Expedia Hotel Recommendation Project

Seungmin Baek, Younghun Kim, Sang Hyeong Woo

CSCI-B351 Introduction to Artificial Intelligence

seubaek@iu.edu, kim754@iu.edu, sangwoo@iu.edu

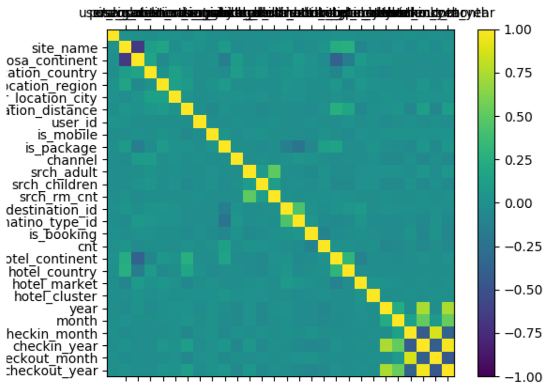
Our group chose 'Expedia Hotel Recommendation Competition' as the topic of our project. The idea of this competition was acquired from 'kaggle.com', an online community where users publish challenges of real-life data science projects. The purpose of the Expedia Hotel Recommendation Project is to contextualize customer data and predict the likelihood a user will stay at 100 different hotel groups. The project was launched on April 15, 2016 and closed on June 10, 2016. The total of 1,974 teams participated in this competition, and it was won by a team called "idle_speculation". Total of 4 different data files are provided for this competition participants including destination.csv, sample_submission.csv, test.csv, and train.csv. The train.csv file contains the total of 37,670,293 data, and there are 24 different data fields. Here are the descriptions of each data fields:

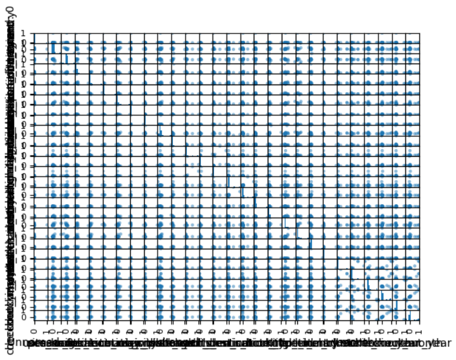
Column name	Description	Data type
date_time	Timestamp	string
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	int
posa_continent	ID of continent associated with site_name	int
user_location_country	The ID of the country the customer is located	int
user_location_region	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

The first task was preprocessing given data. On top of the fact that some of the training data provided by Expedia are dirty and irrelevant, using all 37,670,293 data for machine learning was impractical and time-consuming. Therefore, we had to clean and reduce the amount of data that will be used for actual machine learning. The very first step of data process was taking '*date_time*' field and separating *month* and *year* values into separate columns. This process was necessary as string-date datatype is not compatible when applying machine learning algorithms, such as XGBoost and Random Forest Algorithms. Then, we had to make sure that the training data and testing data did not overlap. We created a new variable called *sample_train* and stored data that did not overlap, and we were able to narrow the number of data down to 456,701. Afterwards, we learned that 183,290 data within *sample_train* variable had null value in *orig_destination_distance* column and 570 in *srch_ci* and *srch_co* columns. Because we had to make sure there aren't any null data, we decided to throw out all data that have null values in *srch_ci* and *srch_co*, then calculate the average value of *orig_destination_distance* column and store it into all null data. These successful procedures of preprocessing led us to have the total of 456,131 sample data that would be used for machine learnings.

Below is the graph showing the correlation between each features. From the graphs we can learn that the features doesn’t have much correlation, thus we won’t be able to use any regression methods in this project:

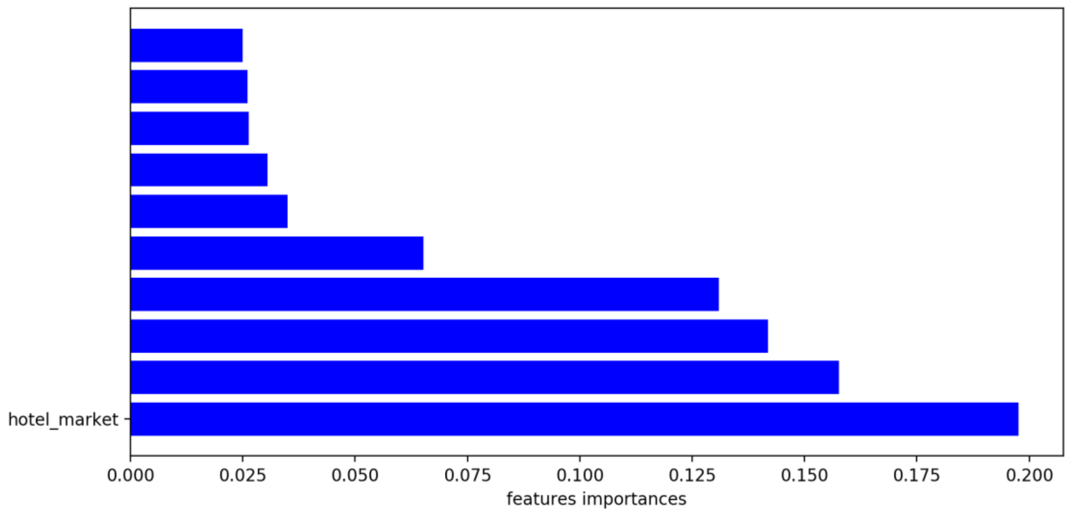




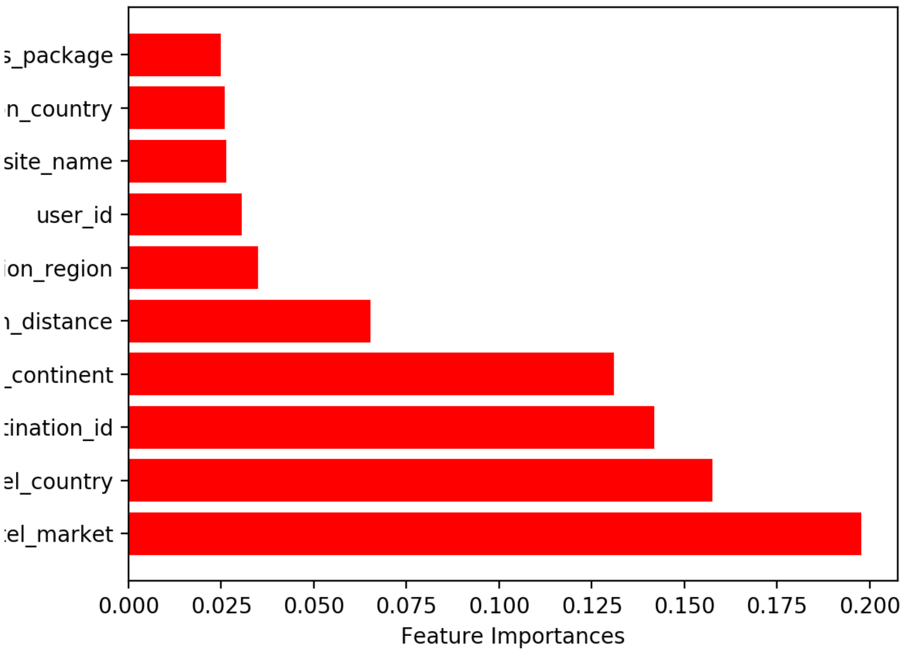
After preprocessing we started applying the XGBoost algorithm first to our train data. XGBoost or Extreme Gradient Boosting is a Greedy Function Approximation using a gradient boosting machine. We initially tried using the XGBoost library after cleaning the data with RStudio. We first created a test data as a matrix, which didn’t have the *hotel_cluster* and target features. Before moving on to the next step, we needed to decide on the parameter and the best number of iterations. The parameter for this project was the ETA value or the learning rate, the ETA value is important because when we choose a good eta value it can prevent overfitting. The value is in the range from 0 to 1. In XGBoost it’s usually best to keep the ETA value as low as possible. And iteration had to kept large since we wanted to test the data set as much as possible. The problem was that we weren’t able to modify the library to fit in our dataset and failed running the code.

The next algorithm that we tried was K-Nearest Neighbors, KNN is a classification algorithm where it calculates the distances of each objects and find the nearest neighbors of each objects. We decided to use this algorithm to see how accurate it is. We implemented our own KNN, yet it didn’t have any problems running but the time consumed took more than a day. Therefore, to get the results we used the built-in algorithm in RStudio to get the error rates. We ran the code with using three k values, 3, 5 and 10.

Afterwards, we applied Random Forest algorithm to our preprocessed data. We first created two different variables x (information) and y (target), and using these two variables we ran an initial machine learning with the test size of 0.2 using SCIKIT-Learn's train_test_split method. We first trained it using the train data and then applied it to the test data. The result of our Random Forest can be found in the submission.csv file which is attached to the zip file. We were able to produce a table with the appropriate formatting of the submission file. But, we weren't able to submit the file to the kaggle board, due to an error which we weren't able to resolve online. However we were still able to get a set with 5 different clusters that recommends to the user, through Random Forest algorithm.
Below is the graph showing importance of each data fields in related to the machine learning:



Below is the graph where we found the importance of features in our sample_train data.



One limitation of our project could be observed from preprocessing stage. As described above, 570 data contained null values in *srch_ci* and *srch_co* columns, and we decided not to include these data in our actual machine learning because we thought those are trivial. As we proceeded, however, we realized that we should not have made such assumption out of a speculation, and instead come up with other alternative method like inserting average values of the two columns into all null places. Such deletion could have affected the accuracy of recommendation prediction.

The biggest challenge we had throughout the project is implementing XGBoost algorithm into our code. We initially planned of implementing XGBoost algorithm to our preprocessed data. The actual implementation, however, was much more difficult than we expected. Finding good resources with easy descriptions and explanations of implementing the algorithm was very difficult. The actual implementation of the algorithm was very confusing as well, and, despite our various attempts, we failed to apply the algorithm to our case. Therefore, we decided to abandon our initial plan of using XGBoost and used Random Forest algorithm instead for machine learning.

In the future, if possible, we would like to try use XGBoost one more time after doing more research. And we would try using a different way to preprocess the data. There are multiple ways of analyzing and interpreting the data, next time we want to try using a different interpretation. Also we would definitely want to check out why our submission file wasn’t able to go up on the kaggle board. Lastly, we would like to try other machine learning algorithms as well. One of the algorithms that we would like to try and study is the Gradient Boost Tree.