Predicting hotel booking by users internet search

Group 57: Xinyu Hu (2691175), Christophe Meijer (2585818), and Andrei Udriste (2712179)

Vrije Universiteit Amsterdam

1 Instruction

Internet shopping is preferred over physical store shopping the last years. Significant influences on this change are perceived usefulness, increased online experiences and perceived ease [13]. If companies stay behind and don't offer their products online they risk missing plenty of sales volume. The shift in competition between physical locations and the online world is leading to different marketing strategies. Remarkably, online stores are perceived as having competitive disadvantages with respect to shipping and handling charges, exchange/refund policy for returns, providing an interesting social or family experience, helpfulness of salespeople, post-purchase service, and uncertainty about getting the right item. The advantages that online stores have in areas such as brand-selection/variety and ease of browsing do not entirely overcome the disadvantages listed above [6]. Also, the advantage of a great range of variety can lead to choice overload, which leads to less qualitative decision making [9].Choice overload might lead consumers to develop search and choice strategies and choice strategies based not only on their preferences and the hotel attributes but also on a new factor, the computer interface. As a company, it can be beneficial to get a good understanding of customers behaviour online to overcome this obstacle. An example of a sector where the world wide web has become a priority to increase sales volume is the hotel industry. Specifically, with the help of intermediary online travel agencies (OTA's) such as booking.com and expedia.com, online hotel booking has become widely popular in the hospitality industry and the share of the revenues generated through online booking has been constantly increasing [3]. Hotel booking is typically considered a high-involvement decision because it occurs infrequently and is a relatively expensive purchase. The advent of OTAs has eased the physical effort of search and comparison shopping for hotels [5].But what factors drive customers to pick a hotel over another? For example, Ert and Fleischer [5] found that the position of the hotels on the website was relevant. Hotels at the top and at the bottom tended to be booked earlier than hotels in the middle of the website. This happened unconsciously as the customers didn't describe the location of the hotels as a reason why they booked it. Another article presented results using time-series data of 56,248 hotel reviews for more than 1000 hotels listed on TripAdvisor. There was a long term significant effect of online consumer review factors on offline hotel popularity when controlling for other hotel characteristics. The amount of available data makes hotel recommendation a popular application for machine learning. The machine learning competition platform Kaggle provides datasets derived from Expedia.com and posts challenges of recommendation of hotels. These datasets often consist of user information that lacks linear structure and the volume of possible prediction classes also increases difficulty [1]. Accordingly, this project tends to overcome these difficulties by applying well-known machine learning algorithms to the dataset and then tweaking as well as combining these algorithms so they produce the correct classifications on the dataset. Our task is to predict what hotel a user is most likely to book considering many online hotel characteristics.

1.1 Related work

Inspiration for this project has been drawn by a competition like this from Kaggle. Several developers have tried to implement different algorithms to solve problems exactly like this. Multiple machine learning techniques were tested simultaneously by Sheny, Wagle and Shaikh [12]. These techniques varied from Naive Bayes, Decision Trees, K nearest Neighbors, Decision trees and logistic regressions. This resulted in low accuracy for most of the algorithms, but after applying Clustering and Ensemble methods accuracy increased. The highest observed was by the Multinomial Logistic Regression, as it handles the numeric data efficiently. Other researchers found that accuracy increased when they kept the predictions under the 150 dollar booking mark. They argued that as hotels get more expensive, other features that the model does not include such as room size, quality of service and other amenities play a bigger role. Strategies as the Lasso and Ridge algorithm are used to apply regularization to help with overfitting. Another algorithm is used by a competitor in the Expedia Kaggle competition. Using K-means and Gaussian distribution to anomaly from each cluster [11].

2 Data

2.1 The data

For this project, Expedia has provided a dataset that includes shopping and purchase data as well as information on price competitiveness. The data are organized around a set of "search result impressions", or the ordered list of hotels that the user sees after they search for a hotel on the Expedia website. In addition to impressions from the existing algorithm, the data contain impressions where the hotels were randomly sorted, to avoid the position bias of the existing algorithm. The user response is provided as a click on a hotel or/and a purchase of a hotel room. Appended to impressions are the following: Hotel characteristics, location attractiveness of hotels, user's aggregate purchase history, competitive online travel agency information. Models will be scored via performance on a hold-out set.

2.2 The approach

For analyzing the data, we tried to preprocess the data. A lot of variables weren't filled in completely, leading to gaps in our dataset. Based on some descriptive statistics we cleaned the dataset. For every model we created a training set and a validation set. The training set was approximately 70 percent of the whole dataset and the validation was based on the other 30 percent. However, the size of the datasets and the train/validation/test split ratios can greatly affect the outcome of the models [10]. A good and allround acceptable ratio is the 70-30 split we used. With less training data, the parameter will have greater variance, with less testing data, the performance statistic will have greater variance. The train/validation split ratio can vary for different models.

2.3 Methods

Baseline In this paper we differentiated four different baseline models. They all try to predict the ranks of the properties belonging to a user search on the likeliness that the property will be booked. The first baseline mode (order) tried to do this by ranking based on the order in which the property is found in the dataset. The second baseline (random) model based the prediction on randomization of user ID and Hotel ID. The other baseline models are a bit more complex. The third baseline (review) returns the ranking based on the review score for each property, from highest to the lowest. The fourth baseline (price) returns the ranking based on the price of the property, from lowest to highest. We expect that the review baseline will be the most accurate, since there is a substantial amount of literature that demonstrates this [14] [7][4], [**?**]

LambdaRankNN Here ranking is transformed into pairwise classification. Basically, the algorithms consider a pair of items at a single time to come up with a viable ordering of those items before initiating the final order of the entire list. To minimize the number of inversions in ranking, gradients (λ) of the cost are being utilized with respect to the model score. These gradients function as arrows as little arrows attached to each document in the ranked list that indicate the direction we'd like those documents to move.

Decision tree Decision trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The leaves of a tree are the final outcome and the decision nodes are where the data is split. The splitting is based on a set of splitting rules based on classification features. This process is repeated on each derived subset in a recursive manner. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions.

Stochastic gradient descent Stochastic gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately. One iteration of the gradient descent algorithm requires a prediction for each instance in the training dataset. The gradient descent procedure is run but the update to the coefficients is performed for each training instance, rather than at the end of the batch of instances.

XGboost It is sequentially growing decision trees as weak learners and punishing incorrectly predicted samples by assigning a larger weight to them after each round of prediction. This way, the algorithm is learning from previous mistakes. The final prediction is the weighted majority vote. XGboost knows several methods and three of them are gbtree booster, gblinear and DART. Gbtree booster uses version of regression tree as a week learner. Gbtree allows to represent all types of non-linear data well, since no formula is needed which describes the relation between target and input variables. This is an enormous advantage if these relations and interactions are unknown. Gblinear uses (generalized) linear regression with l1 and l2 shrinkage. But since it's an additive model itself, only the combined linear model coefficients are retained. The third model is Gblinear uses (generalized) linear regression with l1 and l2 shrinkage. But since it's an additive process, and since linear regression is an additive model itself, only the combined linear model coefficients are retained. DART adopted dropout method from neural networks to boosted regression trees

Light Gradient Boosting Machine (LGBM) LightGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage. There are several methods to finetune LGBM. Gradient-based One-Side Sampling, or GOSS for short, is a modification to the gradient boosting method that focuses attention on those training examples that result in a larger gradient, in turn speeding up learning and reducing the computational complexity of the method. Dart adds the concept of dropout from deep learning to the Multiple Additive Regression Trees (MART) algorithm, a precursor to gradient boosting decision trees.

2.4 Performance measure

The evaluation metric for this competition is Normalized Discounted Cumulative Gain. NDCG is per query and averaged over all queries. Hotels for each user query are assigned relevance grades as follows: A 5 gets assigned if the user purchased a room at this hotel. A 1 gets assigned if the user clicked through to see more information on this hotel. A 0 gets assigned if the user neither clicked on this hotel nor purchased a room at this hotel. Submissions for each user query are recommended hotels in order from the highest grade (most likely to purchase a hotel room) to the lowest grade (least likely to purchase a hotel room or click on the hotel). The submission gets compared to the actual solution and this leads to the final accuracy score. For some of the methods a Kaggle submission score was not possible instead a cross-validation score and an accuracy score are utilised to estimate value. Cross-validation is often used as a score in applied machine learning to compare and select a model for a given predictive modeling problem, because it is user friendly and has a lower bias than other methods. Accuracy is the ratio of number of correct predictions to the total number of input samples.

3 Results

3.1 Data Understanding

This dataset is about hotel booking prediction. The two most important variables people considered daily about booking hotels are date time and price. It is a common sense that the hotels like to raise their prices on holidays. The plot below is the comparison between the number of booking and the prices in the non-Saturday night vs. Saturday night. It is clear to see that the prices in Sat night are higher than non-Sat night. However, the higher prices would limit the number of bookings. The free market has been playing its role, and this dataset demonstrates its ability to regulate prices and supply and demand.

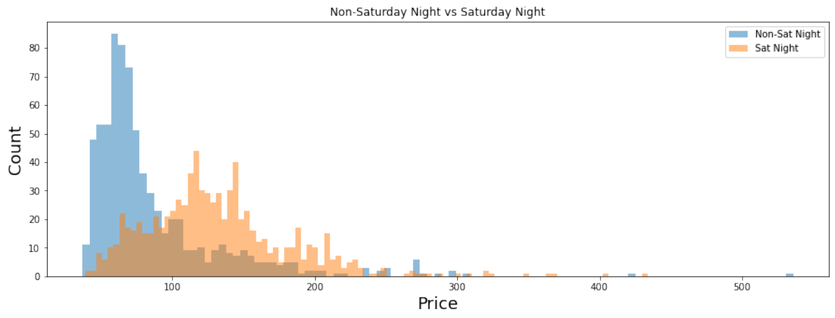

Fig. 1. Sat night vs non-Sat night in train data

3.2 Preprocessing

The first step in any big project that involves a lot of data would be to prepossess the data, more exactly to apply different Exploratory Data Analyses (EDA) techniques on the dataset. The first and most important step in EDA is to check the frequency of missing value of in which columns they appear. If we observe Fig 1 and Fig 2 we can observe that there is a considerable number of columns that have a high frequency of missing values. Because of the high number of missing value we decided to drop those columns. The same approach was done for the test data, we checked the number of missing values and if that was higher than 10% then we drooped the respective columns.

3.3 Data cleaning

After deleting all the columns that have more than 10% of their data missing we observed that there was still one column that had missing values, the *prop_review_score* who had 0.1% of the data missing, this would translate in 7364 missing values for the training data and 7266 missing values for the testing data. The first option that we took into consideration was to delete the rows that contained the specific missing values, but we soon discovered that those rows contained valuable information for the ranking algorithm, mainly those rows contain values for the *booking_bool* and *click_bool* and can not be deleted, because it would negatively impact the model. Taking this into consideration we decided to replace the missing values with the median of the *prop_review_score* column, which is 4.0.

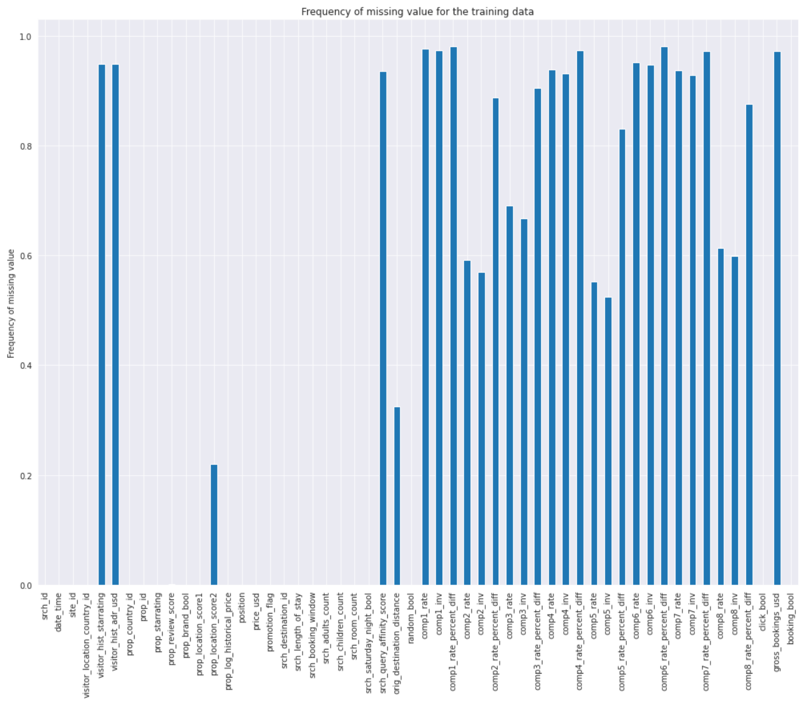

Fig. 2. Frequency of missing values for the training set

One other observation that can be seen is that in both Fig 2 and Fig 3 the frequency of missing values is almost identical. The only noticeable difference between the two graphs is the fact that the training data has an extra column *gross_bookings_usd* that has more than 90% of the data missing, this means that it will be eliminated after the prepossessing has been finished.

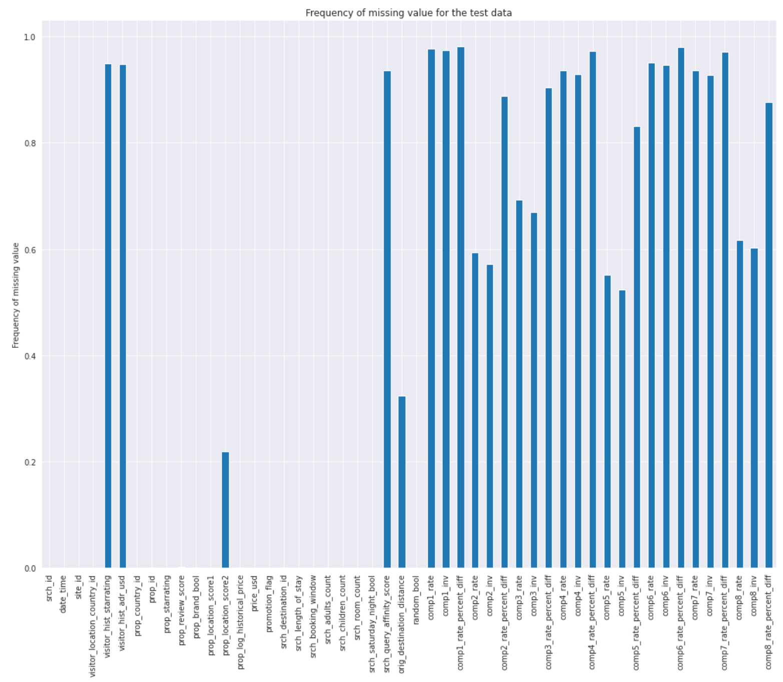

Fig. 3. Frequency of missing values for the test set

The next step in EDA analasys is to create a correlation matrix for the entire training data, to observe if there is any correlation between any of the columns.

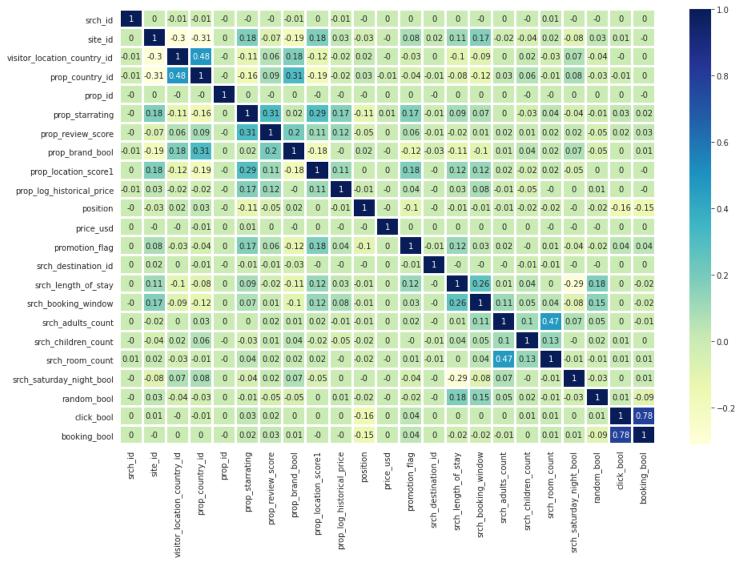

Fig. 4. Correlation matrix for the training data

In Fig 4 is presented the heat-map of the correlation matrix for the training data. The correlation matrix could help us by point out if there is any correlation between any two columns. If there exists any correlation we can, and is big enough we can aggregate/delete one of the respective columns, that we deem less important. If we look at our correlation matrix in Fig 4, we can see that the biggest correlation is between the *click_bool* and *booking_bool*, with a correlation of 0.79. But, because both of those columns are really important in computing the score, it is impossible to aggregate them together or delete one of them. The next biggest correlation values are between columns *srch_room_count*/*srch_adult_count* and *prop_country_id*/*visit_location_country_id* which have a correlation of around 0.47/0.48. But because the correlation was still relatively small and for the rest of the columns was even more small, we decided to not drop or aggregate any columns.

The last step in the prepossessing part is to split the training data in a training and validation data. We do this because we want to test the accuracy of the models on a set of data that the model has not seen before. After the data has been split, the last step is to save the data in specific files, so that is much more easy to access later one in the project.

3.4 Models

In this section we are going to discuss about the models/benchmarks that we used and created. We managed to create 4 benchmarks, 2 simple ones and 2 more complex. Concerning the models, we had two working models, the XG-boost and Light Gradient boosting method, and more models that obtained a great accuracy when using the cross validation method, but not so great NDCG validation.

3.5 Benchmarks

In Fig 5 the results of all the four benchmark models are displayed. Here we see the accuracy score on the training data and the validation data. The price model scored the highest (0.27) of all models on validation data, while the other models showed no significant difference (around 0.25). This was not in line with our expectations suggesting review benchmark should score higher. Another observation would be that the accuracy on the validation data was higher than the accuracy of the training data for all benchmarks. This is weird since we expected that the accuracy will be the same as the one obtained from the training data, since they come from the same distribution. Unfortunately there was no option to obtain the accuracy of the test data without uploading a file on Kaggle. After uploading the file we obtained an accuracy of 0.2, which is lower than the expected accuracy of 0.27, this would indicate that our method of evaluation was a little inaccurate.

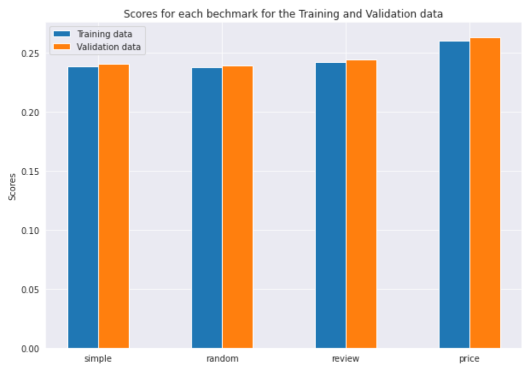

Fig. 5. Benchmark scores

3.6 Light Gradient Boosting Machine (LGBM)

In Fig 6 the results of all three different accuracyes obtained from three different methods of LGBM are displayed. Here we see the accuracy score is the highest using the GBDT method, obtaining a score of 32.17%, while the lowest score is obtained using the DART method (31.57%). One big observation would be the fact that all the methods have almost a similar score, so there is not a real advantage in using one method over the other. The last observation that could be seen, would that the model returns a better accuracy over all of the benchmarks that we implemented, so there is a real advantage in trying to implement the model. The LGBM library uses the LabbdaRank method as it's main algorithm. Parameters of the algorithm used are objective='lambdarank', n_estimators=100, learning_rate=0.1, max_position=5, random_state=69, seed=69, boosting=['gbdt', 'goss', 'dart']. The parameters related to numbers are chosen randomly. The non-numeric parameters are selected according to the model requirements. The LambdaRank solves the correlation order problem in sorting by using a probabilistic model [2].

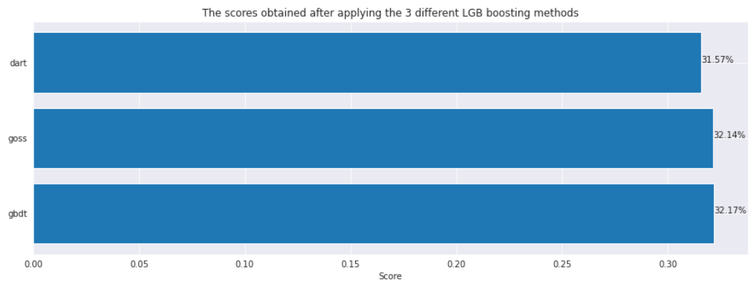

Fig. 6. Light Gradient Boost Machine scores

3.7 XGboost

In Fig 7 the results of all three different methods of XGboost on training and validation data are displayed. Here we see the accuracy score is the highest in the GBDtree (32.19%) and method and DART (32.21%) and lowest in GBlinear (27.14 %). The first observation that it can be observed is that one of the methods has a much smaller accuracy score than the other two methods. The GBlinear method has an accuracy almost equal with the price baseline, but this was expected, since it uses a linear model to predict the ranking, which is worst compared with the other methods (GBDtree - tree, DART - gradient descent). For the final model we decided to use the XGboost model with a GBDtree boost method and the parameter indicated below, because it had the best accuracy using our evaluation method. After uploading the data file on Kaggle we expected to obtain an accuracy of 0.27, since we already experience the difference between our evaluation method and the one from Kaggle. But we obtained a different result, to be more precise we obtained a score of 0.31, which is much more close to the score predicted by our evaluation method. XGboost ranking method is based on LambdaMart algorithm. Parameters of the algorithm used are booster=['gbtree', 'gblinear', 'dart'], objective='rank:pairwise', random_state=42, learning_rate=0.1, colsample_bytree=0.9, max_depth=6, n_estimators=100, subsample=0.75. Again, the numerical parameters were chosen by random. Pairwise is a learning to rank algorithm. Its principle is to select a single training target and to compute the gradient for other targets in the relevant order [8].

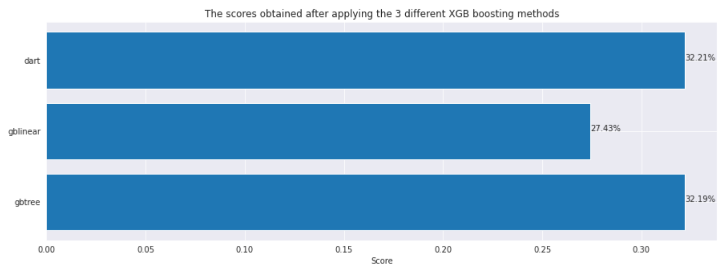

Fig. 7. XGboost scores

3.8 Other algorithms

There are several classification algorithms that were used to create submission files, but they are not working properly. It the below table shows that they all got pretty scores in cross validation and accuracy tests which are almost all above 0.9. However, the scores kaggle systems given is around 0.02. The reason behind this phenomenon is that the split method used by these algorithms is based on sklearn, and the default value is set to 0.25, so that the resulting data set for testing is much smaller than the original data set.

Algorithm	cross validation score	accuracy score
Gaussian Naive Bayes	0.9394070717424012	0.9392
Decision Tree	0.9055990121858386	0.8986
K Nearest Neighbours	0.9546007084929241	0.9538
Stochastic Gradient Descent	0.9532011052969137	0.9538
Logistic Regression	0.9548003092913273	0.9538

4 Conclusion

Our task was to predict the ranking of different hotels and by doing that to increase the probability that a hotel will be booked in the future. To do that we had to performed a multitude of different machine learning methods on a dataset that includes features about hotel booking predictions. This proved quite difficult since the dataset was quite big and had a lot of missing values, but after processing the data, the task became much more easier. Our results stated that XGboost and Light Gradient boosting method were superior in predicting hotel bookings based on the user search history over other methods or the baselines that we implemented, but both ended very close. Remarkably, algorithms with high cross validation scores and accuracy scores tend to score very low on Kaggle. The explanation as mentioned before is that the split method used by these algorithms is based on sklearn, and the default value is set to 0.25, so that the resulting data set for testing is much smaller than the original data set. Scores can be useful for hotel management or online travel agencies. One can imagine that changes in pre-processing can lead to different results. We tend to choose not to delete features in the dataset that were similar, because they added too much value. Other Exploratory Data Analyses techniques can be used to differentiate between features even more. Nevertheless, we showed we could overcome non-linearity and handle the immense volume of the dataset to a certain degree by applying these machine learning algorithms.

References

1. Arruza, M., Pericich, J., Straka, M.: The automated travel agent: hotel recommendations using machine learning (2016)
2. Burges, Christopher J.C., R.R., Viet Le, Q.: Learning to rank with nonsmooth cost functions
3. Cezar, A., Ögüt, H.: Analyzing conversion rates in online hotel booking. International Journal of Contemporary Hospitality Management (2016)
4. De Pelsmacker, P., Van Tilburg, S., Holthof, C.: Digital marketing strategies, online reviews and hotel performance. International Journal of Hospitality Management **72**, 47–55 (2018)
5. Ert, E., Fleischer, A.: Mere position effect in booking hotels online. Journal of Travel Research **55**(3), 311–321 (2016)
6. Kacen, J.J., Hess, J.D., Chiang, W.y.K.: Bricks or clicks? consumer attitudes toward traditional stores and online stores. Global Economics and Management Review **18**(1), 12–21 (2013)

7. Li, H., Ye, Q., Law, R.: Determinants of customer satisfaction in the hotel industry: An application of online review analysis. Asia Pacific Journal of Tourism Research **18**(7), 784–802 (2013)
8. Li, L., Chandramouli, S.: Learning to rank with xgboost and gpu. Nvidia Developer blog (2020)
9. Nagar, K., Gandotra, P.: Exploring choice overload, internet shopping anxiety, variety seeking and online shopping adoption relationship: Evidence from online fashion stores. Global Business Review **17**(4), 851–869 (2016)
10. Rácz, A., Bajusz, D., Héberger, K.: Effect of dataset size and train/test split ratios in qsar/qspr multiclass classification. Molecules **26**(4), 1111 (2021)
11. SAN, Gandhi, V., MXK: Kaggle competition: Production time series of price anomaly detection. https://www.kaggle.com/nikitsoftweb/production-time-series-of-price-anomaly-detection (2019)
12. Shenoy, G.G., Wagle, M.A., Shaikh, A.: Kaggle competition: Expedia hotel recommendations. arXiv preprint arXiv:1703.02915 (2017)
13. Suki, N.M., Ramayah, T., Suki, N.M.: Internet shopping acceptance. Direct Marketing: An International Journal (2008)
14. Zhao, X.R., Wang, L., Guo, X., Law, R.: The influence of online reviews to online hotel booking intentions. International Journal of Contemporary Hospitality Management (2015)