

---

# Hotel Recommendation

---

**Kaige Yang**  
University College London  
Kaige.yang.11@ucl.ac.uk

## 1 Work Flow

This project we follow the following work flow.

1. Project Overview
2. Data Loading
3. Data Understanding
4. Brain Storming on How to Solve the Problem
5. Data Cleaning
6. Data Transformation
7. Baseline Model
8. Exploratory Data Analysis
9. Feature Engineering
10. Baseline Model Again
11. Feature Selection
12. Model Selection
13. Model Fine-Tune

## 2 Project Overview

Expedia is interested in predicting which hotel group a user is going to book. Expedia has in-house algorithms to form hotel clusters, where similar hotels for a search (based on historical price, customer star ratings, geographical locations relative to city center, etc) are grouped together. These hotel clusters serve as good identifiers to which types of hotels people are going to book, while avoiding outliers such as new hotels that don't have historical data.

### Goal:

Predict the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event. For each user event, predict up to 5 hotel clusters.

### Evaluation:

Mean Average Precision @5

$$\frac{1}{|U|} \sum_{u=1}^{|U|} \sum_{k=1}^{\min(5,n)} P(k) \quad (1)$$

where  $|U|$  is the number of user events,  $P(k)$  is the precision at cutoff  $k$ ,  $n$  is the number of predicted hotel clusters.

Mean Average Precision at K is the mean of the average precision at K (APK) metric across all instances in the dataset. APK is a metric commonly used for information retrieval. APK is a measure

of the average relevance scores of a set of the top-K documents presented in response to a query. For each query instance, we will compare the set of top-K results with the set of actual relevant documents, that is, a ground truth set of relevant documents for the query. In the APK metric, the order of the result set matters, in that the APK score would be higher if the result documents are both relevant and the relevant documents are presented higher in the results. It is, thus, a good metric for recommender systems.

```
import ml_metrics as metrics
score = metrics.mapk(labels, preds.tolist(), 5)
```

Figure 1: Metric: MAPK

The issue remained is how to get the top 5 predicted labels based on probabilities.

```
import numpy as np
def top_k_labels(prob_matrix, k):
    a = prob_matrix.argsort(axis=1)
    b = np.flip(a, axis=1)
    c = b[:, :k]
    return c
```

Figure 2: Top k Predicted Labels

Other related metrics:

P@K: How many relevant items are present in the top-k recommendations of your system. For example, to calculate P@3: take the top 3 recommendations for a given user and check how many of them are good ones. That number divided by 3 gives you the P@3.

AP@K: The mean of P@i for i=1, ..., K. For example, to calculate AP@3: sum P@1, P@2 and P@3 and divide that value by 3. AP@K is typically calculated for one user.

MAP@K: The mean of the AP@K for all the users. For example, to calculate MAP@3: sum AP@3 for all the users and divide that value by the amount of users.

### 3 Data Understanding

Train set contains the following columns shown in Figure ?? . Three columns, 'is-booking', 'cnt', 'hotel-cluster' are not available in test dataset. An extra ['id'] exists in test dataset.

### 4 Brain Storming

Before diving into the data processing and modelling, it is better to have a deep thought on how to solve the problem from the intuition level. We ask fundamental questions.

**KG: What information do we need to solve the problem?**

Intuitively, what hotel clusters are interested by the user depends on the information of users and the information of hotels. In terms of user information: the purpose of the booking: leisure, working, family trip?, the intended time, the preference: location, transportation, room number, price, the duration of stay, membership, booking history. On the side of hotel information: opening time, location, transportation around, room types, price, room services, ratings, any discount, membership.

**KG: What information is provided by the data?**

The information contained in the dataset is relative limited compared with what we need. In particular, the information regarding hotel is almost none. We discuss the information provided by each column in detail.

- datetime: The time of the search session. **KG: useful. How many day book before visiting? holiday trip?**

Column name	Description	Data type
date_time	Timestamp	string
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	int
posa_continent	ID of continent associated with site_name	int
user_location_country	The ID of the country the customer is located	int
user_location_region	The ID of the region the customer is located	int
user_location_city	The ID of the city the customer is located	int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	double
user_id	ID of user	int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	int
channel	ID of a marketing channel	int
srch_ci	Checkin date	string
srch_co	Checkout date	string
srch_adults_cnt	The number of adults specified in the hotel room	int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	int
srch_rm_cnt	The number of hotel rooms specified in the search	int
srch_destination_id	ID of the destination where the hotel search was performed	int
srch_destination_type_id	Type of destination	int
hotel_continent	Hotel continent	int
hotel_country	Hotel country	int
hotel_market	Hotel market	int
is_booking	1 if a booking, 0 if a click	tinyint
cnt	Numer of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

Figure 3: train/test.csv columns

srch_destination_id	ID of the destination where the hotel search was performed	int
d1-d149	latent description of search regions	double

Figure 4: destinations.csv columns

- site-name: **KG: maybe useful**, if users from different site show different preference.
- posa-continent, user-location-country, user-location-region, user-location-city: these are the location of user when the search session happened. **KG: maybe useful**, if users from certain area are interested in similar hotels.
- ori-destination-distance: how far from the user and the hotel **KG: Not useful**, in general. In cases that the user intends to change a hotel nearby, maybe useful.
- user-id: **KG: Useful**, the same user may have consistent preference over hotel.
- is-mobile: **KG: Not useful** in general, except hotels displayed in mobile and pc are different.
- is-package: **KG: Maybe useful** in sense that members are more likely to book than random user.
- channel: id of market channel. **KG: if some channel offer discounts**, this maybe useful.

- srch-ci, srch-co: check in/out time: **KG: very useful**
- srch-adults-cnt, srch-childer-cnt: how many people. **KG: useful**
- srch-rm-cnt: number of room. **KG: useful**
- srch-destination-id, hotel-continent, hotel-country: the location of hotel. **KG: not very useful**
- hotel-market: **KG: what is this?**
- is-booking: **KG: not useful, missing in test set**
- cnt: number of similar events in the context of the same user session. **KG: yes, show interests of user**
- hotel-cluster: target variable.

**Note:** Note that the above discuss are based on intuition, whether some features are useless needs to be confirmed by data analysis.

**KG: How to process the data?**

Based on the discussion above, we need to process the data to make desired information explicitly.

- Remove apparently useless columns.
- Generate new features: number of nights, number of total people. the gap between the time of search session and visiting date.
- Figure out the type of trip: business, leisure, family trip.
- datetime: holiday, weekend, etc.

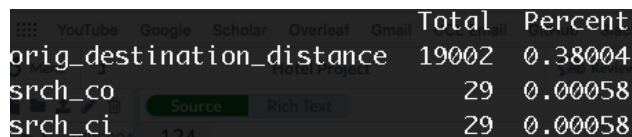
**KG: What model to use?**

Many models are can used to multi-label classification problem.

## 5 Data Cleaning

We handle missing values and outliers.

Three columns contains missing value" ['ori-destination-distance', 'srch-ci', 'srch-co'].



	Total	Percent
orig_destination_distance	19002	0.38004
srch_co	29	0.00058
srch_ci	29	0.00058

Figure 5: Missing Value

In dealing with missing values, we typically have options: 1), Remove the rows or the entire columns. 2), Impute values to replace missing values.

As the discuss above, 'ori-destination distance' could be useful. Regarding the rest columns, we can impute 0 to for missing value to indicate no check in/out dates are available. In summary,

- 'orig-destination-distance': impute the mean.
- 'srch-ci', 'srch-co': impute 0.

```
# fill na
train['night_num'] = train['night_num'].fillna(0)
train['ci_day'] = train['ci_day'].fillna(0)
train['ci_month'] = train['ci_month'].fillna(0)
train['adv_days'] = train['adv_days'].fillna(0)
train['orig_destination_distance'] = train['orig_destination_distance'].fillna(train['orig_destination_distance'].mean(), inplace = True)
```

Figure 6: Fill Missing Values

To check the outliers, we show histogram of all numerical features.

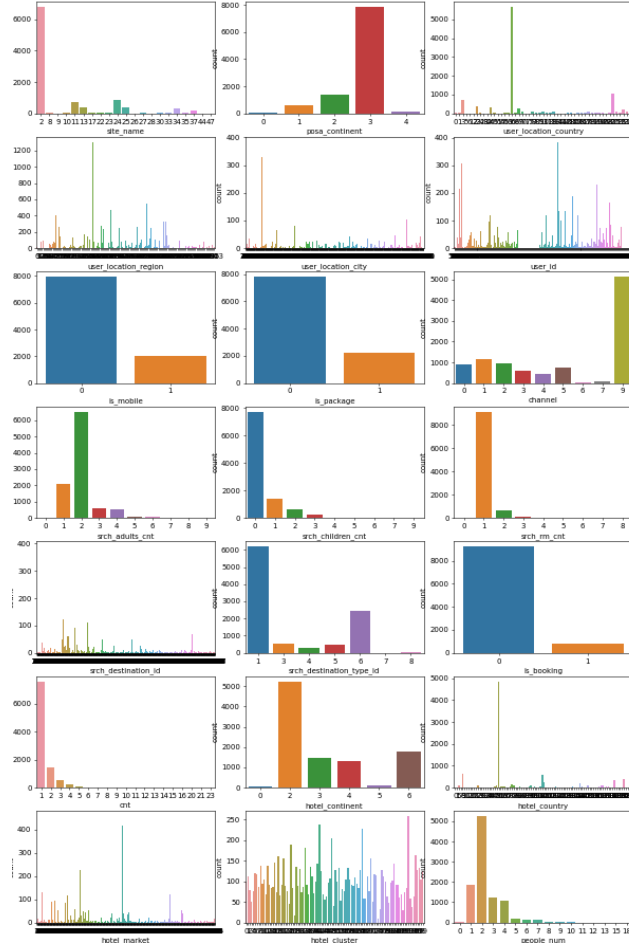


Figure 7: histogram

Figure 7 shows that all features do not follow normal distribution. However, this is reasonable for the project. It is unsuitable to remove or impute outliers.

Since this is a multi-label classification problem, it is good to check whether labels distribute balanced. As shown in Figure 8, the distribution of hotel cluster seems reasonable. There are few hotel clusters appear rarely.

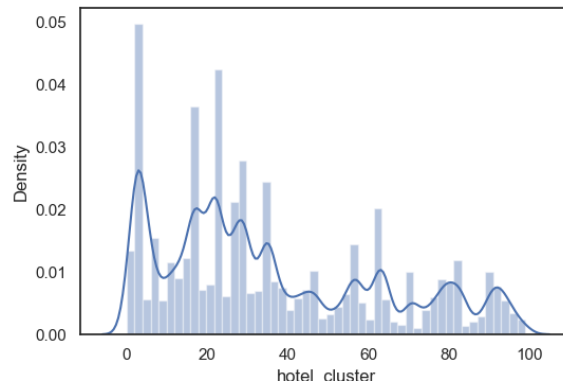


Figure 8: The distribution of hotel cluster (target labels)

## 6 Data Transformation

We transform all features into numerical values to get the data ready for modelling. In this dataset, we need to transform boolean and datetime into numerical value.

- 'date-time', 'srch-ci', 'srch-co' are splited as day, month, year, holiday.
- stay duration: 'srch-co' - 'srch-ci'
- holiday: convert from Boolean to integer 1/0.

## 7 Baseline Model

Now the dataset is ready for modelling. We can run a baseline model to have an understanding on the performance of raw dataset. The performance of baseline will guide feature engineering, feature selection and model selection.

Note that in order to calculate the evaluation metric, the probability of each label is required. Therefore, we need models that return the probability of each label instead of a single label. Decision Tree Classifier always gives the top 1 most likely result, which makes it not suitable for this task. We use Random Forest Classifier as the baseline model.

	Accuracy	MAP@5
Train	0.16	0.26
Test	0.14	0.23

Table 1: Random Forest Baseline Performance

As the number of unique labels are large ( $\approx 100$ ), the accuracy would be low. In recommendation system, MAP@5 is a more suitable performance metric.

## 8 Exploratory Data Analysis (EDA)

In this section, we explore the dataset to investigate the correlation between features.

- Scatter Plot
- Correlation Matrix

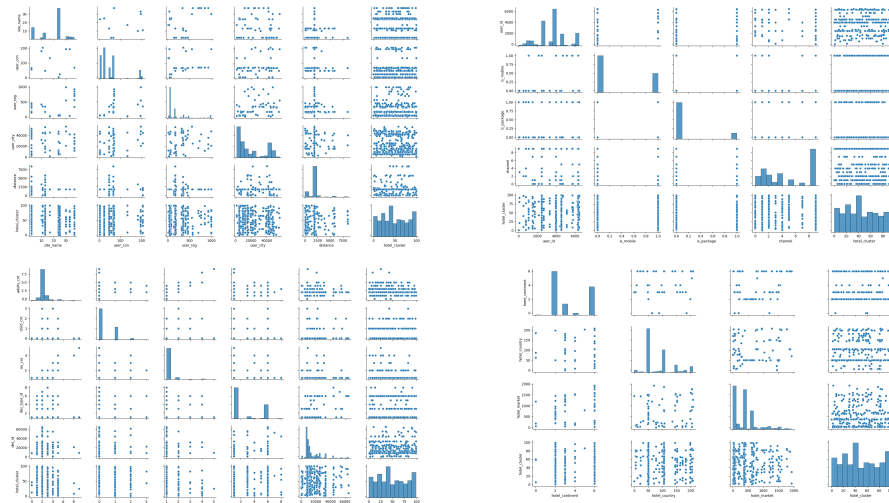


Figure 9: Scatter Plot

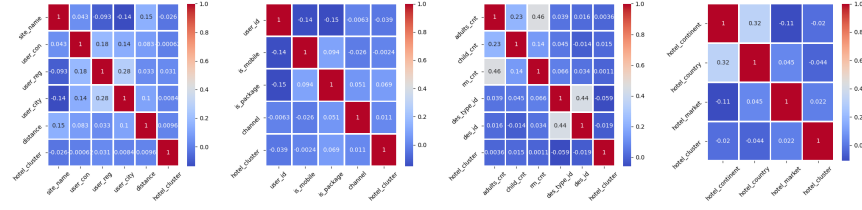


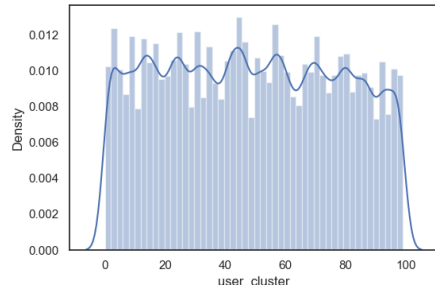
Figure 10: Correlation Matrix

The correlation matrix show the linear relationship between features and the target label. Figure 10 shows the linear relationship is quite weak. This is due to the relationship are high non-linear.

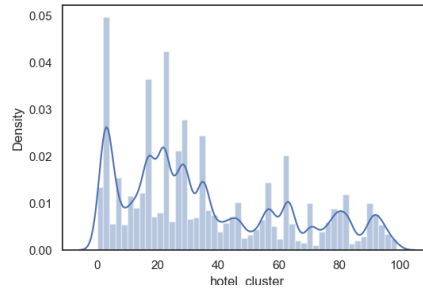
## 9 Feature Engineering

Following the discussion in Brain storming, we generate new features to make important information more explicit.

- Is the check in date a holiday.
- The total number of people.
- The total number of nights.
- Days before check-in.
- How many days search the hotel before check in date,
- Any children.
- Cluster hotels based on location.
- Cluster users based on location.



(a) User Cluster



(b) Hotel Cluster

Figure 11: User/Hotel Cluster

## 10 Feature Selection

Modelling with too many features could result over-fitting or the curse of dimensionality. Feature selection techniques are useful to mitigate such problems.

- Feature Importance of Random Forest.
- Explained Variance of PCA.

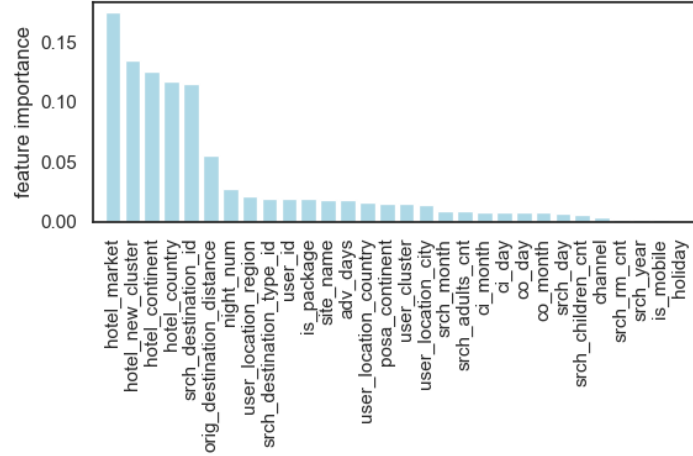


Figure 12: Random Forest Feature Importance

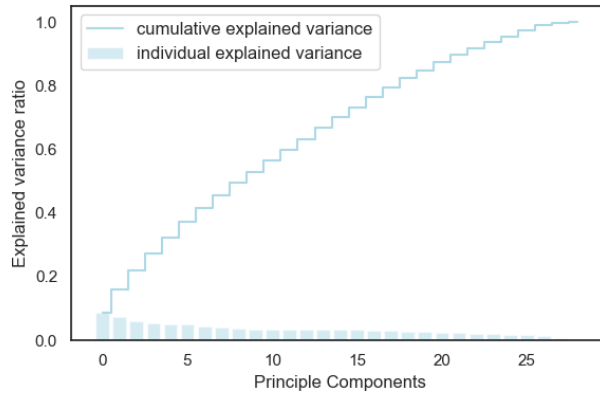


Figure 13: PCA Score

Figure 12 points that the most important features are 'Hotel Market', 'Hotel New cluster'. Figure 13 indicates no principle components contain a dominant part of information. This can be investigate deeper with  $t - SNE$  algorithm as it is a non-linear dimensionality reduction techniques.

## 11 Model Selection

Many models are suitable for multi-class classification. We use KNN, Random Forest, Neural Network, XGBoost and LightGBM.



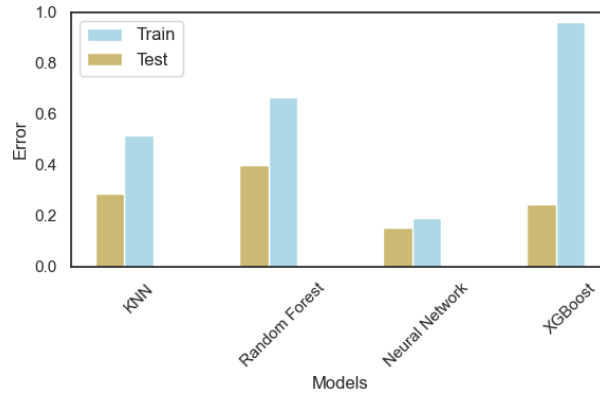


Figure 14: MAP@5

Models are trained with default hyper-parameters. Figure 14 shows that Neural Network is underfitting, while XGBoost suffers seriously overfitting.

## 12 Model Fine-Tune

To push models to their limits, We fine-tune the hyperparameters of Neural Network and XGBoost.

### Neural Network

- Layer number : 3
- Dropout : 0.2
- Hidden Node: 128, 64, 32
- Activation Function : ReLU
- learning rate: 0.01
- epoch-num: 300
- batch-size: 32

### XGBoost

- $\eta$ : 0.1
- max-depth: 10
- min-child-weight: 20
- n-round: 200

	KNN	Random Forest	Neural Network	XGBoost
Train	0.51	0.66	0.48	0.52
Test	0.30	0.4	0.28	0.19

Table 2: MAP@5

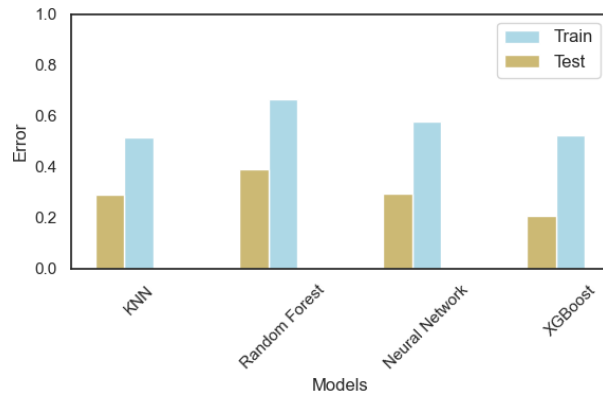


Figure 15: MAP@5

### 13 Further Improvement

Every data processing step has its own influence on the final model performance. Several approaches can be tried:

- Replace K-means by t-SNE
- Combine raw data with PCA.
- Add external data.