Expedia Hotel Recommendation Model

Victoria Peterson[1 [2871118]], Marvin Frommer[2 [2727106]], and Max Bohle[3 [2728282]]

Vrije University Amsterdam

Introduction

Recommendation of personalized algorithmic content is the basis for much of modern real-world application of data mining and machine learning such that entertainment, productivity, and sales are dictated by these models. One such recommendation system is observable in hotel recommendations that attempt to rank products such that the user might be more likely to book or click on the hotel based on history from similar users. In this case, the prediction goal is a *learning-to-rank* problem where-in hotels must be ranked for each user search query (`srch_id`) according to their likelihood of being chosen.

This process involves the implementation of a data-mining pipeline starting from data analysis, then data cleaning and feature engineering, to finally modeling and evaluation. The approaches made include the implementation of a k-nearest neighbors model as a basic high-cost baseline model followed by a gradient-boosted decision tree model that are designed to optimize NDCG, an evaluation metric that rewards correctly ranked high-relevance results at the top of the list.

1 Business Understanding

In working on our own algorithm for recommending hotels based on user search data we also looked at how others approached similar prediction problems. Specifically, given our wide range of data we were interested in which predictors were most commonly and successfully used by others and what they could share about their approaches that could help us. To do so we discussed our own approach with other teams from the current competition and also looked at the documentation provided by some of the top teams of the original 2013 Kaggle competition [1]

We found that generally a good recipe for success involves the following steps: First, a lot of time and effort should be spend on exploratory data analysis (EDA), data cleaning and building a good understanding of the data composition, quality and purpose. In hand with that is feature engineering and, more generally, working with the data to ensure it is used properly and to its fullest potential. This step is crucial for good predictive algorithms as they can only be as good as the data given to them. Especially with such an extensive dataset it is important to consider each variable carefully to decide whether or not, and in what way to include it. Additionally, a lot of the variables had $> 90\%$ missing values meaning many of them had to be dropped entirely.

Next, is choosing a good algorithm. We see a lot of teams working with learning to rank (machine-learned ranking) models such as LambdaMART and XGBoost. Research shows that the LightGBM LambdaRank Gradient Descent model is more efficient and scaleable than XGBoost without reducing accuracy, thus making it an effective choice in modeling this ranking task [2]

Lastly, generally it is useful to combine and compare multiple models to see which works best with the given problem. So our goal was to perfect the first two steps as much as possible, then test the predictions of a specific model and then try and compare different models to improve our results.

2 Data Understanding

The Expedia Hotel dataset contains a wide variety of attributes that describe both user behavior and hotel characteristics which may be broadly grouped into search-specific attributes like `srch_length_of_stay`, user attributes like `visitor_hist_starrating`, hotel attributes like `prop_starrating`, interaction outcome like `booking_bool`, and competitor features like `comp_rate`. This gives a total of 54 attributes with 4958347 entries in the training data set, and 51 attributes with 4959183 entries in the testing data set, a set of data that excludes the `click_bool`, `booking_bool`, and `gross_bookings_usd` present in the training set. Each row corresponds to a hotel considered during a search, identified by `srch_id` and `prop_id`

Table 1 summarizes key attributes from the training data set, including data types, missing value percentages, uniqueness, and outlier frequency based on interquartile range (IQR).

Attribute	**Type**	**Missing (%)**	**Unique (%)**	**Outlier (IQR)**
`visitor_hist_starrating`	categorical	94.92%	0.01%	NA
`visitor_hist_adr_usd`	numerical	94.89%	0.16%	0.23%
`prop_review_score`	categorical	0.15%	0.5%	NA
`price_usd`	numerical	0.0%	1.54%	5.94%
`orig_destination_distance`	numerical	32.43%	10.7%	8.42%
`srch_booking_window`	numerical	0.0%	0.1%	8.54%
`srch_adults_count`	categorical	0.0%	<0.01%	NA
`prop_starrating`	categorical	0.0%	3.6%	NA
`click_bool`	binary	0.0%	2	–
`booking_bool`	binary	0.0%	2	–
`comp1_rate`–`comp8_rate`	+1/0/-1	>95%	Low	–

Table 1: Selected attributes with data quality indicators.

2.1 Key Observations

– **Highly missing attributes:** Fields such as `visitor_hist_starrating` and `visitor_hist_adr_usd` have over 94% missing values, making imputation

less useful for models that can't handle missing data such as k-nearest neighbors. Models such as LightGBM LambdaRank are capable of using these missing entries in their training process and thus don't require imputation.
- **High-cardinality columns:** Attributes like `orig_destination_distance` and `srch_destination_id` have hundreds of thousands of unique values, benefiting from normalization or clustering for downstream modeling.
- **Outliers:** Price-related columns such as `price_usd` and `visitor_hist_adr_usd` show large IQR outlier counts, which may be removed or normalized using using log transformation and z-scores within attribute groups.
- **Binary targets:** The `click_bool` and `booking_bool` serve as the foundation for constructing the ranking relevance score where bookings are rarer than clicks and thus weighted higher in the relevance function.
- **Competitor columns:** The `compX_rate` and `compX_inv` fields are more than 95% missing and provide similar information without a need to know about specific competitor values, allowing for aggregation into composite features like `sum_comp_rate`.

2.2 Feature Correlation Analysis

Preliminary correlation analysis of the Training data set reveals several noteworthy patterns across features:

- `visitor_location_country_id` and `prop_country_id` show a moderate positive correlation of 0.48, suggesting a tendency for users to search within their own country.
- `prop_country_id` and `orig_destination_distance` have a low-to-moderate negative correlation of -0.37, reflecting a marginal tendency toward shorter travel distances.
- `position` is slightly negatively correlated with both `click_bool` (-0.16) and `booking_bool` (-0.15), supporting the intuition that higher-ranked listings are more likely to be interacted with.
- `log_price` and `price_usd` both have nearly 0.0 correlation with `click_bool` and `booking_bool`, indicating that customer decisions tend to go against the a frugal intuition and instead relies on a more complex relationship of different attributes.
- `prop_starrating` correlates positively with the `log_price` at 0.47, but doesn't correlate with `price_usd` at 0.01, indicating that the expected price increase relationship from a fancier hotel might be captured better by the log of the price than the normal price scale.
- Several `compX_rate` features (e.g., `comp3_rate`, `comp6_rate`) and their corresponding `compX_rate_percent_diff` values are highly correlated (above 0.6), confirming that these attributes share redundant information and can be aggregated.

3 Data Preparation

3.1 Missing Values

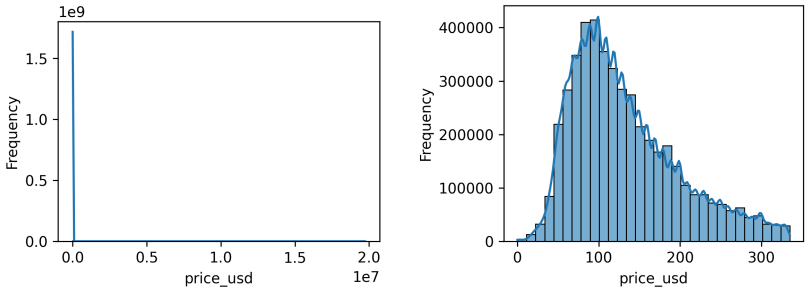
As observed in Table 1, data such as `visitor_hist_starrating` or the competitor data `comp_rate` and `comp_inv` contain a large number of missing entries which may affect model training depending on the algorithm used.

A model such as k-nearest neighbors requires that missing entries be imputed or interpolated with new data based on the aggregate data set, using the median of numerical data or the mode of categorical data to fill the empty spots. KNN imputation may also be used to impute these missing values, but the large number of data entries makes this computationally taxing, requiring down-sampling to a sample data set with similar overall characteristics for imputation.

The imputation of attributes with missing data greater than 5% is both computationally expensive and loses the attribute relationships that allow the model to learn from the data properly. As such, the usage of a model like LightGBM LambdaRank that can learn from the missing data entries becomes convenient in training a ranking model.

3.2 Outliers

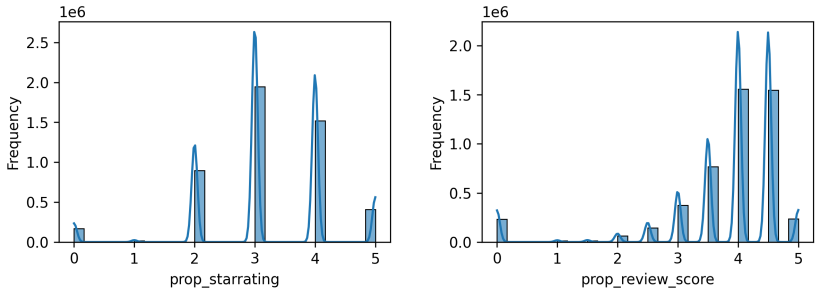
Missing data handling becomes even more important as the IQR outliers of the numerical data are removed and thus require handling based on the model being used. Data such as `price_usd` present in Figure 1 shows how the removal of outliers using the IQR threshold of 1.5 gives more realistic data and allows for either imputation or use as a missing value in the chosen model.



(a) Base `price_usd` (b) Cleaned `price_usd`

Fig. 1: Distribution of `price_usd` frequency of Training data set before and after IQR outlier removal

The removal of IQR outliers may also be applied to numerical data such as `prop_location_score`, `prop_log_historical_price`, `srch_length_of_stay`, and `srch_booking_window`



(a) `prop_starrating` (b) `prop_review_score`

Fig. 2: Distribution of `prop_starrating` and `prop_review_score` categorical data

Certain attributes like `srch_adults_count`, `prop_starrating`, and `prop_review_score` in Figure 2 are saved as numerical ranges, but are limited to a specific range of integers by the user interface inputs of the Expedia website, making them categorical values that don't require outlier removal.

3.3 Label Construction

To represent user preference strength in a ranking context, a `relevance` label may be defined that reflects user behaviors when booking hotels based on the `booking_bool` and `click_bool`.

$$\texttt{relevance} = 5 \times \texttt{booking\_bool} + \texttt{click\_bool}$$

Clicking on hotel recommendations occurs at a rate of 0.044749% while booking a hotel occurs at a rate of 0.027911%, so a weight should be applied to the label based on both the frequency difference between the two, as well as the increased importance of booking compared to just clicking. The label guides supervised learning with models such as LightGBM (using the LambdaRank objective) and the k-nearest neighbors model.

3.4 Data Aggregation

To use the k-nearest neighbors (KNN) algorithm treated later in this report, we needed to transform the data so that each search query (`srch_id`) was represented by a single row. This was necessary because KNN compares complete search queries with each other, not individual hotel listings. Since each query returns multiple hotel options, we aggregated these rows into one per query.

We did this in two main steps. First, we processed the data in chunks to avoid memory issues with large files. Within each chunk, we grouped the data

by `srch_id` and took the mean of the numeric features that we thought would be most useful for the recommendation task. This helped capture the average characteristics of the hotels shown in each search. We also added a count of how many hotel listings were associated with each search, called `hotel_count`.

Although our code also allowed for keeping the first value of some columns, we didn't end up using that feature in this version of the project—only mean-based aggregation was applied.

The following columns were averaged for each `srch_id`:

- `prop_starrating` – average star rating of listed properties
- `prop_review_score` – mean review score across listings
- `prop_brand_bool` – proportion of branded properties
- `prop_location_score1` – average location score
- `price_usd` – average price in USD
- `promotion_flag` – proportion of listings with promotions
- `srch_length_of_stay` – typical stay duration
- `srch_booking_window` – mean booking window in days
- `srch_adults_count`, `srch_children_count`, `srch_room_count` – average group composition

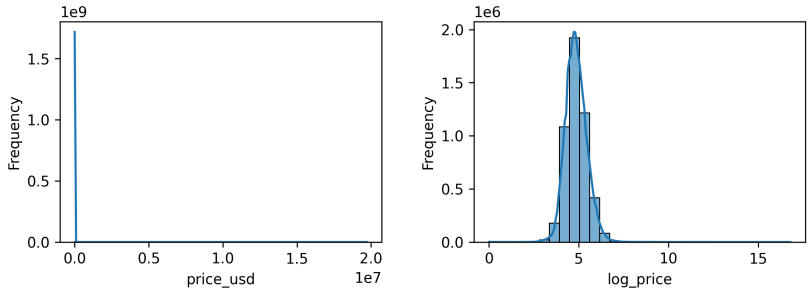
Note that while some of the aggregated features are technically categorical in nature, we treated them as quasi-numerical during aggregation. This approach is reasonable when the categorical values have an inherent order or when they are binary, as the mean can still provide a meaningful summary—for example, representing an average level or a proportion. Treating such variables as numerical simplifies the aggregation process and can still yield useful information for the recommendation task.

After all the chunks were processed, we combined them and re-aggregated the full result to make sure everything was correctly grouped. The final dataset had one row per search query, which made it suitable for use in the KNN model.

3.5 Feature Engineering

The Gradient Descent LightGBM model is less limited by data set size and thus allows for an alternative approach that allows for not only efficient training, but also the addition of new significant features based on the data set.

Such added features provide an effective approach to outlier handling in normalizing skewed attributes using a combination of log transformation and z-score normalization. This compresses large outlier values and stretches smaller ones, producing a more symmetric, bell-shaped distribution. Since the log scale emphasizes relative (percentage) differences over absolute ones, it helps stabilize variance and improve feature scale uniformity across the dataset. This effect is clearly demonstrated in the transformation of `price_usd`, as shown in Figure 3.



(a) `price_usd` (b) `log_price`

Fig. 3: Distribution of `price_usd` frequency before and after log normalization to `log_price`

Additional steps to improve the predictive performance and interpretability of the ranking models involves the construction of statistical, relational, and group-normalized features that express meaningful patterns in the training dataset's user behavior, hotel quality, and search contexts.

The first of such features is the aggregation of the eight competitor rate and inventory columns into summary variables where the original attributes expressed competitor advantage and disadvantage as $+1/-1$ values, allowing for the summation of these values to be an express overall advantage compared to all competitors while reducing sparsity and dimensionality.

- `sum_comp_rate`: Sum of all `compX_rate` columns, indicating how the property compares to competitors in terms of pricing.
- `sum_comp_inv`: Sum of all `compX_inv` columns, reflecting overall competitor availability.

Rank-based features within each search (`srch_id`) help the model understand how each hotel compares to others in the same search context by placing them in an order from most-important to least-important (e.g. Higher hotel reviews are better than low hotel reviews).

- `price_rank`: Rank of the price in ascending order
- `prop_star_rank`: Rank of the property location score in descending order
- `prop_review_score_rank`: Rank of the property star rating in descending order
- `location_score_rank`: Rank of the property review score in descending order

This ranking expresses some personal bias of decision making that places a 5-star hotel above a 4-star hotel in user-desirability, which may not always be an accurate representation of customer choices, so the results of this decision must be closely observed for its impact on the final model.

Some additional features added to the data set include the usage of specific flags for hotel features that point towards the "best" of a specific data attribute within a specific user search. These flag attributes express the importance of low prices and high ratings similar to the ranking attributes, but seek to reinforce the outcomes deemed to be important in this context.

- `is_cheapest`: True if the property has the minimum price in the search.
- `is_best_rated`: True if the property has the highest review score in the search.
- `is_top3_cheapest`, `is_top3_prop_review`: Flags for properties within the top three by price or review score.

To capture global signals about hotel performance, some statistical values were computed to express the importance of certain attribute relationships such as "value-for-money".

- `avg_prop_review`, `avg_prop_star`, `prop_avg_price`: Averages across all occurrences of the same `prop_id`.
- `value_score`: Defined as review score divided by log-transformed price, indicating value-for-money.
- `has_hist_rating`: Binary feature denoting whether a user has historical rating data.

As part of the price-transformation process, additional normalization techniques were applied within individual search IDs and demographic groups to contextualize hotel price further.

- `price_z`, `price_percentile`: Z-score and percentile within a search.
- `relative_to_dest_price`: Price normalized by average price for the destination.
- `price_per_adult`: Price per adult in the search party.

Temporal information is also extracted from the date and time data to identify possible indicators for user decision making based on time-dependent behavior patterns.

- `is_weekend_search`, `is_weekday_search`: Binary indicators based on the day of week.
- `search_hour`: Hour of the day when the search was made.

3.6 Ethics and Geographic Normalization

An important consideration when deploying machine learning algorithms—particularly in real-world recommendation systems—is ensuring ethical treatment of data and users where algorithms may unintentionally learn and amplify societal or economic biases present in the data. It is therefore the responsibility of data-mining practitioners to identify, mitigate, and document these biases during feature engineering and model evaluation.

In this dataset, one clear avenue for potential bias lies in the economic and cultural differences between countries such as cases where some countries consistently have higher hotel prices due to economic stratification. If left uncorrected, these differences could lead to unfair treatment of properties or users from certain regions.

To minimize geographic bias and improve model fairness, group-level normalization may be applied to to key features, computing both ratios to the group mean and Z-scores.

- User country normalization
  - `price_vs_country_avg` — price divided by the average price in the visitor's country.
  - `price_country_zscore` — Z-score of the price within the user's country.
- Hotel country normalization
  - `price_vs_location_avg`, `price_location_zscore`
  - `review_vs_location_avg`, `review_location_zscore`
  - `star_vs_location_avg`, `star_location_zscore`

These features allow the model to evaluate hotels in context, rather than based on raw absolute values, helping to reduce cultural and economic bias across regions.

4 Modeling

4.1 K-Nearest Neighbors Model

The first technique we employed for the recommendation task was an unsupervised method based on k-nearest neighbors (KNN). To do this, we used the per-search aggregated dataset described in Section 3.4. Based on this dataset, we identified the 5 nearest neighbors of each `srch_id`, using Euclidean distance computed over the aggregated numerical features.

After retrieving the nearest neighbors for a given query, we selected all hotel properties that had not yet been shown to the original `srch_id` but had been presented to its neighbors. From this pool of previously unseen hotels, we created a ranking using a simple ratio-based score:

$$\text{score} = \frac{\texttt{prop\_review\_score}}{\texttt{price\_usd}}$$

This metric promotes listings with high review scores and low prices, which are generally considered more attractive to users.

Only one listing per hotel (`prop_id`) was kept to avoid duplication. The output of this step was a recommendation list of new properties for each search query, based entirely on patterns learned from similar past searches.

4.2 Gradient Boosting Decision Tree

The simple K-Nearest Neighbors (KNN) model approach struggled to capture non-linear feature interactions and capturing the structure of ranking tasks, resulting in weak ranking performance and low validation NDCG scores. Furthermore, it was not naturally suited for learning-to-rank tasks, as it focused on similarity in feature space rather than optimizing order relevance within search sessions.

To address these shortcomings, a gradient boosting model using LightGBM with the `lambdarank` objective was implemented, as the approach is specifically tailored for ranking problems. LambdaRank optimizes a loss function that approximates ranking metrics such as NDCG, which is useful in scenarios like Expedia hotel search, where the goal is to correctly order hotel listings within each user query.

The model implementation uses the constructed `relevance` label for ranking and splits the dataset using `GroupKFold`, grouping by `srch_id` to avoid data leakage.

The LightGBM model was configured with the following parameters after an iterative grid search with early stopping on validation NDCG@5 to identify the best performing model using NDCG score.

- **objective:** `lambdarank`
- **metric:** `ndcg`, evaluated at top-5 ranks
- **boosting_type:** `dart` — dropout in boosting improves generalization
- **learning_rate:** 0.05
- **num_leaves:** 32
- **min_data_in_leaf:** 25
- **drop_rate:** 0.1

5 Evaluation

5.1 Evaluation Metric: Normalized Discounted Cumulative Gain

To evaluate the effectiveness of learning-to-rank models, the Normalized Discounted Cumulative Gain is used as a ranking metric that captures both the relevance of retrieved items and their positions in the ranked list.

NDCG@5 is used since users typically only view and interact with the top few results, so evaluating at rank 5 ensures that the model is rewarded for correctly placing relevant listings at the top of the list, which has the most direct impact on user experience and business outcomes.

5.2 K-Nearest Neighbors Model

Using a held-out validation set, we achieved an NDCG@5 score of 0.16760, compared to 0.14834 on the training set indicating that the model is not severely overfitting the training data, but ultimately lacks predictive power. This is not unexpected, as KNN is generally not well-suited for complex ranking or recommendation tasks like the one at hand. The main limitations include:

- **Lack of task-specific learning:** KNN does not optimize for the target metric (e.g., ranking accuracy or booking likelihood), since it does not learn from labeled booking or click data. It simply relies on feature similarity in the aggregated space.
- **Choice of k:** Using a small value like $k = 5$ can make the model overly sensitive to noise or unrepresentative neighbors, especially in a large and diverse dataset. A larger k may better capture general trends.
- **Data sparsity and variability:** Each search query is unique, and the space of possible hotel combinations is large. As a result, the nearest neighbors might not always represent meaningful or relevant user behavior.

Overall, while the KNN approach provides a simple and interpretable baseline, its limitations make it insufficient for high-performance recommendation in a real-world setting. Supervised learning methods that directly model the likelihood of user interactions (clicks, bookings) offer more flexibility and predictive power for this kind of task.

5.3 Gradient Boosting Decision Tree

Compared to the earlier KNN model, the LightGBM LambdaRank implementation significantly improved performance, achieving an NDCG@5 of 0.47939 on the training set and 0.47888 on validation, while the small gap between the two indicates stable generalization and minimal overfitting. The model outputs a `predicted_score` for each hotel listing, which is used to rank hotels within each user search.

To interpret the model's decision-making process, feature importance is analyzed using two standard metrics provided by LightGBM: **gain** and **split**. In Figure 4, gain measures the total improvement in the model's objective function contributed by a feature across all trees, while split counts how frequently a feature is used to make a decision node.

Features `price_z` and `prop_avg_price` are highly ranked in both gain and split, highlighting the significance of normalized price and location metrics in ranking decisions compared to the non-normalized `price_usd`.

Additionally, the `prop_location_score2` dominates both gain and split importance, suggesting that this feature contributes strongly and consistently. Since both `prop_location_score1` and `prop_location_score2` are abstract desirability values, this feature importance expresses an effective algorithmic computation by Expedia themselves.

Features such as `promotion_flag`, `random_bool`, and `prop_log_historical_price` appear more often in splits, suggesting they serve as common branching criteria even if their gain is moderate. In contrast, engineered binary indicators such as `is_top3_cheapest` and `sum_comp_rate` rank lower in both views, suggesting limited influence in the final model.

Overall, the importance plots validate the effectiveness of price and location normalization features and support the model's emphasis on learned economic and positional signals.

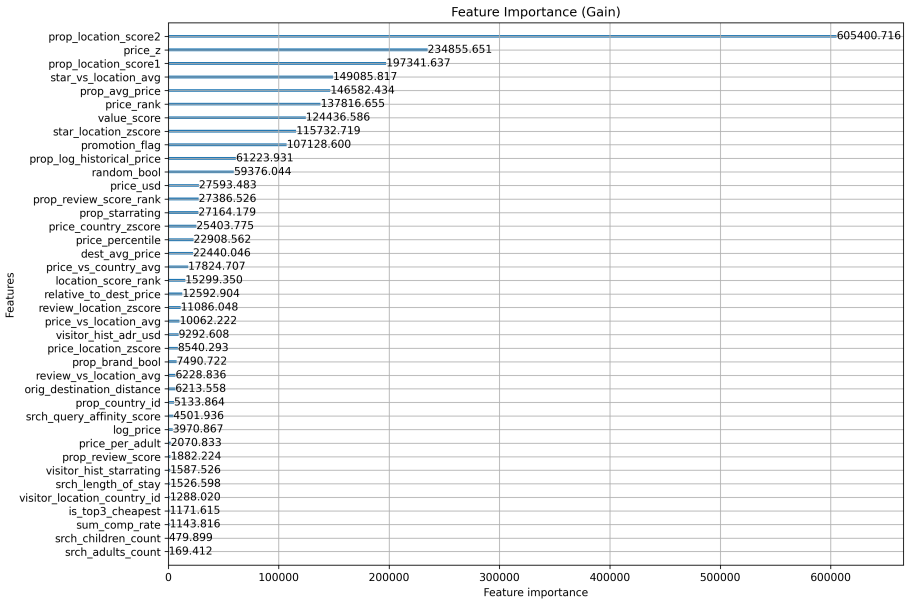


Fig. 4: Feature importance gain measuring the improvement in the NDCG loss function

Fairness via Geographic Normalization.

As part of the efforts to reduce country-specific biases for the non-engineered features `price_usd`, `prop_review_score`, and `prop_starrating`, the model's fairness was assessed by comparing NDCG@5 scores and the corresponding coefficient of variation (CV) across different hotel countries (`prop_country_id`) in Table 2. The CV serves as a proxy for fairness where lower values indicate more consistent ranking performance across regions.

The configuration including all geographic normalization features – which adjust price, review score, and star rating relative to country-level averages and standard deviations – achieved the best overall NDCG@5 on both training (0.51216) and validation (0.48938). It also had one of the lowest variation coefficients (0.4513), indicating a balance between accuracy and fairness.

In contrast, the model trained with no normalization features performed worse in both accuracy and fairness with a validation NDCG@5 of 0.47888 and CV of 0.4557, suggesting that the inclusion of these features helps the model better generalize across geographic regions.

When combined with geographic normalization features that take the average and Z-scores of the attributes, the listwise LightGBM model demonstrated more consistent performance across regions than baseline models, helping reduce location-based ranking bias.

Geographic Normalization Features Included	Training NDCG@5	Validation NDCG@5	Coefficient of Variation
All	0.51216	0.48938	0.4513
None	0.47939	0.47888	0.4557
`star_vs_location_avg` & `star_location_zscore`	0.44379	0.48411	0.4498
`price_vs_location_avg` & `price_location_zscore`	0.46904	0.47078	0.4619
`review_vs_location_avg` & `review_location_zscore`	0.50706	0.44065	0.4607

Table 2: Effect of geographic normalization features on fairness and ranking performance.

6 Discussion

6.1 Feature Engineering

When selecting features to engineer for the model, we decided to create attributes that we believed would mimic customer behavior patterns such as `price_per_adult` and `is_top3_cheapest`, but feature importance analyses in Figure 4 and feature split analysis suggest they had limited impact on model performance. In contrast, statistical features like `price_z` and `prop_avg_price` proved more influential, highlighting the value of incorporating relational and distributional context into the model.

These statistical features, including z-scores, percentiles, and normalized group-based metrics, helped capture complex patterns not evident in raw data. Their inclusion contributed to improved NDCG@5 scores and reduced performance variance across user and region groups.

This contrast underscores the importance of domain-informed statistical transformations in learning-to-rank systems where future work could explore temporal dynamics, user intent modeling, or unsupervised user clustering to further enhance model effectiveness.

6.2 Deployment

The implementation of the model pipeline follows a modular script-based architecture, where core functionality is divided across multiple source files. Scripts import reusable utility functions from dedicated modules such as `csv_utils_basic`, `plotting`, and `gbdt_model`, which helps maintain code clarity and avoids duplication of logic.

Each model run is managed via a main execution script which aligns well with the prototyping phase of the assignment, it presents several challenges for larger-scale deployment. Specifically:

– Scripts are tightly coupled and assume fixed file paths, which limits portability and reuse in different environments.

- Model configuration is hardcoded, making experiments less reproducible or automatable at scale.
- There is no use of a configuration management system (e.g., YAML or CLI arguments), which would improve flexibility.
- Execution is centralized in large monolithic scripts rather than orchestrated through modular pipelines or ML frameworks.

Future improvements could include the adoption of configuration files and pipeline abstraction tools to ensure scalability, maintainability, and reproducibility for deployment in real-world settings.

6.3 Overall

Limitations of our current approach include the lack of user-specific history beyond a single session, lack of features that could be used to improve predictive performance, and reliance on static engineered features. Future improvements may involve hybrid model implementations, personalized embeddings, or incorporating feedback loops to continuously adapt rankings.

7 Conclusion

Exploration of the Expedia Hotel data set used a range of data mining techniques to predict and rank hotel listings based on user interaction data with extensive data preparation and feature engineering tasks to capture both statistical patterns and user behavior.

Through comparative modeling, traditional methods like k-nearest neighbors were found to be limited in performance and scalability, especially on sparse, high-dimensional data. In contrast, the LightGBM LambdaRank Gradient Descent model achieved strong ranking performance, with a validation NDCG@5 score of 0.489 with minimal overfitting between training and validation sets.

Statistical transformations such as log and z-score normalization alongside relational features like price rank and value scores were key contributors to model performance and point towards possible improvements through the addition of similar features. Moreover, by incorporating geographic normalization features, bias was reduced across countries while also enhancing model effectiveness, as reflected in both NDCG scores and reduced performance variance across property regions.

The final results highlighted the critical role of both model selection and hyperparameter tuning in achieving strong performance. More notably, they demonstrated the substantial impact of feature engineering, as many of the engineered attributes outperformed baseline features in importance rankings, contributing significantly to the model's overall effectiveness.

References

1. "Personalize Expedia Hotel Searches - ICDM 2013." Kaggle, `https://www.kaggle.com/c/expedia-personalized-sort`. Last accessed 20 May 2025
2. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 3149–3157.