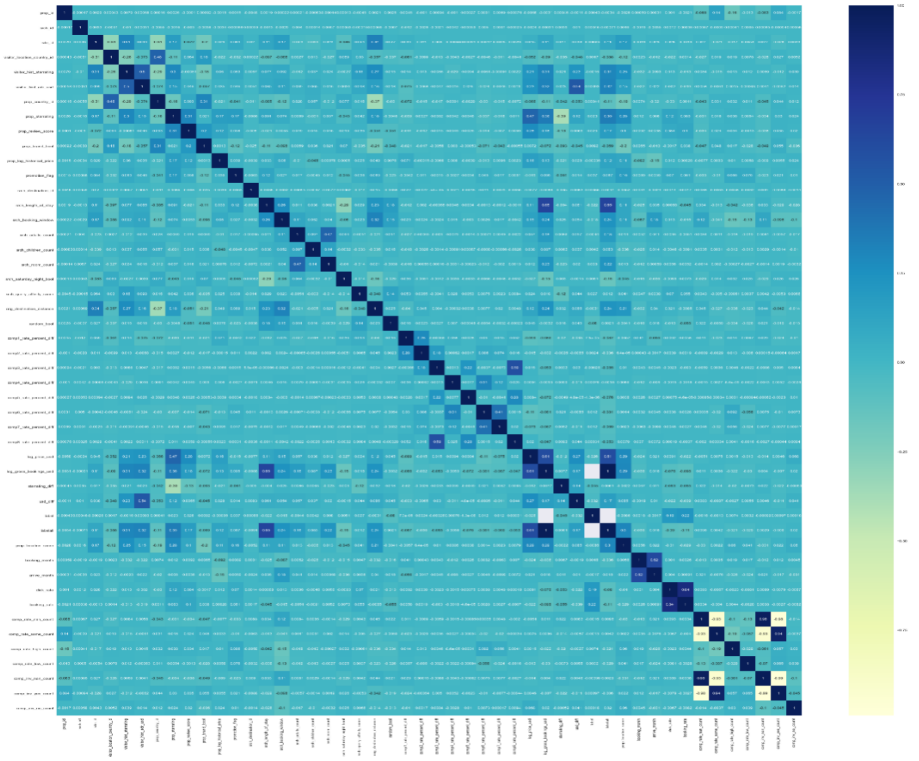

Fig. 6. Correlation heatmap after feature engineering

2.6 Selection of models

This competition is in recommendation systems by predicting user queries and hotel attributes and listing the order of hotels that users are most likely to book. By observing the data information, it can be found that many data columns have NaN values. But the presence of missing values does not mean they lack data information, such as the missing values of visitor_hist_starrating and visitor_hist_adr_usd; the user may be a new user. To fully consider the value of missing values in the prediction model, we compared the support of various models for missing values and selected models that support missing values XGBoost and LightGBM for training and prediction, which also support Learning To Rank.

Learning To Rank Algorithm The ranking has always been the core research problem of information retrieval, which can be mainly divided into the following two categories: The relevance Ranking Model, which ranks documents according to the similarity between query and document; and the Importance Ranking Model, which does not consider query but only according to the web page or document The graph structure between documents is used to judge the authority of documents [11]. Learning To Rank (LTR) is a ranking method of Supervised Learning, which is easy to integrate various features and has a set of mature theories to solve problems such as sparsity and overfitting. Commonly used LTRs are divided into three types: PointWise, PairWise, and ListWise [12].

XGBoost Model XGBoost (eXtreme Gradient Boosting) is an optimized distributed gradient boosting library that implements machine learning algorithms based on the Gradient Boosting framework. Chen Tianqi formally proposed it in the paper "XGBoost: A Scalable Tree Boosting System" in 2016 [13]. XGBoost is an improvement on the gradient boosting algorithm. Newton's method is used to solve the extreme value of the loss function, and the loss function Taylor is expanded to the second order [14]. In addition, a regularization term is added to the loss function. The objective function during training consists of two parts, the first part is the loss of the gradient boosting algorithm, and the second part is the regularization term. Its advantages are fast speed, sound effects, can handle large-scale data, support multiple languages, and support custom loss functions; but its disadvantages are that the algorithm has too many parameters and complex parameter adjustment, which is not suitable for processing ultra-high-dimensional feature data [15].

LightGBM model LightGBM is a scalable machine learning system launched by Microsoft in 2017 [16]. A distributed gradient boosting framework is based on the gradient boosting decision tree (GBDT) algorithm. To meet the needs of shortening the model calculation time, the design ideas of LightGBM mainly focus on reducing the use of data on memory and computing performance and

reducing the communication cost of multi-machine parallel computing. Its advantages are that it is simple and easy to use and provides mainstream Python++language interfaces; it is efficient and scalable, it is efficient, fast, and highly accurate when processing large-scale data sets, and it does not require high hardware resources such as memory; robustness Strong, compared with deep learning models, it can achieve similar results without fine-tuning parameters; it directly supports missing values and category features, and does not require additional special processing of data. However, its disadvantage is that it cannot model the spatiotemporal position relative to the deep learning model and cannot sufficiently capture high-dimensional data such as images, speech, and text [17].

2.7 Division of training set and validation set

Since the problem of learning to sort involves a user query, the data in the dataset is related to the query srch_id, and the same srch_id query cannot be divided into different training sets and verification sets, so it needs to be divided into groups. To ensure that the data of the same group are divided together, the data is divided into a training set, validation set, and testing set.

2.8 Evaluation of the model

This task uses NDCG (Normalized Discounted Cumulative Gain) for accuracy evaluation, which is an indicator that comprehensively considers the relationship between the model sorting result and the actual sequence and is also the most commonly used indicator to measure the sorting effect.

3 Results

3.1 XGBoost and LightGBM model training

Comparison of training effects between XGBoost and LightGBM We use the same feature data and labels, respectively, to compare the effects with XGBoost and LightGBM for training. In selecting parameters, the learner parameter booster of XGBoost and the learner parameter boosting_type of LightGBM both use tree models. The learning target parameter objective The rank type is selected, the metric parameters of the metric standard are all NDCG, the initial learning rate is set to 0.05, and the maximum depth is 6. The corresponding num of leaves in LightGBM is set as 31.

Under the initial settings of several parameters above, in times of local training, the NDCG of XGBoost is around 0.529, while that of LightGBM is around 0.507, making XGBoost better than LightGBM. In terms of training time, XGBoost takes 1m16s, while LightGBM takes 34s, which is less than 1/2 of XGBoost's time. However, after uploading the results to Kaggle, the evaluation score of LightGBM was 0.338 compared to 0.315 for XGBoost, which was relatively better and faster.

Comparison of training effects of different evaluation indexes The original training set has four more attributes than the test set: click_bool, booking_bool, position, gross_booking_usd. To compare the influence of different evaluation indexes on the prediction results, we use the same feature data, the same parameters, and the same model algorithm to train the results. The evaluation indicators are compared with the predicted results. Since LightGBM has an error in Label all training, we use XGBoost for training and prediction. In the evaluation index Label design, the click_bool and booking_bool, which most intuitively represent the user's hotel booking behavior, are used as the Label. The location factor is gradually added to the index labeled. The price factor gross_booking_usd is used as the index Labelall.

During the training process of the training set and the verification set, the NDCG of Label, Labeled, and Label is all 0.53, 0.84, and 0.88, respectively, and the training time does not change much. The NDCG of Kaggle's pre-evaluation scores Label, Labeled, and Label all are 0.315, 0.134, and 0.089, respectively. It can be seen that the addition of the missing position and gross_booking_usd indicators in the test set has dramatically improved the fit of the model training. But it also reduces the model's generalization ability, so the opposite effect is obtained in the test set.

3.2 Analysis of feature significance

To understand the degree of correlation between features and targets, reduce the complexity of the model, improve the prediction model, and increase the interpretability and generalization ability of the model, we have tried a variety of methods to analyze the significance of features.

Variance analysis in features ANOVA is the essential feature selection method. A feature with a more considerable variance is considered more functional. In contrast, apart from a minor conflict is deemed not to affect model training because the value of each sample in this feature is close.

Gradient boosted decision trees Gradient boosted decision tree prediction models provide estimates of feature significance during training, indicating the usefulness or value of each feature in building grown decision trees within the model. The more attributes used to make critical decisions using a decision tree, the higher their relative importance.

For example, when using the XGBoost model for training to get a feature importance histogram. For the importance of 44 features, 23 features were selected according to the median importance of weight and gain parameters for model training. The training score is 0.526, and the Kaggle score is 0.31108. There is a slight decrease in all of them, indicating that the filtered features have a specific contribution to the model.

Interpretable Model SHAP Based on the game theory interpretable model SHAP, the influence of feature importance on the model is analyzed. Also, take the XGBoost as an example for exploring the effect. The horizontal axis of the feature ranking graph is the shape value, and the vertical axis is the feature value. Each point corresponds to a sample. The sample size is stacked vertically, and the color represents the feature value. For example, the red value of click_rate is high, and the SHAP value is distributed on the negative semi-axis, indicating that it harms the prediction; the blue value is low, and the SHAP value is distributed on the positive semi-axis, which has a positive impact on the forecast.

From Fig.7, the prop_location_score, prop_starrating, booking_rate, and random_boolde of the sample are blue arrows to the left, which harm the prediction. promotion_flag, click_rate, porp_country_id, prop_review_score are red arrows to the right, which positively affect prognosis. Thirty-seven of these features have only a 0.01 impact. We retain 12 features for retraining according to their contribution to the model. The training score is 0.527, and the Kaggle score is 0.31270. There is a specific decrease in the full-feature dataset training model, but better than the 23 features of the feature selection of the XGBoost framework, indicating that SHAP is better for feature evaluation.

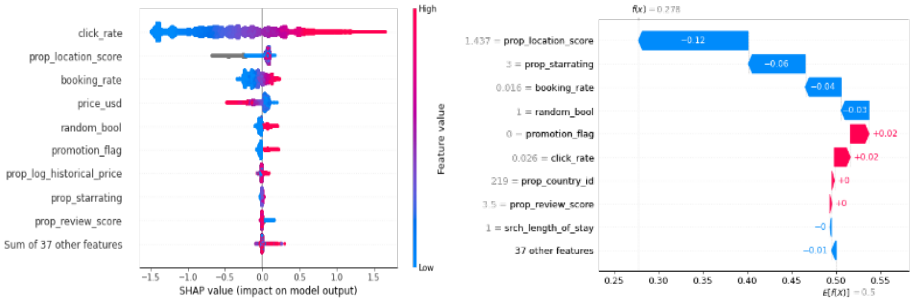

Fig. 7. SHAP feature ranking analysis and single-sample feature analysis diagram

3.3 Model tuning

Model tuning plays an essential role in the relevant data and generalization ability of the model. We realize automatic parameter tuning through the cross-validation method of grid search.

3.4 Final Model Training and Prediction

Due to the above evaluations for models in this task, we finally chose to use the Label as the evaluation index and LightGBM to do training. Predict the test set with the model obtained by training, and sort from high to low according to user queries and predicted rankings. The final hotel prediction ranking results and parameters are as follows in Table.1:

Table 1. Final parameters selection and score

num_iterations	learning_rate	max_depth	num_leaves	min_data_in_leaf	Result_score
900	0.06	5	31	30	0.36338

4 Discussion

By solving the problems encountered in this task, we further understand the data mining process. For example, at the beginning of data reading and processing, 10 million pieces of data are encountered, the reading speed is slow, and the memory consumption is significant. The processing speed of each run is slow, resulting in memory overflow and training freezes. By querying the data, we speed up the reading of data through segmented reading and then merge and test the reading speed of data files in different formats. The problem is that CSV files are slow but have good compatibility. For issues that occupy a large amount of memory and take a long time to process, we have adopted two methods: one is to define appropriate data types through EDA when reading files to reduce memory usage; the other is to write data type conversion functions, Type conversion is performed according to the maximum and minimum values of the data, thereby reducing the memory footprint of the data.

When dealing with missing values and normalizing, some models and algorithms can directly deal with missing values, such as LightGBM and XGBoost. For normalization, since the tree-based model does not care about the variable's value but only cares about the distribution of the variable and the conditional probability between the variables, we can omit the normalization process during data preprocessing.

In addition, we also have a preliminary understanding of the machine algorithm Learning To Rank in the recommendation field and use the LightGBM and XGBoost frameworks for task training and prediction. Trial and error were made in evaluating the choice of index labels. We found that after adding the noise indicator, although the training score can be significantly improved during training and validation, it dramatically reduces the model's generalization ability. At the same time, although parameter tuning can improve the model's accuracy, the most critical work to improve the accuracy of the model is feature engineering. Due to time reasons, the feature engineering of this task only tried the weighted mode and did not conduct more tests on other methods such as averaging. In addition, we also experimented with the interpretable model SHAP for feature selection. After reducing the number of model features to 1/3, the accuracy of the model only decreases slightly. To a certain extent, this method is beneficial in reducing the complexity of the model.

References

1. Zhang, S., Yao, L., Sun, A., Tay, Y. Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), **52**(1), 1-38(2019)

2. Hwangbo, H., Kim, Y. S., Cha, K. J. Recommendation system development for fashion retail e-commerce. Electronic Commerce Research and Applications, **28**, 94-101(2018)
3. Choi, J., Lee, H. J., Kim, Y. C. The influence of social presence on customer intention to reuse online recommender systems: The roles of personalization and product type. International Journal of Electronic Commerce, **16**(1), 129-154(2011)
4. Thorat, P. B., Goudar, R. M., Barve, S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. International Journal of Computer Applications, **110**(4), 31-36(2015)
5. Kim, H. N., Ji, A. T., Ha, I., Jo, G. S. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electronic Commerce Research and Applications, **9**(1), 73-83(2010)
6. Konstan, J. A., Riedl, J. Recommender systems: from algorithms to user experience. User modeling and user-adapted interaction, **22**(1), 101-123(2012)
7. Brynjolfsson, E., Hu, Y., Simester, D. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. Management Science, **57**(8), 1373-1386(2011)
8. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S. Facing the cold start problem in recommender systems. Expert Systems with Applications, **41**(4), 2065-2073(2014)
9. Zhu, X., Yang, Y., Chen, G., Medo, M., Tian, H., Cai, S. M. Information filtering based on corrected redundancy-eliminating mass diffusion. Plos one, **12**(7), e0181402(2017)
10. Chaudhari, K., Thakkar, A. A comprehensive survey on travel recommender systems. Archives of Computational Methods in Engineering, **27**(5), 1545-1571(2020).
11. Belter, C. W. A relevance ranking method for citation-based search results. Scientometrics, **112**(2), 731-746(2017)
12. Liu, T. Y. Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval, **3**(3), 225-331(2009)
13. Chen, T., Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 785-794(2016)
14. Shi, R., Xu, X., Li, J., Li, Y. Prediction and analysis of train arrival delay based on XGBoost and Bayesian optimization. Applied Soft Computing, **109**, 107538(2021)
15. Wang, C., Deng, C., Wang, S. Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. Pattern Recognition Letters, **136**, 190-197(2020)
16. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T. Y. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30(2017)
17. Ahmed, I., Kumara, I., Reshadat, V., Kayes, A. S. M., van den Heuvel, W. J., Tamburri, D. A. (2021). Travel Time Prediction and Explanation with Spatio-Temporal Features: A Comparative Study. Electronics, 11(1), 106.