Assignment 2 - Group 95

Herold, Leonard mail@lherold.me 2681224,
Radhakrishnan, Abijith mail@abijith.net 2667575, and
Dudev, Emil e.dudev@student.vu.nl 2669002

1 Introduction

Information has become ubiquitous in our everyday lives, and the amount of information human beings have to handle has drastically changed over the last thirty years [9]. People have adapted to this with different strategies [10], and studies indicate the implications of these strategies as the top half of a web page receive 80% of the users' attention [O4]. Naturally, businesses have acted upon this and personalizing web pages based on the users' preferences. A prominent example is Netflix's highly personalized user interface, being one of the company's reasons for success [O16]. Netflix achieves this high level of personalization by leveraging its Recommender System that ranks interesting movies by predicting the user interest. Similarly, other businesses rely on predicting such user preference, possibly making the difference between a customer purchasing or looking somewhere else. To catch the user's attention Machine Learning (ML) algorithms trying to predict ranking scores affecting the position of an item to sell. An approach by companies to improve on such predictions is to publicly carry out challenges, such as the often-discussed Netflix Prize that awarded a one million dollar reward for the team that is able to predict a user's movie rating best.

This paper relates to a previously carried out competition that comprised ranking hotels on the Expedia Inc. booking portals [O3]. Specifically, this work was carried out in an offshoot of the original competition being the Data Mining Course at Vrije Universiteit Amsterdam 2020. This paper aims at highlighting the process of using Data Mining (DM) to create ML models that Learn To Rank (LTR) of hotels that are most likely to be booked by a user.

We follow the steps of the Cross Industry Standard Process for Data Mining (CRISP-DM) that apply to the scope of this DM problem, and consequently, this paper is structured according to this process. First, a detailed assessment of the task, including the previous work done on this dataset, is discussed in Section 2. Next, Section 3 presents the underlying dataset characteristics and discusses interesting and relevant findings for the task. Based on this, the data preparation and feature engineering activities are discussed in Section 4, shedding light on the rationale behind the added features. Subsequently, the models used are discussed as well as the models' tuning for the ranking task in Section 5. This is followed by an evaluation in Section 6, displaying the performance of the model. Lastly, the work's final results are discussed in Section 7, and Section 8 concludes the work as a whole and covers the lessons learned.

2 Task 1: Business Understanding and Related Work

This section covers the two main aspects to gain a thorough understanding of the task given in the assignment. First, the activities around CRISP-DM's *Business Understanding* are discussed. Next, the *related work*, comprising the work and material found regarding the original Personalize Expedia Hotel Searches Kaggle challenge, is presented.

Following CRISP-DM guidelines, the first step comprised gathering more information about the context of the task. As the challenge given for this work is an offshoot of a previous Kaggle Challenge, it was decided to focus on both, the original challenge and the academic adaptation. If not mentioned otherwise, facts apply to both challenges and differences identified are highlighted accordingly.

Expedia Inc.'s Personalize Expedia Hostel Searches challenge, which was carried out seven years ago, is like the academic adaptation hosted on Kaggle. Kaggle is a data science community platform that allows hosting challenges online [O3].

It is a common practice for a business to hold such challenges to see new innovative and better approaches to their current practice. Expedia Inc., an American online travel agency, initiated the challenge around the IEEE International Conference on Data Mining 2013. The importance of the challenge can be reasoned in the fact that online travel agencies can gain competitive advantages if they do well in placing the user's favoured hotel options at the top [19]. Logically, Expedia Inc. rewarded 25,000 USD to the person or team that could rank hotel options to maximize bookings of users [O3]. As a result, it can be inferred that CRISP-DM's DM objective can be formulated as follows: *The goal of this DM project is to rank hotels in user searches in order to maximize the chance of booking.* The DM's success criteria are also defined by the Kaggle Challenge but differ slightly in the academic version. While the original challenge evaluated the success by measuring the Normalised Discounted Cumulative Gain (nDCG)@38 score, the challenge of this work used is nDCG@5 scoring. Furthermore, the relevance scores for the items in each search query are identical [O7,O3]. Therefore, when assessing related work, it was acknowledged that reported results for the original challenge might not replicate for the academic version. However, the overall application and steps taken by others should hold for the given task too.

Building on previous findings, a search for related work was conducted using two methods. Specifically, this consisted of a web-based search (i.e. Google), scientific search database (Google Scholar and Scopus). Notably, there were no peer-reviewed articles that directly concerned the original Kaggle Challenge. However, Google Scholar yielded four non-peer reviewed works that directly concerned the challenge. These consisted of students' papers from Stanford University [O5,O8] and from the University of Minnesota [O1] as well as a paper from the third place's participant of the Kaggle competition [O5]. The student papers and the paper of the third place, as it was published on the arXiv.org preprint servers, were handled with the necessary caution, but highlighted interesting approaches. The latter paper also motivated a more thorough search for additional material of the challenge results. As a result, the final presenta-

Table 1: Related Work

Source	Models													Data	
	Linear Regression	Logistic Regression	Random Forest	Naive Bayes	Perceptron Classifier	Deep Neural Network	Gradient Boosting	Factorization Machine	Ranking SVM	LambdaMART	Collaborative Filtering	Outliers	Missing Values
[O19]							✗					✗	✗
[O18]	✗		✗				✗		✗	✗		✗	✗
[O10] and [12]		✗	✗			✗	✗	✗	✗	✗			✗
[O8]		✗	✗	✗					✗				✗
[O5]			✗		✗						✗		✗
[O1]		✗	✗									✗	✗

tions of the first, second and third place (cited in that order) of the competition [O19,O10,O18] could be retrieved. Additional insight was also drawn by evaluating the Kaggle discussion boards (i.e., [O6]). Table 1 shows the overall synthesis of the related work found (excluding the Kaggle discussion boards). Due to the type of related work found, it is important to highlight that the different models might not be an ideal solution. Furthermore, some model choices could not be reconstructed (i.e., Collaborative Filtering), as specific information was missing. However, all papers assessed justified some models (i.e., logistic regression) as baseline models, that were used as a benchmark model. From all work evaluated, it can be concluded that two model approaches indicated the most promising results. These are gradient boosting techniques and the related LambdaMART LTR approach. The research also revealed that analysis regarding outliers and missing values in the following data understanding seemed reasonable, as all sources treated missing values, and half of the work accounted for outliers.

3 Task 2: Data Understanding

After CRISP-DM's business understanding and exploring the related work, this section elaborates on the dataset understanding process and the knowledge inferred. First, the technical setup and general *dataset statistics* are highlighted; next, more detailed analysis in regard to *data distributions* are presented.

We used Google Colab as our computation platform and simple mathematical and statistical tools in Python to aggregate and interpret the data provided. The dataset consists of user features, hotel features and user-behaviour logs from Expedia's 2013 search history, which was used in the Kaggle Competition [O3] mentioned in the section above. User features describe user-specific information such as demography, usage and booking history of the user, while hotel features include geological information, competitor pricing and availability, as well as its corresponding ratings based on location and user feedback. User-behaviour features comprise of how the user interacted with the displayed search results

which include information about whether the user clicked or booked a hotel from the results, including the net price after booking.

Both the training and test dataset are large in size and consist of 4.9 million entries each. Each of the datasets holds the information of almost 200,000 searches and 130,000 properties. The main difference between them being the absence of user-behaviour features ('click_bool', 'booking_bool' and 'gross_bookings_usd') in the test set. To efficiently handle the data, we down-sampled the dataset to 10% of its size based on the search ID and used it for local development of the application. However, all figures given in the subsequent sections are based on the entire data available.

For exploratory data analysis, we chose not to include the test set to avoid optimistically biased performance estimates. This is based on Hastie and Tibshirani [7], who caution that otherwise the cross-validated error will be artificially low if both training and test data are used together. This could result in a model that works well for the given dataset but would fail to perform in real-world applications, which in our case would be the Kaggle challenge.

The remainder of this section describes the detailed findings of the training dataset. For preliminary analysis of the training set, we plotted histograms and box plots against different features in the dataset. Also, a broad overview of the training set's characteristics is given in Table 2. Furthermore, to get an understanding of how the data differs for the various groups, plots were grouped by the low-cardinality features, such as property country and origin site. When it comes to privacy, the data is anonymized and no user, property or their corresponding locations can be directly identified. This led to the (positive) outcome of our work being data-driven and unbiased.

Table 2: High-Level Analysis of Training Set

Characteristics	
Total Rows	4,958,347
Search Records	199,795
Properties (Hotels)	129,113
Search Destinations	18,127
Countries	210
Websites	34
Price Range	0 to $19 million
Time Period	1-11-2012 to 30-06-2013

With respect to data availability, we observed that most of the competitor pricing information is missing (Figure 1). The sparsity proportions for competitor data ranges between 52% and 98%. Similarly, features for historical user data are also missing, and no sane defaults can be assumed. Consequently, the only features with missing data, which can be filled in are *prop_review_score* and *prop_location_score*2 (by either a minimum review score or an average), with less than 0.15% and 22% of the properties falling in this case, respectively.

Furthermore, we checked for outliers in both numerical and categorical features of the dataset and found some protuberant values, especially for *price_usd*. We noticed that several properties have unrealistic values as their booking price (highest with $19 million and the lowest with zero).

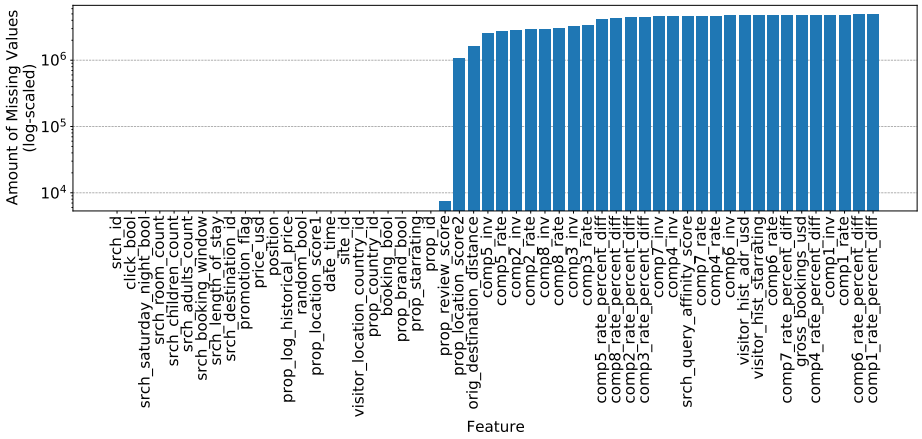


Fig. 1: Dataset sparsity; most sparse features are related to competitor data.

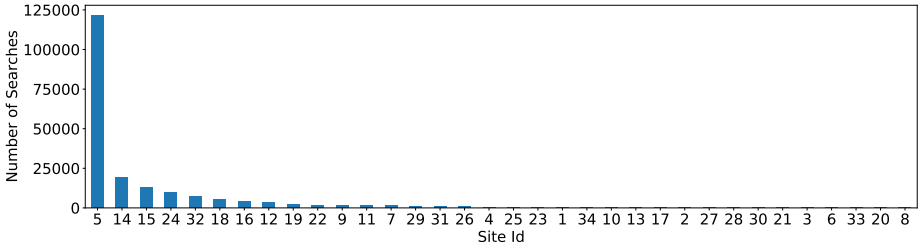


Fig. 2: Search query distribution over websites. It follows Zipf's law.

When it comes to location data, we observed a significant skew showing the characteristics of a Zipfian distribution (Figure 2). To be precise, 61% of the properties are located in a single country, and similarly, 62% of the searches originated from the Expedia website with id 5 (which we believe is an indicator of the user's origin country). This leads us to look into models which can deal with the class imbalance problem.

In regards to user behaviour, we looked at the property book and click data (see Figure 3. We observed that 93% of searches experience only a single property click, while 99% of them consist of less than five clicks. Additionally, we noticed that all of the search results have at least one click on its property and 69.3% of it leads to a booking. Even though

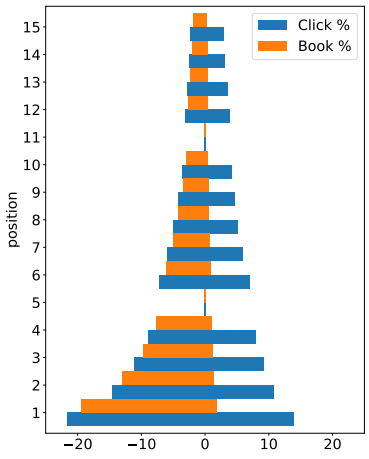


Fig. 3: Percentage of clicks and bookings per position ranking for ordered listing (left) and random listing (right). The position bias is evident.

we observed strong position bias for clicked properties in both randomly-listed as well as non-randomly listed searches, the clicks that got translated to a booking were only 12.85% for the former compared to the 93.9% of the latter.

This observation demonstrates the significance of position bias and it further shows that most users would open a single property and would then decide whether to book it or not, effectively illustrating the importance of presenting the user with the best matches.

After our comprehensive review of the given datasets, we concluded that the data set is well fitted to investigate the impact of rankings even though it has its limitations, such as the presence of outliers and missing values. These limitations will be handled in the next step.

4 Task 3: Data Preparation And Feature Engineering

After presenting the background of the dataset in the previous two sections, this section covers the feature engineering and preprocessing steps. In particular, it is highlighted how and on which rationale a *replicable process for feature engineering and predicting the final query rankings* is established. First, a general overview of the setup is given, and subsequently, individual components are elaborated. Furthermore, it is also shown on which *basis the dependent and independent variables are analyzed*; results of this analysis are presented in the subsequent Section 6.

To facilitate reproducible and comparable iterations in feature engineering, parameter search spaces and model development, a unified pipeline was implemented. This enabled swift iterations and straightforward comparisons between different models versions and prediction results. The pipeline (c.f. 4) is separated in four major steps, including five preprocessing tasks, feature diagnostics, a cross-validated hyper-parameter search, and train and prediction phase followed with steps to prepare a Kaggle submission and data to generate model explanations. The pipeline is implemented to allow development on a 10% random sample of the data locally, which is necessary due to hardware resource limitations, and run on Google Colab using the entire datasets.

Reproducible results of the pipeline are established in two ways. First, the notebook, including the outputs, and additional source files are versioned using git, which enabled us to implement single features and maintain different development branches with low effort. Second, the raw training and test set are the

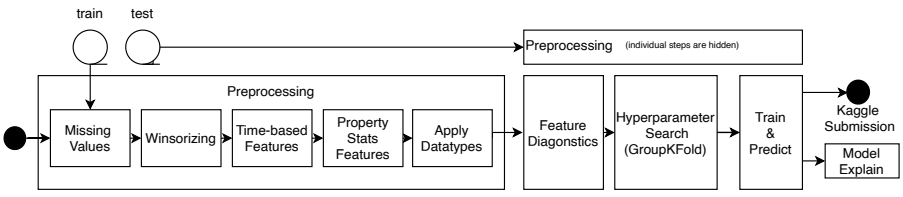


Fig. 4: Steps of The Unified Pipeline that allowed swift iterations and reproducible results.

input for the pipeline preprocessing that is applied for each run. Furthermore, all random states, specifically the GroupKFold and model seeds, are pre-set and thus yield the same results over repeated runs. The GroupKFold ensures that entire searches are preserved, instead of a KFold mechanism, which could split search queries into different folds. Apart from easing the comparison of different model scores based on nDCG@5, the pipeline also enabled us to determine the difference between the output files. This helped us to prevent submission of identical submission files to Kaggle.

Missing values for the features *prop_review_score* and *prop_location_score*2 are inferred based on the lower quantile, acting as a baseline score, per property country. For other features, as reasoned in the data understanding, no missing values are replaced. *Winsorizing* is used to treat outliers for the *price_usd*, and other features were kept as is.

In regard to *time-based* and *property-based* features engineering, several approaches were thought of. The time-based features combined raw features into new features. For example, the day of the week for check-in and check-out were implemented based on the features *srch_booking_window*, *srch_length_of_stay* and *date_time*. For property-based feature engineering, numerical features such as *prop_starrating*, *prop_location_score*1, *prop_location_score*2, *price_usd* etc. were used to build features specific to the property under consideration. The mean, median and standard deviation were calculated for these numerical features. Features such as $history\_price\_diff\_$ and $srch\_id\_max\_price\_diff\_$ were added to denote the difference between the previous price of the property, the maximum price of a property for the given set of search results and the current price respectively.

The final step of the preprocessing is to apply specific datatypes to the dataset. This was not only required to indicate categorical features to the model framework used (Scikit-learn, LightGBM), but also reduced the memory size required to during training. Applying datatypes to the dataset reduced the memory footprint by 30%, resulting in total usage of 2,362 MB for the data itself.

After the preprocessing the training dataset, the pipeline visualizes the dataset using a correlation plot to ensure that features added do not exhibit collinearity, which could lead to overfitting and less prediction power of the model. Subsequently, a hyperparameter search using a GroupKFold [O2] split methodology is executed. Based on the model parameters yielded by the cross-validation step, the final model is trained and used to predict the ranks of the training dataset. The result of the pipeline is a Kaggle submission file in case it differs from previous results, and diagnostics data that allow generating model explanations (i.e. analysis regarding the dependent and independent variable analysis).

To enable *analysis between independent and dependent variables*, which is referred to as feature analysis in the remainder of this paper, different methods were explored and tested (i.e. Local interpretable model-agnostic explanations (LIME)[18]). In fact, during our research for interpretable ML models, the two most promising model-agnostic approaches for feature analysis turned out to be LIME and SHapley Additive exPlanations (SHAP)[14]. Both had their advan-

tages and disadvantages, and although a thorough discussion is beyond the scope of this work, this paragraph elaborates on the rationale behind which approach based on our assessment was selected.

For instance, we found that LIME is a suitable and established method to use. Still, it can suffer from hard to interpret explanatory model summaries [O12]. At the same time Shapley Values, which are the values SHAP is based on, are argued to be the most reliable and built on a solid theoretic foundation (Game Theory) [14]; SHAP itself computes Shapley values in a coalitional game theory to provide explanations [O14]. However, computing Shapley values for explanations can be computationally intensive [O13] to the degree that improved adaptations emerged. Specifically, KernelSHAP and TreeSHAP [13] were proposed as improvements that were inspired by local surrogate models (LIME) and adapted to three-based models (e.g. gradient boosting), respectively.

Overall, we found TreeSHAP [O13] to be a reliable tool for examining feature importance and accumulated local effects and explaining individual predictions. By using the SHAP package [O11] it is possible to describe a prediction by calculating the contribution of each feature to the prediction. Based on the research conducted, we implemented the SHAP feature importance plot and the feature influence plot in the last step *Model Explain*, as shown in Figure 4, allowing us to evaluate the importance and impact of different features in the model. Furthermore, SHAP also provides feature dependence plots to compare interdependence and correlation between different features. A more detailed view of the final model is shown in Section 6 after the model is described in the subsequent section.

5 Task 4: Modelling

This section takes a more in-depth look into the model we have chosen, the preliminary research we conducted, answering the question as to *why we chose this model, as well as how it works* to fit our use case of ranking Expedia properties. Additionally, we take a closer look at the *model's parameters*, the ones we changed from the default values, as well as the ones which we fine-tuned.

The task at hand is a canonical one from Information Retrieval (IR). Years of research in the field, as well as the advance in ML, has led to LTR models to rise to the top in ranking competitions [3, 2, 8]. We have focused on the leading family of gradient boosting LTR models. LambdaMART is the most well-known such model [1], and we have specifically chosen Microsoft's LightGBM gradient boosting implementation, with its support for a LambdaRank-type objective.

In addition, similar libraries which have support for gradient boosting ranking tasks are CatBoost [16] and XGBoost [4]. Both were considered as viable alternatives to LightGBM, yet due to time constraints, they were not empirically tested. Preliminary research [5] suggests that CatBoost was introduced by Yandex to specifically better handle categorical features with the novel idea of using Ordered Target Statistics, however since its release, LightGBM has implemented an alternative, which takes into account the tree-based nature of the

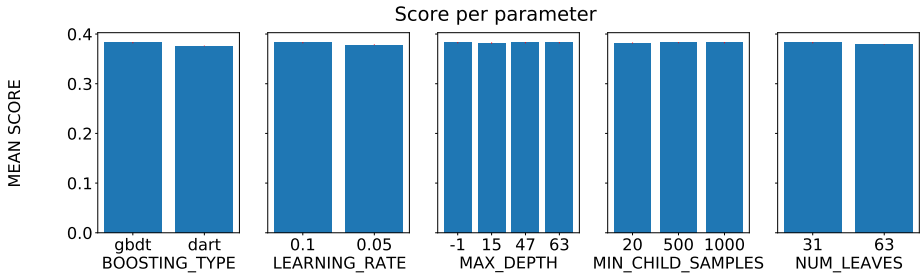


Fig. 5: GroupKFold cross-validation across the tuned hyperparameters.

algorithm and can deal with high volume categorical data [6, 8]. Looking at others' experience [O15,O17], we can conclude that CatBoost can indeed handle categorical features better (which account for about half of our features), yet it has a slightly higher training time. Regarding XGBoost, we know [O9] that it is several times slower than the models, which was a deciding factor. We were led by the thought that fast iteration cycles, which allow for more feature engineering and hyper-parameter tuning, are preferred to extensive gradient boosting searches, which may yield slightly better results. We believe that the gains in performance through feature engineering can outweigh the gains of a exhaustive, but slow XGBoost model.

Being a gradient boosting model, LightGBM builds an ensemble of hierarchical trees, each improving on the results of the previous one [11]. Furthermore, it is able to optimise a LambdaRank-type ListWise LTR objective [1], which is currently the State-of-the-Art, being able to optimise ranking order, and most-importantly giving priority to the highest-attention, top-most, ranks.

Having selected LightGBM as our model of choice, we needed to handle some scenarios not supported by the library. Building upon our previous experience in Data Mining (Assignment 1 of this course), we immediately opted to use scikit-learn [15], mostly due to the thought-out API and ease of use. To LightGBM's scikit wrapper, we had to introduce early stopping, which integrated well with scikit's GridSearch. This allowed us to determine the optimal number of trees to be traversed during inference, such that overfitting does not occur.

Moreover, to strike a balance between high accuracy and not overfitting, we explored parameters like **num_leaves**, **min_child_samples**, **max_depth**, **learning_rate**. However, as can be seen in Figure 5, the difference in the achieved nDCG was minimal and did not yield the expected accuracy boost, despite going over more than 400 hyper-parameter combinations and the whole search taking more than 20 hours to complete. DART [17] was also tested as an alternative to gradient boosted decision trees, as it incorporates dropout and is a good fit for a large dataset. Similarly, no boost in our score was recorded from this change.

We would like to point out the distinction between feature types and the way a model (LightGBM in this case) treats the provided input. Specifically, our highest increase in score was observed when the input features were explicitly

annotated with type information. This allowed the model to distinguish between ordinal and categorical features, impacting on how the trained tree is built.

Bringing all of the above together, for this assignment we used Microsoft's LightGBM library with its ability to optimise rankings. We adjusted the default parameters, where appropriate to account for the specifics of our input data (providing explicit type info), and we conducted a thorough model fine-tuning of the hyper-parameters, including testing out MART with dropouts, along with gradient boosting. We also applied several techniques to prevent overfitting, e.g. early stopping (capping the tree traversal during inference). In the next section we look at the results that our model obtained, and in the following one we further analyse our findings.

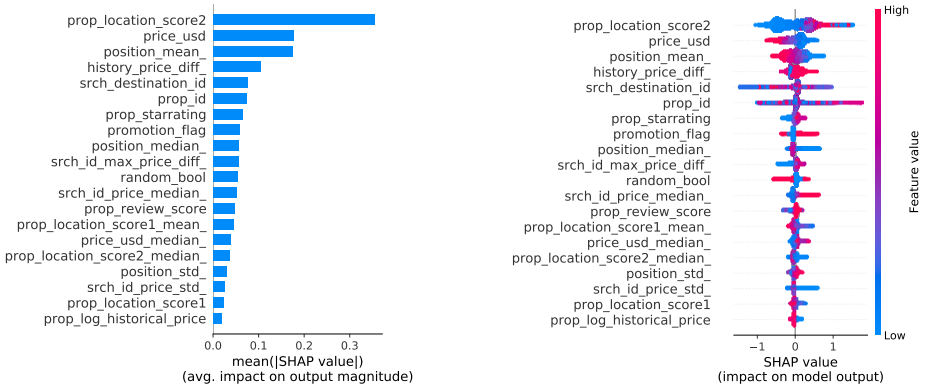
6 Task 4: Evaluation

This section sheds light on the workings of the LightGBM model previously described, and allowing to evaluate the model created. Lundberg and Lee's [13] SHAP method is used to creating model explanations discussed below. It was decided that for this paper that the summary plots of SHAP are the ideal trade-off between detail and essence. SHAP would allow to explaining all features individually, but this seemed not feasible regarding the page limit and not helpful for the overall assessment. SHAP provides contrastive explanations and allowing to compare predictions with the average predictions. The following paragraphs elaborate on the feature importance and on the feature effects.

The feature importance of the model is relevant, as it indicates which features are essential to predict. The intuition behind the Figure 6a is that large Shapley values indicate the feature importance for the model's predicted values, and as such the average of all absolute Shapley values on the testset is plotted [O13]. Inferring from the displayed importance, it can be said that *prop_location_score*2, *price_usd* and *position_mean* are the top three important features changing the ranking score on average by 35.39 %, 17.76% and 17.4% respectively. It can also be seen that, apart from *position_mean* the engineered features impact the predictions on average by around 5% or less.

Further insights can be retrieved from Figure 6b, showing the feature effects of a sample of the predictions as points for each model feature denoted on the y-axis. Each point is colored, indicating if the value of that instance is high or low. The x-axis represents the effect on the prediction; on the negative side of the x-axis points reduce the ranking score predicted by the model, on the positive values feature values increase the ranking score. Taking the feature *prop_location_score*2 as an example, it can be inferred that a high value has a positive impact, while low value of the score has a negative impact on the prediction (e.g., the feature *price_usd* shows the opposite behaviour).

To conclude, using both types of plots, and the additional dependence plots per feature, which were omitted in this report due to the lack of space, allows a thorough and comprehensive explanation of the inner works of the LightGBM model. It also shows, that for our final model Expedia's *prop_localtion_score*2 is



(a) Importance (b) Effects

Fig. 6: SHAP Feature Evaluation Plots

a well-engineered scoring mechanism, which indicates the property's desirability, as in case of the model, is well-suited for ranking the results.

7 Discussion

This final section critically reflects our work and elaborates on aspects we have learned from this assignment. Thus, below technical, team-focused and model-related aspects are discussed.

Regarding our choice among libraries, LightGBM turned out to be beneficial. The maturity and extensive development since its release 2016 contribute to this. Further, its good performance in other online competitions helped us to obtain very good results in the beginning already. Moreover, the tools built within the library and its surrounding ecosystem allowed us to build a unified pipeline easily and later to analyse the model's performance, as well as the feature importance.

Given our sound choice of a boosting model and a ranking optimisation objective, we (as of this writing) ranked on 12 th place, out of a total of 73 groups. Works toward improving the model by adding new features and tweaking parameters, we managed to get nDCG scores higher than 85% of all groups. However, we acknowledge that more could have been achieved. For instance, we only evaluated a single baseline model and used one of the State-of-the-Art libraries, focusing on improving it. This decision was influenced by our previous work on the first assignment, where not all of our models were well-developed. However, as was outlined at the end of Section 5, hyper-parameter tuning for LightGBM was a time-consuming process, which did not yield any major benefits. Having said that, the empirical evaluation of CatBoost and/or XGBoost, retrospectively appear worthwhile.

In addition, another shortcoming of our work is the lack of an in-depth single-case examination. Our analysis focused on deriving conclusions and locating findings in the eternity of the dataset in relation to the rankings we obtain. Given

the class imbalance problem, which was discussed in Section 3, a look into the ranking of individual search queries may have revealed some of the weaknesses of our model. Nonetheless, during the course of this assignment, we have built on our teamwork challenges. We tackled collaboration issues we experienced during the first assignment by formalising our work process, explicitly writing down tasks in a Kanban board and communicating through open pull requests, requiring review approval. Due to COVID-19, we conducted tri-weekly video calls to stay in sync, as well as to plan our next steps. This, along with explicit task deadlines, helped keep setbacks and issues in general accounted for.

Additionally, we deepened our knowledge in DM Techniques, gained an understanding of the LTR sub-field of ML, and contrasted it with traditional ML tasks. Our feature selection process improved upon the previous assignment's one, and with this task, we were able to thoroughly develop an ML model. Furthermore, the importance of model parameters and data representation became evident to us, as well as the need for bulkier machines or sub-sampling in the presence of high-volume data. We also stepped into the field of Explainable Artificial Intelligence (XAI) by looking at the importance of input features.

8 Conclusion

With this work, we investigated the impact of rankings on search and booking decisions in the context of hotel searches in Expedia. We followed CRISP-DM's guidelines which allowed us to go back and forth between different stages easily.

After understanding the business goal and exploring the related works, we translated the business goal into a data mining project objective. Being a well scrutinized and relatively old project, we were able to find enough papers as well as web resources to analyse and explore. For the subsequent steps in CRISP-DM, we used Google Colab as our computation platform and Google Drive as our storage handler for easy cloud computation. This is allowed us to train the model with the complete training set and the size of the dataset no longer posed a hurdle in computation or memory usage. Even though we were not able to perceive any significant pattern for the features in the dataset initially, we uncovered several in our detailed data understanding steps which aided us in the following tasks of data preparation and feature engineering. After carefully considering several models, we chose LightGBM, which is Microsoft's implementation of LambdaMART. This was a crucial step in assuring good results. A detailed evaluation of the results was done using SHAP, a game-theoretic method to interpret the outcome of a ML model, which showed us the importance and influence of different features.

Even though the randomness caused by real-world situations makes it challenging for real-world applications of DM, we learnt that this can be solved by careful curation and understanding of both the data and the problem setting given.

References

[1] Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Tech. Rep. MSR-TR-2010-82 (June 2010), https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

[2] Burges, C., Svore, K., Bennett, P., Pastusiak, A., Wu, Q.: Learning to rank using an ensemble of lambda-gradient models. In: Proceedings of the learning to rank Challenge. pp. 25–35 (2011)

[3] Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Proceedings of the Learning to Rank Challenge. pp. 1–24 (2011)

[4] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system (2016). https://doi.org/10.1145/2939672.2939785

[5] Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363 (2018)

[6] Fisher, W.D.: On grouping for maximum homogeneity. Journal of the American statistical Association **53**(284), 789–798 (1958)

[7] Friedmann, J., Hastie, T., Tibshirani, R. (eds.): Oxford Handbook of Innovation, chap. Data Mining, Inference and Prediction. Springer series in statistics New York (2008)

[8] Gulin, A., Kuralenok, I., Pavlov, D.: Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank. In: Chapelle, O., Chang, Y., Liu, T.Y. (eds.) Proceedings of the Learning to Rank Challenge. Proceedings of Machine Learning Research, vol. 14, pp. 63–76. PMLR, Haifa, Israel (25 Jun 2011), http://proceedings.mlr.press/v14/gulin11a.html

[9] Hilbert, M.: How much information is there in the "information society"? Significance **9**(4), 8–12 (2012)

[10] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010)

[11] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in neural information processing systems. pp. 3146–3154 (2017)

[12] Liu, X., Xu, B., Zhang, Y., Yan, Q., Pang, L., Li, Q., Sun, H., Wang, B.: Combination of diverse ranking models for personalized expedia hotel searches. arXiv preprint arXiv:1311.7679 (2013)

[13] Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)

[14] Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in neural information processing systems. pp. 4765–4774 (2017)

[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[16] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features (2017)

[17] Rashmi, K.V., Gilad-Bachrach, R.: Dart: Dropouts meet multiple additive regression trees. In: AISTATS. pp. 489–497 (2015)

[18] Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)

[19] Schafer, J.B., Konstan, J., Riedl, J.: Recommender systems in e-commerce. In: Proceedings of the 1st ACM conference on Electronic commerce. pp. 158–166 (1999)

Online References

[O1] Agarwal, S., Styles, L., Verma, S.: Learn to rank icdm 2013 challenge ranking hotel search queries, https://www-users.cs.umn.edu/ verma076/letor.pdf, retrieved on 2020-05-13

[O2] lean Documentation, S.: sklearn.model_selection.groupkfold — scikit-learn 0.23.1 documentation (2019), https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupKFold.html, retrieved on 2020-05-19

[O3] Expedia Inc.: Personalize expedia hotel searches - icdm 2013 (2013), https://www.kaggle.com/c/expedia-personalized-sort/overview, retrieved on 2020-05-12

[O4] Fessenden, T.: Scrolling and attention (2018), https://www.nngroup.com/articles/scrolling-and-attention/, retrieved on 2020-05-12

[O5] Jiang, X., Xiao, Y., Li, S.: Personalized expedia hotel searches (2013), http://cs229.stanford.edu/proj2013/XiaoJiangLi-PersonalizedExpediaHotelSearches.pdf, retrieved on 2020-05-13

[O6] Kaggle: Personalize expedia hotel searches - icdm 2013 (2013), https://www.kaggle.com/c/expedia-personalized-sort/discussion/6228#33280, retrieved on 2020-05-13

[O7] Kaggle: 2nd assignment dmt (2020), https://www.kaggle.com/c/2nd-assignment-dmt-2020/overview/evaluation, retrieved on 2020-05-13

[O8] Khantal, N., Kroshilina, V., Maini, D.: Rank hotels on expedia. com to maximize purchases (2013)

[O9] Laurae: Getting the most of xgboost and lightgbm speed: Compiler, cpu pinning (Jan 2018), https://medium.com/data-design/getting-the-most-of-xgboost-and-lightgbm-speed-compiler-cpu-pinning-374c38d82b86

[O10] Liu, X., Xu, B., Zhang, Y., Yan, Q., Pang, L., Li, Q., Sun, H., Wang, B.: Presentation - combination of diverse ranking models for personalized expedia hotel searches (2013), https://www.dropbox.com/sh/5kedakjizgrog0y/AACtU8PFG9BQpK7jOVTcPFR_a/ICDM_2013/5_binghsu.pdf, retrieved on 2020-05-13

[O11] Lundberg, S.M.: lundberg/shap: A game theoretic approach to explain the output of any machine learning model. (2020), https://github.com/slundberg/shap, retrieved on 2020-05-20

[O12] Molnar, C.: 5.7 local surrogate (lime) — interpretable machine learning (2020), https://christophm.github.io/interpretable-ml-book/lime.html#fn37, retrieved on 2020-05-20

[O13] Molnar, C.: Shap (shapley additive explanations) (2020), https://christophm.github.io/interpretable-ml-book/shap.html, retrieved on 2020-05-20

[O14] Molnar, C.: Shapley values (2020), https://christophm.github.io/interpretable-ml-book/shapley.html, retrieved on 2020-05-20

[O15] Nahon, A.: Xgboost, lightgbm or catboost-which boosting algorithm should i use? (Feb 2020), https://medium.com/riskified-technology/xgboost-lightgbm-or-catboost-which-boosting-algorithm-should-i-use-e7fda7bb36bc

[O16] Netflix Inc.: Netflix recommendations: Beyond the 5 stars (part 1) (2012), https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429, retrieved on 2020-05-12

[O17] Swalin, A.: Catboost vs. light gbm vs. xgboost (Jun 2019), https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

[O18] Wang, J., Kalousis, A.: Presentation - personalized expedia hotel searches 2st place (2013), https://www.dropbox.com/sh/5kedakjizgrog0y/ AAAahzn-TeGL_VKvq_jBpDMiCa/ICDM_2013/4_jun_wang.pdf, retrieved on 2020-05-13

[O19] Zhang, O.: Presentation - personalized expedia hotel searches 1st place (2013), https://www.dropbox.com/sh/5kedakjizgrog0y/ AABhKazV866m3uDGa2QYoFsoa/ICDM_2013/3_owen.pdf, retrieved on 2020-05-13