Machine Learning Engineer Nanodegree Capstone Report

Project Overview

This project was a past competition titled “Expedia Hotel Recommendations” [1] from Kaggle. Hotel booking can be daunting and tedious task as there are hundreds of hotels to choose from at each destination. Through this competition, Expedia aims to provide personalized hotel recommendation to its users by predicting the likelihood the user will stay at different hotel groups. Some interesting references that I will be referencing include Vik Paruchuri from Dataquest [2] who has an interesting take on using non-machine learning methods and from Aditi from USCD [3] who recommended a few good machine learning algorithms for this project. I am interested in this project because I personally would like to experience better hotel recommendations and improve my hotel booking experience.

Problem Statement

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 centre etc) are grouped together. The goal here is to predict the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event provided in the logs of customer behaviour. I would model this as a multi-class classification problem.

Metrics

I will be using the Mean Average Precision @ 5 (MAP@5) as the evaluation metric. The same metric was used for the Kaggle competition. The formula [4] is as follows:

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

where |U| is the number of user events, P(k) is the precision at cutoff k, n is the number of predicted hotel clusters. It basically means we are making 5 predictions per user event. For each prediction, calculate the precision at that position and sum all 5 precisions, then take the average across all user events.

Since hotel recommendation system usually recommends more than 1 hotel, MAP is a good evaluation metric as it allows for multiple recommendations, rewarding correct recommendations that comes first more.

Data Exploration

The customer behaviour dataset that Expedia provided includes information like what customers searched for, how they interacted with search results (click/book), whether the search result was a travel package. There is a total of 23 features. Table 1 below shows the schema and column description of the datasets.

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	Number of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

Table 1

The dataset is split into the training data from 2013 to 2014 and test data from 2015. Training data includes all the users in the logs, including both click events and booking events. Test data only includes booking events. However, the test dataset did not have the hotel cluster column which is not useful to me as I cannot evaluate my models with the test dataset. I instead used 10% of the training data as test data.

There is also an additional destination dataset which consists of 149 features extracted from hotel reviews text.

The training data set contains 37,670,293 rows. I dropped rows with NA values from any columns after which I randomly sampled 100,000 rows for my analysis due to lack of computation power.

Exploratory Visualization

Figure 1 below shows the correlation coefficient heatmap of the features and the predictor hotel_cluster. The following observations can be made:

- hotel_cluster does not have any strong correlation with any of the features.
- is_packaged has a positive correlation with duration. This means that people on a longer duration travel tend to book their hotels through a packaged deal.

- srch_rm_cnt has a positive correlation with srch_adults_cnt. This is expected as one would need more rooms to accommodate more adults.

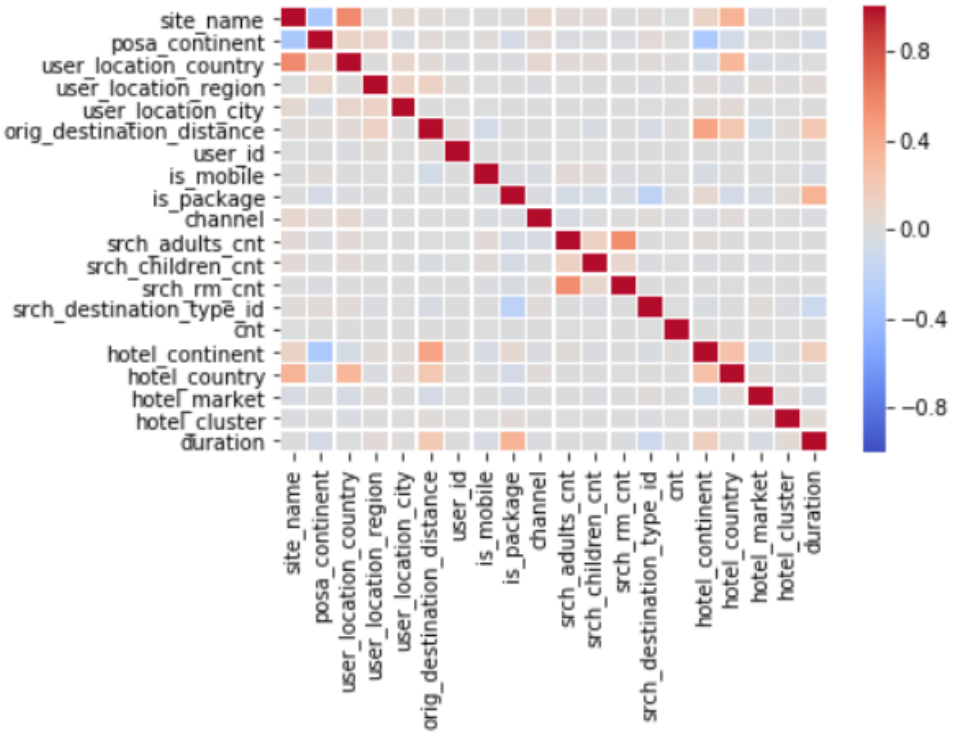
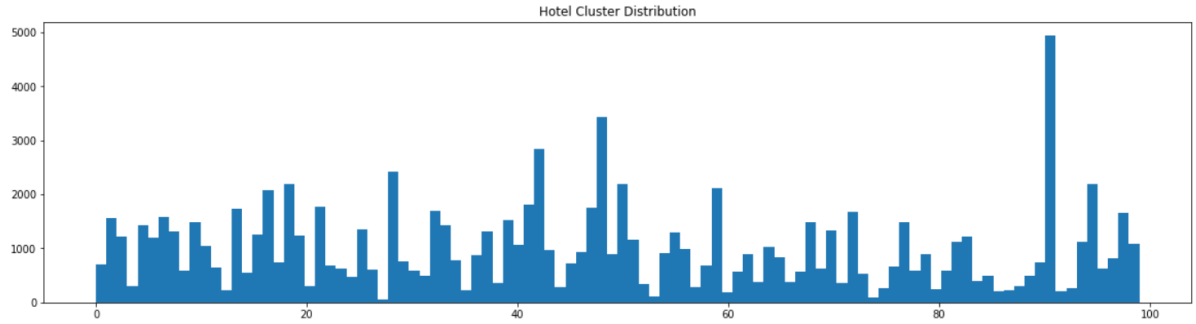


Figure 1: Correlation Coefficient Matrix

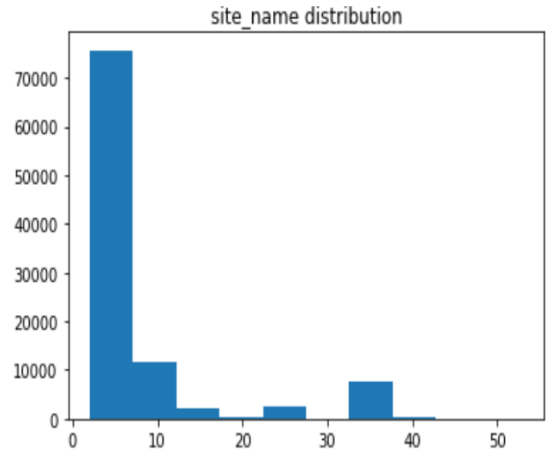
Figure 2 below shows the frequency distribution of the hotel_cluster. The distribution is not evenly spread out among all the distinct 100 clusters, which takes values 0 to 99. The performance of prediction models may not accurately predict those clusters with less data.

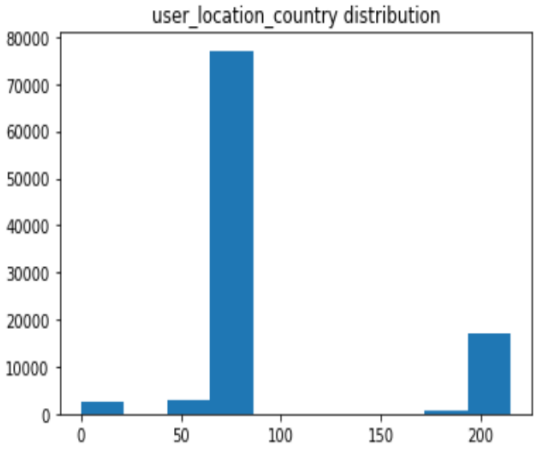


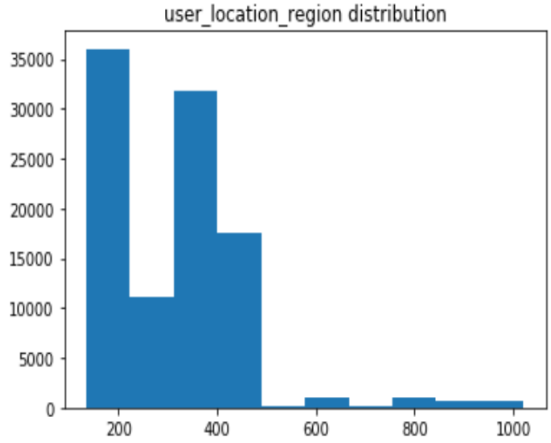

```
The most booked hotel cluster is: 91
```

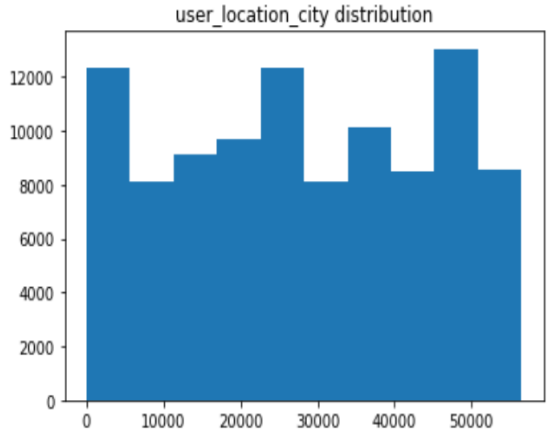
Figure 2: Hotel Cluster Distribution

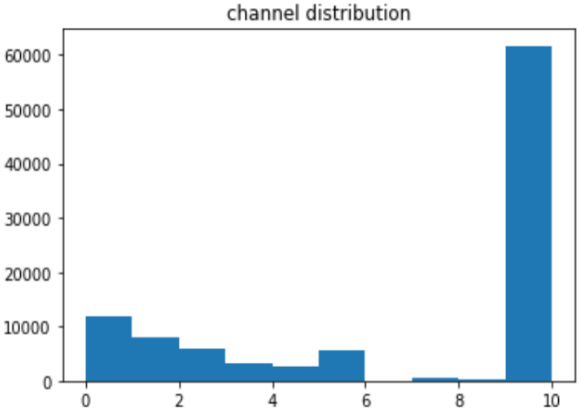
Figure 3 consists of the frequency distribution of all the categorical features. All of them are not evenly distributed. Some of them, like user_location_city & hotel_market, has many levels (>2000) which makes encoding them a challenge as it will increase the dimensionality tremendously.

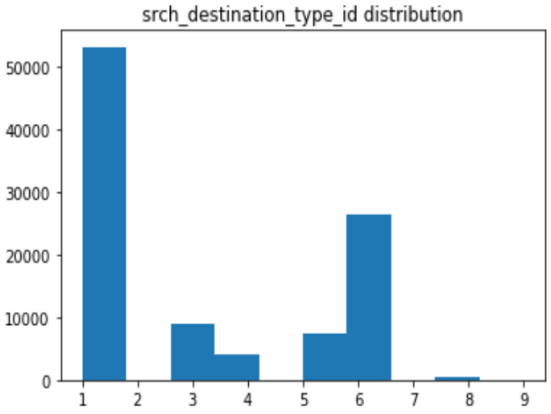

site_name distribution
70000
60000
50000
40000
30000
20000
10000
0
0
10
20
30
40
50

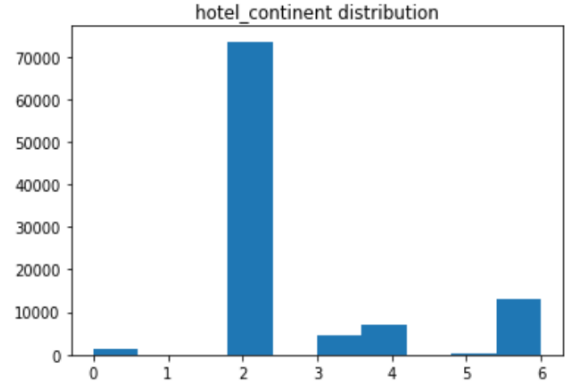

user_location_country distribution
80000
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200

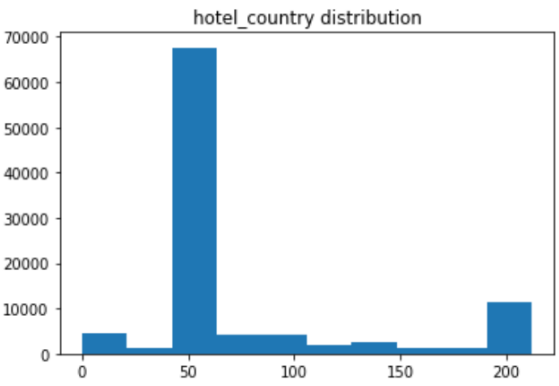

user_location_region distribution
35000
30000
25000
20000
15000
10000
5000
0
200
400
600
800
1000


user_location_city distribution
12000
10000
8000
6000
4000
2000
0
0
10000
20000
30000
40000
50000


channel distribution
60000
50000
40000
30000
20000
10000
0
0
2
4
6
8
10


srch_destination_type_id distribution
50000
40000
30000
20000
10000
0
1
2
3
4
5
6
7
8
9


hotel_continent distribution
70000
60000
50000
40000
30000
20000
10000
0
0
1
2
3
4
5
6


hotel_country distribution
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200

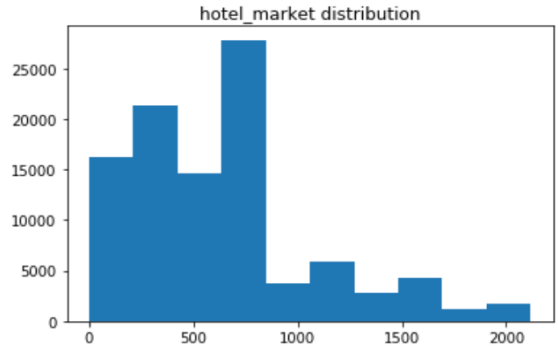
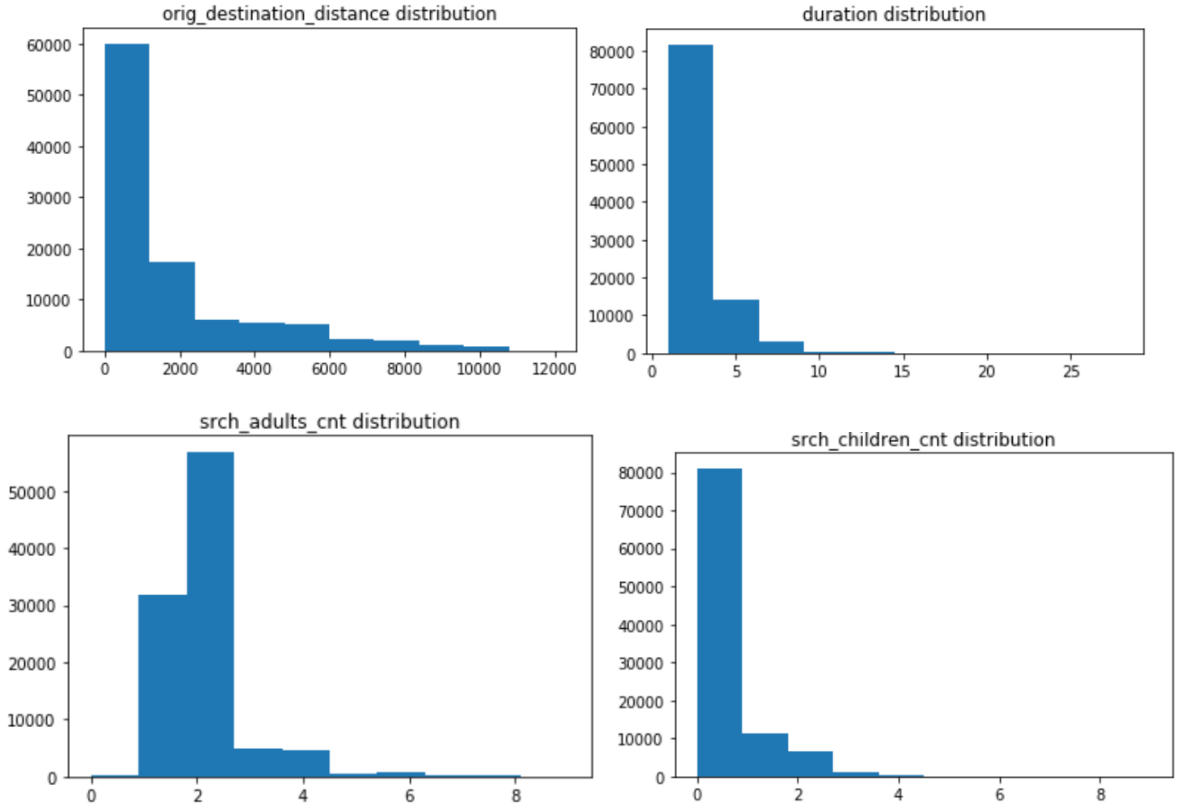


Figure 3: Frequency distribution of categorial features

Figure 4 shows the frequency distribution of all the continuous features. All of them are highly skewed. Extreme values may negatively affect the performance of a learning algorithm. The orig_destination_distance feature's value range is 1200 times more than that of the other continuous features. Learning algorithm may treat orig_destination_distance as a more important feature which is not true.



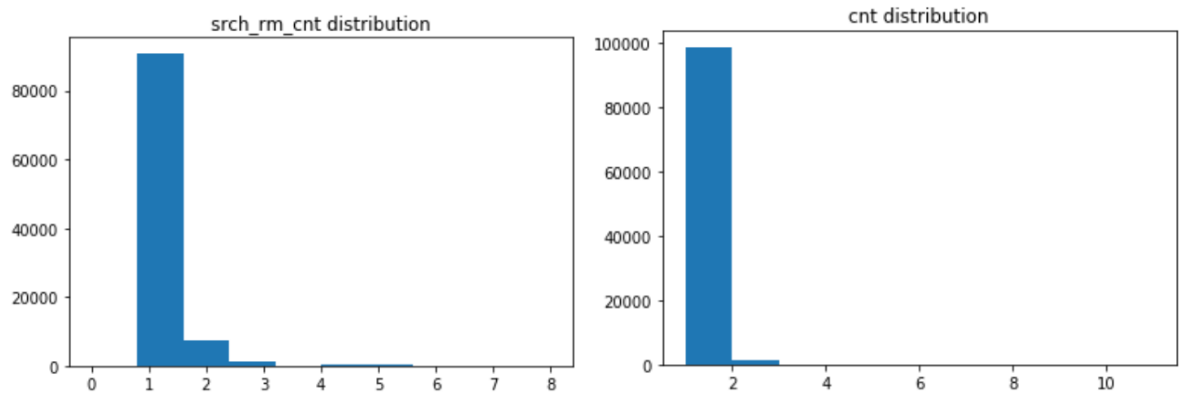


Figure 4: Frequency distribution of continuous features

Algorithms and Techniques

I considered 3 machine learning algorithms for supervised learning to model this multi-class classification problem: K-Nearest Neighbour (KNN), LightGBM and XGBoost. The training data, with all features and hotel cluster, will be used to train these models. The trained model will then be used to predict the hotel clusters of each of the entries in the test data.

KNN

The KNN is a simple classification algorithm that predicts the hotel cluster of a data point based on the nearest k observations in the training data. The hotel cluster that has the highest vote will be the prediction.

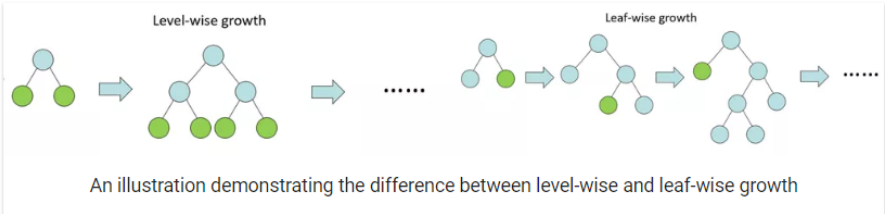
Say I have 100 training data points, K=5 and I want to predict the hotel cluster of a test point. I first calculate the distance (can be Euclidian, Chebyshev or Cosine etc) between test point and each training point. I then sort the 100 distances in ascending order and look at hotel cluster labels for the first 5 training points. The hotel cluster that has the highest frequency will be used as the prediction.

I have used default parameters (as stated in the official KNN documentation [5]) and k to be 5 when training the KNN model.

LightGBM & XGBoost

Decision trees for classification are a method of splitting the data based on features. Each branch in a decision tree divides the data into several groups base on certain criteria. Each leaf node is allocated with a single label. When predicting using the decision tree, the data is allocated to the appropriate leaf node base on the branch criteria, and the prediction is the label of that leaf node.

Gradient Boosting Decision Trees (GBDT) is a way of combining the predictions of multiple decision trees for better performance. Both LightGBM and XGBoost are a variation of GBDTs that uses leaf-wise growth strategy, rather than level-wise, when growing decision trees. Leaf-wise training is more flexible which makes it a better choice for large dataset like the Expedia data set we have here.



An illustration demonstrating the difference between level-wise and leaf-wise growth

The key challenge in training a GBDT is the process of finding the best split for each leaf as the algorithm goes through every feature of every data point. For a large data set, this will require lots of computational power. LightGBM and XGBoost separates themselves from typical GBDTs by using special methods to do splitting which allows them to complete training much faster.

LightGBM uses Gradient-based One-Side Sampling to do splitting. The idea behind this method is that not all data points contribute equally to training; data points with small gradients tend to be more well trained (close to a local minima). This means that it is more efficient to concentrate on data points with larger gradients. Hence, we ignore data points with small gradients when computing best splits.

XGBoost uses Histogram-based methods to do splitting. It groups features into sets of bins and perform splitting on the bins instead of the features since small changes in the split do not make much of a difference in the tree performance. This reduces training time significantly.

I have mostly used the default parameters for implementing LighGBM (as stated in the official LightGBM documentation [6]), set objective=multi-class and num_class=100. For XGBoost, I have used the default parameters (as stated in the official XGBoost documentation [9]).

Benchmark

The benchmark model that I have used is one that always uses the top 5 most common hotel clusters, in order of most to least common, in the training data as the prediction. The top 5 most common hotel clusters can be found by analysing the frequency distribution of hotel_cluster column. They are clusters 91, 48, 42, 28 and 95.

Data Pre-processing

From the data exploration visualization above, encoding the categorical variables is a big challenge because they have many levels. Doing the usual one hot encoding will increase the dimensionality tremendously. I adopted the approach of reducing the number of levels of each categorical variable by grouping all levels in the last 5 percentile of the frequency distribution as “rare-values”. I also implemented binary encoding instead of one hot encoding which reduced the dimensionality further. Binary encoding transforms each distinct level of a variable into a unique binary string and then split it into separate columns. A variable with 32 levels will only require 5 columns to encode (2^5=32) as compared to 32 extra columns to encode when using one-hot encoding.

For the skewed continuous variables, I applied a logarithmic transformation on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. I also applied normalization to these features to ensure each feature is treated equally when applying supervised learners.

For the 149 features extracted from hotel reviews text in the destination data set, I did a Principle Component Analysis on these features to reduce the dimensionality to 10 columns and then added them to the training data set as additional features, joined by srch_destination_id.

The check-in (srch_ci) and check-out (srch_co) dates can be used to calculate the duration that the customer would like to stay in the hotel for. I feel like the duration of stay will most likely impact the type of hotel that the customer would like to stay in. For example, a customer staying for long period of time may be more price sensitive that those who are having a short stay, hence they may prefer

budget hotels. I removed rows with check-in dates later than check-out dates which do not make any sense.

Finally, I dropped the columns date_time, user_id, srch_ci, srch_co, is_booking and srch_destination_id for my analysis. srch_destination_id is no longer needed after we joined the hotel reviews features to the training data. The newly created duration feature considered the srch_ci and srch_co features. is_booking is dropped because we are only interested in customer data that resulted in booking, ie is_booking=1. The user_id feature has too many levels, including it will cause the dimensionality of the problem to spike.

Implementation

After the data pre-processing, there are 99,543 rows and 76 features in the dataset. I created a validation and test data set using 10% of the original data for each set with the train_test_split function from sklearn. model_selection package. The rest of the data is the training set.

For training the benchmark model, I rank the hotel cluster frequency vector in the training data set from most to least frequent. I then take the top 5 most frequent hotel cluster as the prediction for every test data point. I used the XGBClassifier function from xgboost package to train the XGBoost model using the default parameters. The KNN model was trained using the KNeighborsClassifier function from sklearn.neighbors package using the default parameters. Lastly the LightGBM model was trained using the lightgbm function using the default parameters as well.

Since we are using MAP@5 metric to evaluate our models, for each test data point, I chose the 5 most probable predictions (in order of most to least probable) as the prediction vector for that point. I used the mapk function in ml_metrics package to calculate MAP@5 for my test data prediction for each model.

I did not encounter any coding complication when during the implementation phase.

Refinement

I did some refinement to the XGBoost, LightGBM and KNN models by doing parameter tuning. This is done by retraining all the models by using different hyper parameters and then pick the one that performs best. I implemented this with the GridSearchCV function in the sklearn.grid_search package. The parameters search dictionary that I used for all 3 models is as follows:

Model	Hyper Parameter	Values
LightGBM	learning_rate	**0.05**, 0.1, 0.2
	num_leaves	**21**, 31, 51
XGBoost	max_depth	3, 6, 9
	learning_rate	0.05, 0.2, 0.3
KNN	n_neighbors	3, 5, 10, **15**

The best parameters for LightGBM are learning_rate=0.05, num_leaves=21. KNN performs the best when n_neighbors=15. I was not able to refine the XGBoost model as I encountered some error that I could not resolve when running the GridSearchCV function.

Model Evaluation and Validation

The MAP@5 scores of the validation set for each of the models trained are summarized as follows:

Model	MAP@5 Score
Benchmark	0.087
XGBoost	0.248
KNN	0.146
LightGBM	0.265

The LightGBM model performed the best with the MAP@5 score of 0.265. It also generalizes well when tested against the test set, producing a MAP@5 score of 0.254.

I tested the robustness of the LightGBM model by training the model with 9 different random states in the train test split and then obtain the MAP@5 score for each of the 9 different test sets. The mean and variance of the 9 MAP@5 scores was 0.264 and 0.000000744 respectively. Since the variance is very low, this may prove that the LightGBM model is data invariant and stable.

Justification

From the model evaluation section, it is obvious that the refined LighGBM model performed around 3 times better than the benchmark model (MAP@5 score of 0.087 vs 0.276), making it the more superior solution. However, I feel that the score is still too low for actual use to provide hotel recommendations. Imagine being recommended the incorrect type of hotel more than 70% of the time when you are browsing the Exepdia site, it will not be a good user experience.

Free-Form Visualization

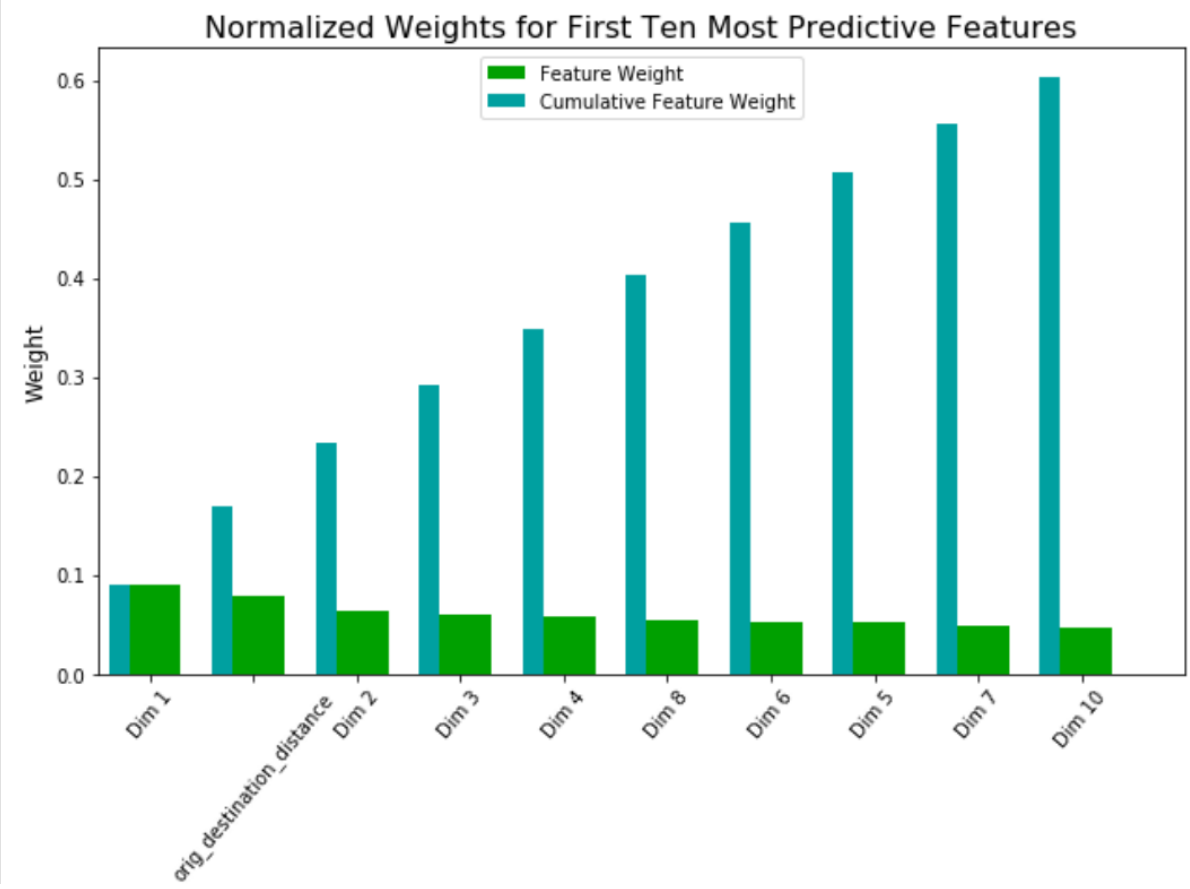


Figure 5

Figure 5 shows the normalized weights for the top ten most important features for the refined LightGBM model. These 10 features, out of a total of 76 features, represent almost 60% of the feature weights. 9 of the 10 features are from the destination data set which represents features extracted from hotel review text. This means that the hotel reviews play a significant role in determining which hotel cluster the customer is going to book.

Reflection

In summary, I did:

1. Data exploration on the frequency distribution of the features.
2. Which led to processing the data by doing binary encoding on categorical features and applying logarithmic transformation & normalization on continuous features.
3. Splitting the final data set into train, validation and test sets. Training three candidate models (XGBoost, KNN & LightGBM) using the training set. Evaluate the three and benchmark models using the validation set. Finally use the best model to predict the test set.
4. Used grid search method to further refine the best model by obtaining better parameters.
5. Did a simple sensitivity test to test the robustness of the best model.

One interesting aspect of the project is that there are several categorical features with large number of levels. Using the usual one-hot encoding method to encode them will result in 53,475 extra columns, increasing the dimensionality of the problem tremendously which will decrease the effectiveness of machine learning models. Through some research I managed to find ways to reduce the dimensionality by grouping rare values together and using binary encoding to encode those features.

One difficult aspect of the project is the large data set provided by Expedia. There were 37 million rows in data set and I had no idea how much data can my laptop process with a reasonable amount of time initially. I tested with all 37 million rows, then sampled 2 million and then 1 million rows to do my analysis before finally settling with 100,000 rows. I also could not afford the time to re-train my models whenever I had to close the kernel, so I had to research for ways to save the trained models down for future reference.

The final refined LightGBM model did fit my expectation for the problem. It managed to successfully make multi-class predictions with a MAP@5 score of 0.28. It can be used in a general setting to solve a multi-class classification problem.

Improvement

The MAP@5 score for the refined LightGBM model was too low for actual hotel recommendation. Improvements can be made to achieve better results.

Including the features user_ID and is_booking in the analysis may help. Individual user has his or her own unique preferences hence using user_ID can help with predictions. Those browsing sessions that did not result in a booking (is_booking=0) should also uncover some useful insights on hotel booking.

I also tried to implement ensemble stacking where I combined all three candidate models to make predictions but was not confident in implementing it in the end. Using a stacking method will definitely improve the overall MAP@5 score as it combines the strength of all three models.

Acknowledgment

[1] Kaggle Expedia Hotel Recommendation Challenge

https://www.kaggle.com/c/expedia-hotel-recommendations

[2] Vik Paruchuri. How to get into the top 15 of a Kaggle competition using Python

https://www.dataquest.io/blog/kaggle-tutorial/

[3] A.Mavalankar, A.Gupta, C. Gandotra and R. Misra. Hotel Recommendation System

https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a038.pdf

[4] Kaggle Expedia Hotel Recommendation Challenge - Evaluation Metric

https://www.kaggle.com/c/expedia-hotel-recommendations#evaluation

[5] KNN Parameters

http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor

[6] LightGBM Parameters

https://lightgbm.readthedocs.io/en/latest/Parameters.html

[7] P.Mandot. What is LightGBM, How to implement it? How to fine tune the parameters?

https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[8] J.Brownlee - A Gentle Introduction to XGBoost for Applied Machine Learning

https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[9] XGBoost Parameters

https://xgboost.readthedocs.io/en/latest/python/python_api.html

[10] M.Pathak - Handling Categorical Data in Python

https://www.datacamp.com/community/tutorials/categorical-data

[11] Keitakurita - LightGBM and XGBoost Explained

http://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/