Contents

1 Introduction

In recent years, we have seen a rapid growth of e-commerce websites due the popularity of Internet. Daily transactions generate enormous amount of data on such websites. Systematic learning from such unstructured data offers a great potential to know their customers and increase their engagement with these sites. Recommendation system is one of such tool which help businesses to personalize their customer's experience by recommending them the relevant products based on their needs and personal interest. It offers the customer a good shopping experience and saves their valuable time of browsing several web pages. Therefore, it plays the crucial role for attracting the customers and the growth of such businesses. Scientists are trying to improvise the traditional recommendation systems with the advent of new techniques. Machine learning is one of the powerful tool to achieve this goal. It has shown a great potential to solve many intractable problems. One such industry which has seen boom in the present decade is the online hotel booking agencies. Through this project, we aim to apply and evaluate the performance of various machine learning techniques for the hotel recommendation system.

In this paper, we propose machine learning to predict the customer's hotel booking preferences based on their interactions on the hotel booking websites. The project idea started with the Kaggle competition called 'Expedia hotel recommendation'. Expedia is a very popular online travel booking agency which regularly records millions of customer's activity logs on their website. The competition challenged individuals to use the Expedia data and use machine learning techniques to personalize their customer's hotel booking experience by giving recommendations on which hotel to book. They provided data[2] for 37 million customers containing information such as their search parameters and session information. The 'hotel cluster' notion was used to convert this challenge into a classification problem. Kaggle competition did not provide any additional information on how these hotel clusters were to be created. Moreover, most of the features provided by them were anonymous. Another problem associated with the given data was that it didn't follow any specific distribution and we could not find any linear relationship between the response and the provided variables. Most of these variables were hashed ids of the nominal attributes and had a high cardinality. Furthermore, predicting a particular hotel cluster from the group of 100 clusters imposed additional challenge on the classification task. Given these challenges, the objective was to predict a list of five hotel clusters that customer would choose for their booking needs. Additionally no labeled test data was released by the Competition Administrators. Therefore, for this project we only used available 'train' data[2]. We used the subset of the data for training the machine learning models and remaining holdout data was used as the test set to evaluate the performance of different classification algorithms. In addition to this, we also explored available unlabeled 'test' data [2] to analyze the process which Kaggle utilized to partition the original data into training and test to emulate the competition environment.

We considered this problem as the supervised learning since we had to train our model using labeled data to predict the unknown class for the test data. During this project, we applied different supervised machine learning techniques to solve this challenging multi-class classification problem. Kaggle used Mean Precision Score at 5 (MAP@5)as the evaluation metric for the given competition. Hence, we used this evaluation metric to measure our final prediction accuracy. The evaluation criteria is discussed in detail in later sections of this paper. Another criteria which was considered for measuring the performance of individual model was to use the cross validation and individual test score accuracy of the trained model. MAP@5 for the top 100 teams varies from 0.60 to 0.49[11]. This was set as a baseline benchmark for final result evaluation.

2 Dataset Description

2.1 Data Summary

Data comprised of 37,670,293 records based on user's search activities on the Expedia during the period 2013-2014. It represented information about 119,8786 unique users who interacted with Expedia during this period. Each row is composed of 24 variables describing the user's search parameters while booking the hotels. Broadly, these parameters can be grouped into following subcategories as follows:

- User attributes
  - user_id: Uniquely identify a specific user in the data
  - user_location_country: Country ID of the user
  - user_location_region: ID of the region the customer is located
  - user_location_city: Represents the user's current city
  - cnt : Number of similar events in the context of the same user session. For instance, cnt of 3 indicates that user clicked 3 times on the link of hotel information page within a single user session
- Hotel attributes
  - hotel_continent : Hotel continent
  - hotel_country : Hotel country
  - hotel_market : Hotel market
- Destination attributes
  - srch_destination_id : ID of the destination where the hotel search was performed
  - srch_destination_type_id : Type of destination
  - orig_destination_distance: Geodesic distance(miles) between a hotel and a customer at the time of search. NULL represents that the distance could not be calculated. This is due the missing value of latitude/longitude or user location
- Hotel search parameters
  - srch_ci: Check-in date
  - srch_co: Check-out date
  - srch_adults_cnt : Number of adults specified for booking the hotel rooms
  - srch_children_cnt:Number of (extra occupancy) children specified in the hotel room
  - srch_rm_cnt : Number of hotel rooms specified in the search
- Binary attributes
  - is_booking: '0' represents user has only seen a hotel on a search result page and clicked on it to see rates, reviews etc. Whereas, 1 indicate user booked the hotel
  - is_mobile: '1' indicates that user is connected from a mobile device otherwise its '0'
  - is_package: Value is '1' if the click or booking was generated as a part of a package (i.e. combined with a flight), otherwise it's '0'
- Expedia attributes

  - channel : Id of a marketing channel. It represents how had user accessed the Expedia website. These IDs represent values such as direct link, search engine marketing id like google's paid links or Meta channel such as Tripadvisor[3].
  - data_time: Represents timestamp relative to the site name

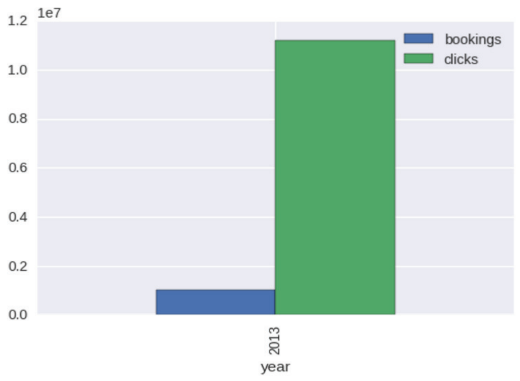
- hotel_cluster : Target having 100 unique categories.

Competition admin explained that the hotel clusters were created using the attributes such as hotel popularity, ratings, user reviews, rates etc[3]. They represented a group of the similar type of hotels in a very broad sense. All the ID features were created using complex hashing techniques to hide direct identification with the actual parameters.

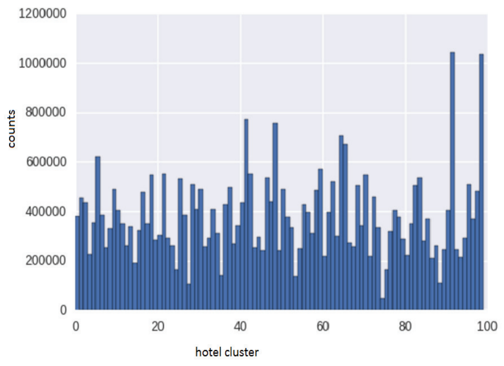
Additionally, Expedia provided another database 'destination'[2] which was an optional data to use for training the model. Each row represented a collection of latent attributes related to the hotel search based on the specific destination. These features were transformed numeric values of some unknown parameters. This information is provided only for 62,106 search destinations contained in the Expedia's training data.

2.2 Data Exploration Findings

The data available reflected high imbalance considering the booking/click events. For example, year 2013 showed only 9% of bookings. This led us to conduct a carefully review process to apply the sampling technique. Some of the hotel clusters had a very high booking/click rates which represented the imbalanced nature of responses. We analyzed the frequencies of bookings/clicks by grouping all the critical parameters such as user country, hotel continent, season etc. We found that hotel continents with ids 2 and 6 were popular among the users. Country with id 66 showed highest click rates. The observations showed that the rates tend to increase during the summer and fall seasons validating the dataset. One of such example is shown in the figure 2.a. Booking frequencies were very low for all the clusters as compared to their click rates(Figure-2.b). This meant that majority of the times customers only viewed the hotel clusters and did not continue towards booking the hotel. This created an added impediment to predict the booking outcome for the hotel clusters. Figure-3 shows the frequent booked/clicked hotel clusters by the users. we observed that there were approximately 35% missing values for the variable orig_destination. Only 15% book/click events were performed as the part of a package scheme.

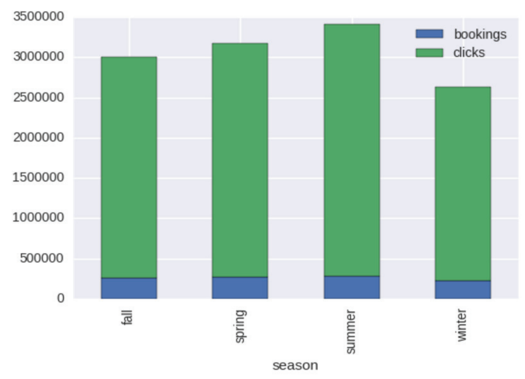


(a) Bookings/clicks distribution - year 2013

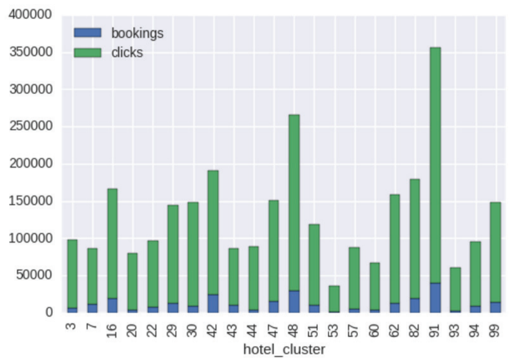


(b) Frequency Distribution for Hotel cluster

Figure 1: Data exploration



(a) Seasonal clicks rate for country id - 198



(b) Some of the popular hotel clusters using booking/click data

Figure 2: Data exploration

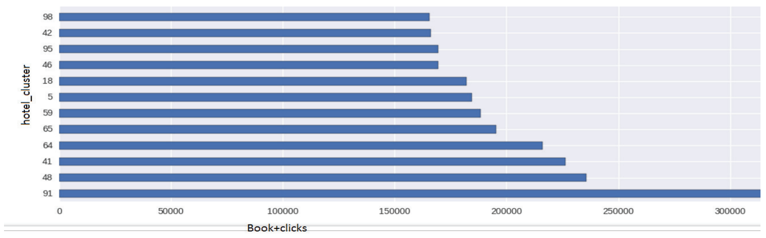
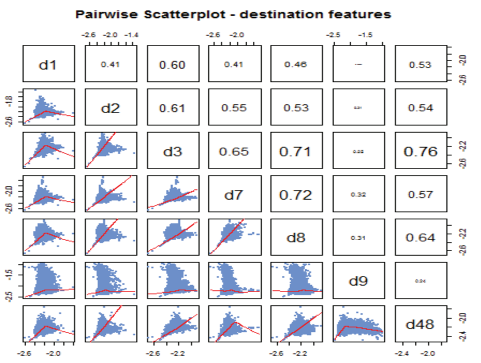
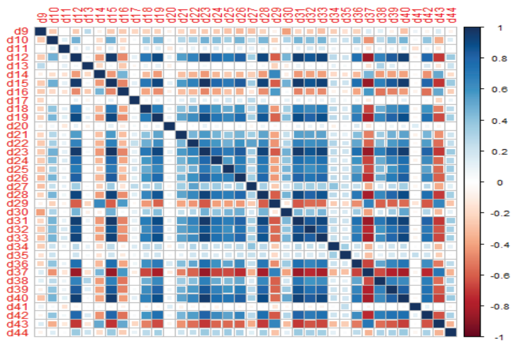


Figure 3: Top 12 popular Hotel Clusters

The 'destination' dataset consisted of 149 latent attributes which represent additional information about the variable search_destination in the train data. These features are highly correlated. Many of the variables show linear relationship between them. [Figure-4]. As a result, additional scrutiny was needed while incorporating these features to train the model. The discussion bulletin on the competition board speculated these values to be log transformation of some numeric attributes. Although, Competition administrators did not provide any additional clarification about these features.



(a) Pairwise scatterplot for some latent features



(b) Correlation plot for variables in the destination data

Figure 4: Destinations data exploration

3 Literature Review

The field of modern machine learning in recent times is actively conducting extensive research on techniques and issues discussed in this. Content-based recommendation systems considers similarity of items based on their attributes. This topic and collaborative filtering (CF) for predicting the customer's preferences for the products is discussed in detail in publication by Jure Leskovec et.al [4]. As compared to content based recommendations, the collaborative filtering focuses on similarities between the user and their product preferences. Jure Leskovec et.al[4] also discussed the utility matrix to learn the similarities between the products, customers and other attributes related to them. They also mentioned using the distance metrics such as Hamming, Jaccard to measure the similarities between the attributes. Ya-Han Hu et.al demonstrated use of collaborative algorithms in hotel recommendation problem using customer's attributes and review data. Chumki Basu et. al. [5] converted the recommendation learning into the classification task and demonstrated the use of inductive learning algorithms such as RIPPER to solve the problem. Gao Huming et. al (2010) implemented the combination of machine learning algorithm(RankBoost) and collaborative filtering in the hotel recommendation system. Expedia has hosted similar competitions in the past [7] where competitors[8][9] discussed the use of ML algorithms such as Random Forest, deep learning, boosting to personalize the hotel search based on the customer's purchase history and attribute associated with the hotels.

The problem of predicting the hotel cluster based on the data provided [2] was an effort to convert the hotel recommendations task into a multi-classification problem. Discussion forum for the competition [1] showed that the participants tried to solve this challenging classification problem by ensemble of multiple ML classifiers such as Random Forest, Support Vector Machine, XGBOOST. The winning team commented on their approach to solve this problem by attributing a good feature engineering and machine learning ensemble technique as the key to solve multi-class classification problem. Team Idle_Speculation described [10] their feature selection process and application of Gradient Descent, Extreme Gradient boosting(XGBOOST) to win this competition. They applied the factorization machine algorithm to learn the variables interactions with the response. Many other participants used traditional ML classifiers such as Support Vector Machine, Naive Bayes as their approach in search of the good model. Participants employed different sampling techniques to build the model considering the scale of the data. Traditional recommendation algorithms(CF and CL) were not found useful in this case due to the anonymous nature of the data provided.

4 Machine Learning Algorithms

We have considered following algorithms for the model building process from the plethora of machine learning techniques. Rationale behind trying out multiple techniques was to evaluate their performances to solve this challenging classification problem. Selection constitutes combination of standard and complex classifiers. In final step, we decided to ensemble them to predict the 5 hotel clusters which user most likely will book based on the their previous information.

4.1 Random Forest (RF)

Random Forest is a popular tree-based classifier which uses the ensemble of many decision trees to make the final prediction. It uses bootstrapping (with replacement) to create N samples of size n from the original data. Using these bootstrapped samples, N decision trees are built. The data points which were not used while sampling are used to estimate the out of bag errors. At the end, majority voting is used for predicting the class. Averaging the result of multiple decision trees reduces the variance caused by a single tree classifier. It randomly picks m variables to build each tree. This selection introduces the randomization during the tree building process. Each tree uses Gini index or cross-entropy as the metric to find the optimal spilt. Generally, we choose $\sqrt{p}$ as default value of m. p represents the number of predictors. James Gareth et. al. [14] advised to select smaller value of m if the features are highly correlated.

Number of trees(n) and size of the random set(m) are considered as the good tuning parameters to increase the overall efficiency of the random forest model. Therefore, we attempted to choose the optimal value of n and m to improve overall accuracy of our model. Additionally, we tried to adjust the depth of tree to reduce the computational pressure. 'Gini' impurity metric was used as an alternative to class entropy to evaluate the goodness of variable as a spilt. It is calculated as follows:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Where, $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from k^{th} class

4.2 K-Nearest Neighbors(KNN)

KNN is one of the simplest and well known instance based learning algorithm. In this approach, no internal model is built from the training data. Instead, we simply estimate the distances of testing points from all the train data. We then considers k nearest neighbors which are close to the unknown instance. It either uses the majority voting or conditional probability estimates to identify the label of unknown instance.

We experimented with multiple distance metrics such as Jaccard, Sokalsneath etc for measuring the distances between categorical data. For numeric features, we used Euclidean distances between data points. We tuned k considering the large class size of the response. Binary distances are calculated as follows:

- $D_{Jaccard} = \frac{|A \cup B| - |A \cap B|}{|A \cap B|}$
- $D_{Sakalsneath} = \frac{|A \cup B| - |A \cap B|}{|A \cup B| - (|A \cap B| + 0.5 * |A \cap B|)}$

4.3 Naive Bayes(NB)

Naive Bayes algorithm is the probabilistic approach to solve the classification problem. It uses the Bayes theorem which is based on conditional probability theory. The Bayes theorem is mathematically represented as follows:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Where,
$P(C|X) =$ Posterior probability of class C
$P(X|C) =$ Class conditional probability
$P(C) =$ Prior probability
$P(X) =$ Probabilities of observing values for input features

This algorithm also assumes features are independent of each other. After making the this assumption, it models the probability of instance belonging to a particular class as the product of probabilities of features given its true class. Mathematically, it is represented as follows:

$$P(C|x_1, x_2, ..x_n) = P(C)\prod_{i=1}^{n} P(x_i|C)$$

Where, $x_1...x_n =$ predictors set

Using the above approach, it estimates the probabilities for all the classes and then predict a class with high probability as an outcome. The probabilistic methods are proven useful to analyse the categorical data. Therefore, we selected this algorithm to train our model considering many categorical features in our data. We used both Multinomial and Gaussian Naive Bayes to solve this classification problem.

4.4 Logistic Regression(LR)

Logistic regression is linear approach to solve the classification problem. It assumes Binomial or Multinomial distribution of the response. During this modelling technique, we model the odd ratio of an instance belonging to a particular class as linear combination of predictors. Probability of instance belong to class c_i is then calculated as follows:

$$P(C = c_i) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdot\cdot+\beta_p x_p}}{1 - e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdot\cdot+\beta_p x_p}}$$

We considered both the variations of logistic regression i.e. Binary and Multinomial classifiers considering the nature of our response. For the binary approach, we ensemble all 100 binary classifiers to predict the final outcome.

4.5 Boosting Classifiers

Boosting technique is very similar to bagging approach since it ensembles multiple decision trees. However, it uses the novel approach to construct these decision trees. These trees are build sequentially. It analyses misclassified observations in each iteration before constructing the next tree. Then, next classifier focuses on the misclassified observations from the previous tree. This process greatly improve the accuracy of the model. In boosting, we consider weighted sum of all the trees instead of considering the majority votes. We used Adaboost and Gradient boosting Machine (GBM) as our boosting classifiers. Both uses the different techniques to learn from the previous classifiers.

Initially, Adaboost assigns equal weight to all the training points. For the next iteration it adjusts the weight of the miss-classified training points. Subsequent tree then focuses on correctly classifying the previously miss-labeled data. All the successive trees follow same mechanism to correct the previous errors. This characteristic of Adaboost improves the overall accuracy of model. Whereas, Gradient Boosting Machine(GBM) uses slightly different to approach to learn of the previous classifiers. Instead of assigning weights, it uses gradient descent optimization technique to minimize the value of differential loss function.

We used both the techniques as our last approach to improve the model accuracy. There are multiple parameters one could apply to tune the AB and GBM models. We attempted to adjust our learning rate and number of trees to boost the model accuracy.

4.6 K-Fold Cross Validation

We used the Cross Validation to analyse the performance of our models. In this method, we divide the data into K folds such that, at each iteration we use subset of data as train (D_t) and remaining data as a validation set(D_v). Technique trains the model only using (D_t) and uses (D_v) to validate the results. We could only used 5 and 3 fold cross validation due the computational limitation.

4.7 Mean Precision Score at 5

In multi-class classification problem, when the class size is very large the mean accuracy score is very strict criteria to evaluate the model performance. Therefore, we generally used MAP@K metric to measure the performance of the model. K indicates any real number based on the nature of the problem. Kaggle decided to use the mean precision score at position 5(MAP@5) as their evaluation metric for this competition. Given this, they challenged participants to predict 5 relevant hotel clusters for each user. We take mean of all the average precision score for n user in the test set to calculate the final prediction. The average precision score at 5 is calculated as follows:

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

Where,
P(K) = the precision at cut-off k in the item list
n = number of predicted hotel clusters
N = size of test sample

5 Feature Engineering

Feature engineering is an essential step for training a good predictive model. The most effective way to train the models is by providing a structured dataset. Multiple threads on the competition forum pointed that feature engineering was an important factor to improve the model accuracy. Therefore, it was prudent to dedicate quality time to generate new features based on the raw data. We considered following approaches:

5.1 Features using Discretization Process

Most of the machine learning algorithms need the data in a numeric form. 'One-hot encoding', Dummy variables and other similar scheme can be used to recode the categorical data in the numeric form before feeding it to a model. Although, it was difficult to directly apply these methods to the date or time variables due to its nature and with the limited number of features, it was decided not to exclude this piece of information to train our models. Another reason behind creating these features was to ensure that seasonal popularity, particular time in week or duration of stay etc. are taken into consideration as they play an important role while making the travel or hotel booking decisions.

The feature set created for this project included three date variables which consist of date_time, srch_ci and srch_co representing time when user performed search query and their check-in and check-out dates. The date and time parameter was split into i.e. day , month, year, hour. Features such as part_of_day, type_of_day, season were created and duration_of_stay was computed using check-in and check-out date. Following this addition, variables srch_adults_cnt and srch_children_cnt were converted into a single binary feature 'is_alone' which was used as an indicator to identify whether the user is travelling alone or with additional passengers.

5.2 Features using Booking/Click History

This feature was replicated based on the kaggle competition winning team's approach [10] and was created from the variable 'is_booking' and combination of user, hotel and searched destination attributes. The rationale behind extracting these features was to represent similarity among users with common characteristics based on their click/books history. For instance, id of countries or cities helped to summarize the booking trends, user's nature etc. that we considered could play a significant role for predicting the response. It was seen that the model learns well when such features are introduced along with the ID attributes. We used rate statistics on booking/clicks events based on the multiple factors. For example, to measure the similarity between the users from same region, we grouped them by their location and calculated booking and clicks rates. Additionally, this data included ID features which were nominal features with high-cardinality. ML algorithm often discards these features while training the models. Many participants suggested that one of the solution to utilize this feature effectively was to introduce the rate variables based on the response and booking outcome. Hence, we used this information while training our model even if the training model has a tendency to discard such categorical features.

Weighing scheme on books/click counts was applied next to gather information about the popularity of variables such as city, destinations etc. It was based on the assumption that high count of booking/click event represents popularity of particular variables. We estimated these counts by simply summarizing the data based on the raw attributes. Once, this is task was completed, we multiplied clicks by a random number x and books by y such that $y > x$. For instance, to calculate the popularity score for a particular search destination the data was grouped by search destination to estimate the total number of books and clicks for all search destination. This was the milestone where weighted sum of both the counts could be taken.

5.3 Features Using Principal Component Analysis(PCA)

Destination dataset is highly dimensional. It was not a feasible approach to introduce all the 149 features for training the model. Additionally, it was found that these variables were highly correlated to each other during initial data exploration [Figure-4]. Similarly, due to the possibility that the high dimensional data may impede the models performance, we considered not using all the 149 features for training the model.

Based on these constraints, it was decided to use PCA to transform these variables. This technique uses the orthogonal transformation of original feature space to find new variables which are linear combinations of the old feature space.We defined these components as principal components of our model. After applying the principal component analysis technique on these latent features, we observed that the first three principal components cumulatively explained around 95% variance in the destination data which is often the case. This techniques also helped to reduce the dimensionality of the data which could have led to the model over-fitting problem.

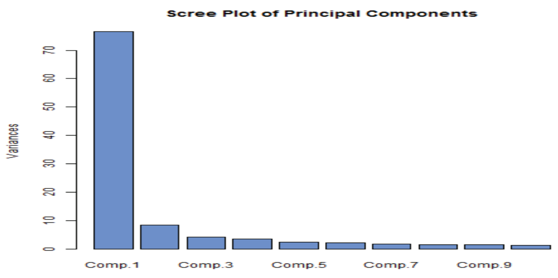


Figure 5: Variance explained by first 9 principal components

5.4 Features Using Hierarchical Clustering

We introduced clustering technique to map the users, hotel and search destinations attributes to the new feature space based on their similarity. This new feature space effectively helped in learning the supervised model [15]. These initial data exploration findings show similar pattern of hotel booking for the variables mentioned. Additionally, we considered hierarchical clustering since it's suitable for the categorical data. Sklearn clustering library which implements Agglomerative clustering algorithm was then used to find similar groups. Agglomerative algorithm created the clusters in top down approach. This implementation of hierarchical clustering is scalable for large samples. 'cosine' distances ($\cos(\theta) = \frac{\vec{a}.\vec{b}}{|a.b|}$) were then used to measure the similarities between these attributes. Following these steps the 'cluster IDs' were introduces as a new feature.

6 Data Preparation

Train and Test Selection

We explored the original train(tr_O) and test (te_O) data to analyze Kaggle's approach to partition the Expedia data for this competition. tr_O represented the user's hotel search activities for the period 2013-2014. The booking events of these users for the year 2015 were included in the set te_O. However, te_O showed around 10-12% new values of cities and destinations. We tried to emulate the setting for the generation of new train and test data to validate our result. We divided the original train data (tr_O) into two subsets as shown in the following diagram. We tried to balance the click and booking events while choosing the random sample. For the same purpose, we randomly selected booking events for around 38,7825 users. And clicks data is included for around 63,4903 users. Decision to keep some imbalance in the book/click data has made to imitate the real scenario. Later, we combined these two samples and again randomly selected data for around 23,5750 to further down-sampled this subset. The final subset obtained following this technique is used as the train set. For the test set, we randomly selected year 2014 booking information for the users in the selected train set. Our test set constituted the booking data for around 30,000 users.

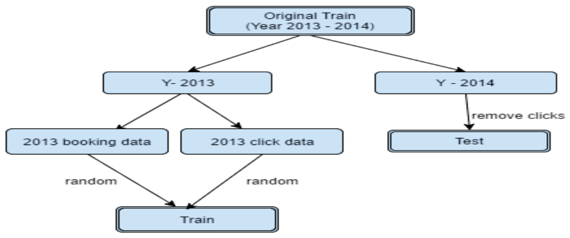


Figure 6: Train and Test Data Selection

In the next step, we applied feature engineering methods described in section-5 and created 35 new features as follows:

- Feature using discretization and principal component analysis process
  - year, month, day, hour, part_of_day, type_of_day, season, duration_of_stay, is_alone (set-D)
  - dest_feature_pc1, dest_feature_pc1, dest_feature_pc1 (set-pca)
- Feature based on Historic book/click rate and weight-age scheme
  - 12 click/book rate variables based on hotel, users and search destination attributes(set-rateVars)
  - 6 variables using feature book/click weighing scheme(set-popVars)
- Features by clustering technique
  - 3 variables created using hierarchical clustering on hotel, user and search destination attributes (set-clustVar)

We replaced the missing values for origin_destination by -1. The missing values of duration-of-stay, seasons, is-packages and is-alone were imputed by their respective modes. The high-cardinal id features such as user_id, search_destination were encoded using the label encoding system which normalize the labels values between the 0 to class size. Categorical features with the low-cardinality were encoded using the dummy variable scheme.

7 Model Building Approach

The primary objective for model building was to follow a sequential approach to find the candidate models. As our first step, we analyzed the performance of multiple machine learning algorithms on the raw data followed by application of feature engineering, parameter tuning and ensemble methods to improve the model performance. This was carried out by extensively used Python machine learning library 'sklearn' to build our models.

We chose Random Forest as our first choice as it is very effective for solving the classification problem. The process described in the article [12] also proved to be catalyst in making this decision. Although we could not see significantly acceptable results in the author's study, we continued to explore build model using this method with the hope to improve its performance by effective feature engineering and combination of other ML algorithms.

K-nearest neighbors technique was considered next for review and to analyze the performance of instance based machine learning on this classification problem. The main advantage of using KNN was its insensitivity to outliers. Additionally, it used similarity metric to make the prediction. Rationale behind using this technique was to consider the similarities between the users to predict their preferences of hotel booking. We applied several binary and numeric distance metrics to measures the similarity between user's search parameters.

Next step to our model search was to try the parametric machine learning algorithms which assumes that the response follows certain distribution pattern to make the prediction. Therefore, we used Naive Bayes and Regularized Logistic Regression and made an attempt at the probabilistic approach for solving this challenging problem.

During the final step, we explored the family of boosting algorithms which included Adaboost and Gradient Boosting Machine(GBM). We decided to experiment these techniques since we observed that the winning team and other top 100 teams employed boosting techniques such as XGboost as their modeling method. We opted out to explore Support vector Machine model further due to our computational limitation.

7.1 Random Forest Model(RF)

The first random forest model for this project was implemented using all the original features expect the time variables. We considered the ensemble of 5 decision trees to make the final prediction. In addition to this, all the predictors for the random subset selection were used instead of limiting the selection to only $\sqrt{no-of-predictors}$. Random Forest technique uses the out-of-bag error estimates to choose the best split for the construction of trees. Based on these errors, it estimates the Gini indices for the predictors to identify the important features. We took advantage of this characteristics in our first model as a mechanism to identify significant features. Variables such as hotel-continent, hotel-market, hotel-country, is-package, search destination, destination became prominently important during the tree building process. We evaluated the model performance using 5-Fold cross validation which generated the average CV score of approximately 5%.

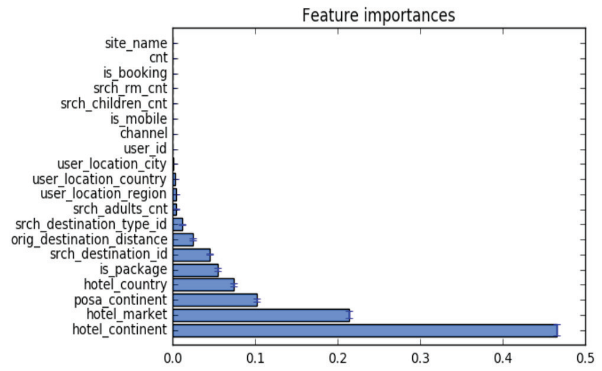


Figure 7: Variable Importance Plot - Random Forest model with original variables

Following this process, we selected top eight variables to build our next iteration model. Model trained with these significant predictors only improved our cross validation score by 1.2%. Hence, new engineered features were introduced in a sequential manner to analyze the contribution towards model accuracy. The subsequent random forest model was trained using the discretized features(set-D) and Principal components(set-pca). This model did not produce any noticeable changes to the prediction accuracy. However, all the principal components and duration-of-stay were significant. In next stage, we implemented random forest model using the combination of Set-1 and original important features to analyze its cumulative effect. This step increased the cross validation score to 6.8%. A total of 2 to 3% increase was seen on cross validation of large train samples.

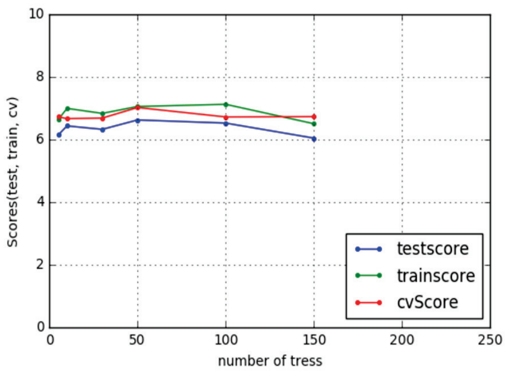
The process continued as we used historic book/click rate variables(set-rateVars). Performance of this model was less effective than the previous fitted models. Hence, it was decided to discard these features from the random forest model due to its potential adverse impact on the model accuracy.

The next model was trained using only popularity variables(Set-popVars). This gave us the mean prediction accuracy of 7.9%.

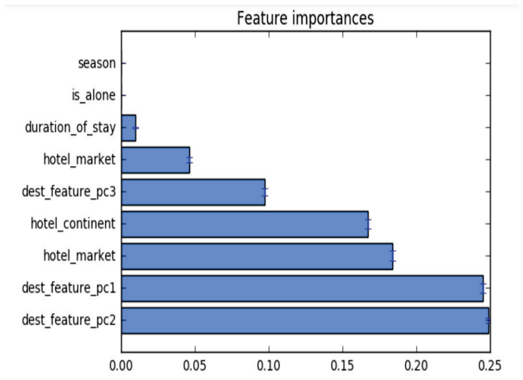
As the single random forest model gave unfavorable results, we considered using a binary random forest model for the next step. We fitted 100 binary classifiers considering our target as 'whether a particular hotel cluster is booked (1) or clicked (0)'. We aggregated the success probabilities of booking all the 100 clusters and chose five hotel clusters which had the highest probability outcome. MAP@5 precision score using this approach varied between 2-5% for the training samples of various sizes.

In final step of model selection, we chose two random forest model candidates, one trained using the combination of set-D features with original important features and the other implemented with variable Set-popVars. This led us to introduce unused variables in a sequence to observe its effect on the model accuracy. During this process, we dropped the new variables based on whether the introduction of that variable decreases the overall cross validation score or not. Unfortunately, no improvement in the accuracy of the random forest model was seen. As the final step of model improvement, we attempted to tune parameters such as number of trees, depth of the tree and split criteria. This identification led us to believe that the ensemble of 30-50 decision trees can improve the overall test accuracy of the model and give a stable cross validation score.

We evaluated the models performance using 5-Fold cross validation. The average CV score was around 5-6%. Hotel attributes such as hotel continent, market, country and id associated with the site were identified to be important during tree building process.

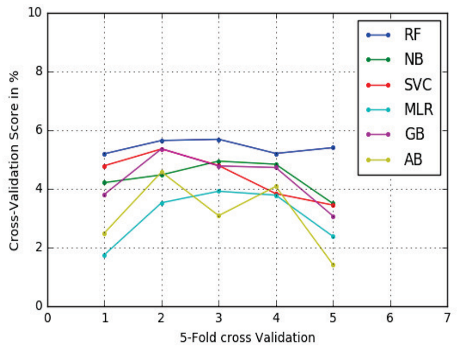


(a) Accuracy score for candidate RF Model-1 using different tree sizes

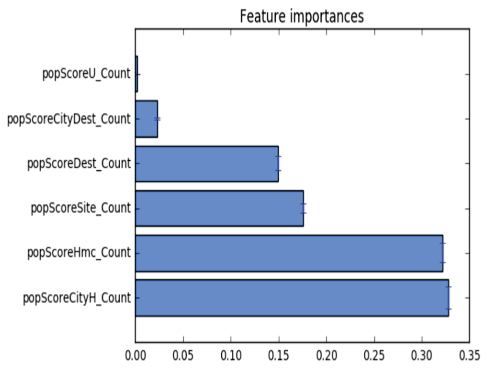


(b) Variable Importance Plot- Candidate Model 1

Figure 8: Evaluation of Final Random Forest Model



(a) 5-Fold cross validation scores of the models trained with original features



(b) Variable Importance Plot- RF Candidate Model 2

Figure 9: Evaluation of Random Forest Candidate Model

7.2 K-Nearest Neighbor(KNN) Models

We used K-nearest neighbor classifiers to fit 4 different models based on hotel, user and destination attributes. We employed this technique since it's not advisable to use KNN model on large features space due to it's computational complexity. Another reason choosing these variables is a similar hotel booking preference among the user having same search parameters. For example, chance of users booking a same hotel cluster if they are considering the same destination are more. Assuming this, we fitted four models based on different combination of these attributes. We applied distance metrics such as Jaccard, Cosine, Sokalsneath to measure the similarity between labeled and unlabeled instances.

KNN modeling with Nominal features

For first KNN model, we used user's city, country, region and seasons. The primary idea using these variable set is to capture similarities among users based on their current location and seasonal hotel booking behavior. We considered Sokalsneath distances for calculation of the distance matrix as it gives overall good accuracy for test data than the other two. We selected 10 nearest neighbors for making the final prediction. We applied uniform weighing scheme which treats each neighbor equally using their distances. It accurately predicted around 5-3% test instances with the use of random selection of training sample. We observed that kNN is sensitive to train sample size. A large sample size gives good test score accuracy. However, computationally its very inefficient. We used sklearn's implementation of K-nearest neighbors to build this model.

For the next iteration, we used hotel's continent, country, season and day. Using this set, we aimed to capture the likelihood of users booking the similar hotel when they travel to different country considering seasonal popularity's of the search destinations. In later step, we fitted several models using the different combination of these attributes to improve the accuracy of the models. Our rationale fitting these different models is to find multiple weak KNN classifiers and ensemble them at the final step. We evaluated the performance of KNN using 3 fold cross validation due to the computational limitation. The average score for the these models was under 2%. We also aggregated predicted outcomes and evaluated its model performance using map@5. Cumulative accuracy on the holdout set was 2.35%. Therefore, we didn't consider these models during the final ensemble approach because of its poor performance and biased result on different random train selection.

KNN modeling with Numeric features

Next, we implemented KNN model using the numeric variables. For first model, we used features created using the historic books/click rate(set-rateVars). We used Euclidean, Manhattan metrics for the calculation of distances between train and test data points. However, for this model we restricted our choice of number of nearest neighbors to five. Another KNN model was built using the popularity variables(set-popVars). Individual accuracy of these model lies between 5-6%. Mean precision score@2 after combining the predicted outcome was 7.34%. KNN didn't provide scope to improve the model performance further by parameter tuning apart from the number of the neighbors and distance metrics. Hence, we considered this model as one of our candidate models for the final ensemble step.

7.3 Naive Bayes Model

As a next step, We applied Naive Bayes machine learning which employs the probabilistic theory to solve the classification problem. It uses the Bayes theorem to predict an unknown response from the posterior probability distribution of the feature space. Naive Bayes hypothesis of considering parameters independence is unrealistic. However, cumulative posterior probability helps to predict the unknown variable in many cases. Assuming this hypothesis is suitable in our case, we fitted our first Naive Bayes model.

We considered probability distribution of user-region, hotel parameters, principal components and binary variables to model our response. We experimented with both Gaussian and Multinomial distribution of parameter space. Gaussian Naive Bayes helped to get the mean CV score of 8.65%. We evaluated its performance on testing set. It accurately predicted around 7.18% test samples. However, Naive Bayes model with Multinomial distribution assumption of parameter space gave us only mean accuracy of 1.8%. Hence, we selected a Naive Bayes model with Gaussian distribution as one of the candidate model.

7.4 Logistic Regression(LR)Model

Logistic regression is one of the popular machine learning algorithm to solve the classification problem. We employed this technique after observing the good performance of parametric algorithm. It uses completely different approach to predict the response. It assumes probability of a response belong to a particular category follows a binomial distribution and model its odd ratio as the linear combination of the feature space. In that way, its similar to the linear regression. We used sklearn's implementation of regularized logistic regression to build our LR models. Considering the large class size, we applied oneVsRest strategy to predict the correct outcome. Using this approach, we fitted 100 binary classifiers and modeled our response as customer books a particular hotel cluster Vs he doesn't book that hotel-clusters. At the end, class label with highest probability value is selected as the final outcome. We correctly predicted about 9% test labels using this technique. Additionally, our 5-fold CV score reached to 9.78%. This training model uses 'liblinear' implementation of l2 penalties to solve the optimization problem. We used 100 iteration for this regularized logistic regression model to predict the final outcome. Logistic regression with multi-class approach was also attempted which uses the logistic loss to solve the optimization problem. LR model with oneVsRest scheme was chosen as the candidate model after observing the accuracy scores.

7.5 Modeling using Boosting

We used boosting techniques as the final approach to solve the problem. We explored two algorithms, Adaptive boosting(Adaboost) and Gradient Boosting machine (GBM) from the family of boosting algorithms. These two algorithms differ from each other by their learning techniques to boost the accuracy of final ensemble. Adaboost targets the miss-classified data in each iteration. Whereas, Gradient boosting tries to optimize the arbitrary differential loss function in a sequential manner. Number of decision trees greatly improves the accuracy of both the algorithms with the computational trade-off.

Our first Adaboost model was built using the combination of principal components and categorical features. While we trained the second model with the numeric features. Ensemble of 50 decision trees was considered to build these models. First model was selected for the final ensemble after analysing its prediction accuracy. It predicted almost 7.38% labels correctly. Subsequent model was implemented using gradient boosting technique. We considered same set of the variables to train the GBM model. Using this, the test accuracy increased by 2%. Due it computational complexity, only ensemble of 40 decision trees was used for identifying the unknown labels.

7.6 Ensemble of Best Models for the Final Prediction

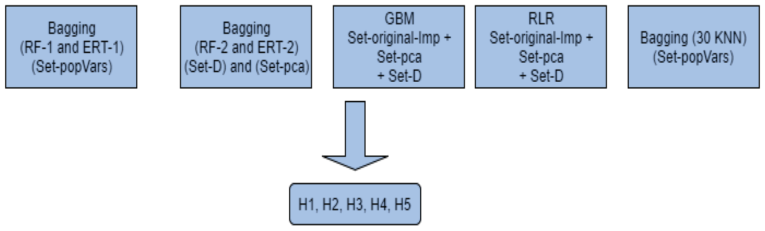


Figure 10: Final Prediction

The section explains our approach to predict the final outcome, i.e. list of five hotel clusters for each user event.

We considered ensemble of Random Forest(RF) and Extra Randomized Tree(ERT) classifier to improve the accuracy of the candidate RF models. ERT is very similar to random forest. The only difference lies in the selection of random split. Unlike random forest, it doesn't estimate the goodness of variable as the split. The selection is completely random choice. Therefore, it helps to reduced the variance of the model further and increase its accuracy. We selected two RF candidate models and implemented the other two models applying the Extra randomised tree(ERT) technique on the respective feature space. Finally, bagging technique is applied to combine the respective random forest models with their ERT variants. We used sklearn implementation of VotingClassifier to achieve this. The average of predicted probabilities was considered to predict the class label. The predicted outcomes of these two ensemble represented the two hotel clusters recommended to the users.

Following this, bagging method was applied to ensemble 30 k-nearest neighbors classifiers to boost its performance. The final predicted values of hotel cluster represent our third recommended hotel cluster. We also used same technique to combine the Adaptive boosting model and Naive Bayes model. However, the cumulative effect did not give us good CV scores. Hence, we did not consider these models for making the final prediction. Instead, Logistic regression and GBM models were used to predict the last two hotel clusters considering their good performances. Aggregation of these five hotel clusters represented our final five hotel clusters for each instance in the test set. We could achieve MAP@5 score of 16.75% using final ensemble approach.

8 Results and Discussion

Multiple models were built using the combination of parametric and non-parametric classification algorithms as a part of this project. Following this build up, we used the ensemble of candidate weak classifiers to recommend the hotel clusters for the Expedia users. The rationale behind using ensemble of the assorted models was to choose different base learners that complement each other's[15]. This ensemble predicted 16.75% test labels accurately considering the overall individual model accuracy of 8%

The decision to build our first model based on initial selection of Random Forest Techniques proved to be the most useful strategy to effectively analyze the efficiency of similar tree-based classifiers. By utilizing this algorithm for our variable selection approach, we greatly reduced the initial feature space. It also helped to identify features which were significant to its unique concept of out-of-bag error estimate. This opened up the possibility to implement multiple models using other supervised machine learning techniques without spending additional time on this step. We could improve the accuracy of simple Random forest model from around 4% to 8% by introducing the engineered features. After multiple iteration of training and testing different samples, we found two candidate random forest models trained on multiple feature space. One of the two models was trained using the popularity variables created by employing method discussed in the section 5.2. Using this model, we predicted around 8% of unknown test instances correctly. The second candidate RF model was trained using the combination of features introduced by technique discussed in sections 5.1 and 5.3. Initial results on the second model gave the test accuracy score of 6.3%. However, as we started increasing the training sample size, the second model increased the accuracy level to 8.3%. The ERT models built on their respective feature spaces outperformed their corresponding RF models. We could achieve the prediction accuracy of 9.8% and 8.9% respectively using ERT models. Therefore, in the final step we combined candidate RF models with their counterpart ERT variants to further improve thier individual performances. Alternatively, it was seen that the Adaboost model which was trained using the set-popVars predicted only 4% user-events correctly. Hence, we did not considered Adaboost for our final ensemble due to its inconsistent performance on different samples. As the GBM model attained 9.7% prediction accuracy, we decided to tune it even further to boost its accuracy.

No noticeable improvement was recorded in the learning rate of tree based algorithms even after employing various feature engineering techniques. We think the high-cardinal id features and large class size of the response contributed to the poor performance of tree-based classifiers. One possible solution for improving the accuracy is to construct many features using multiple statistical methods on the high-cardinal feature space.

During the training simulations, we observed that the K-nearest neighbors performance on the categorical features was very poor. However, it predicted around 7% testing labels correctly with the model trained using the numeric features. Accuracy score increased further by 1 to 2% after increasing the training samples. The technique to find optimal K value proved to be computationally very expensive. As a result, we considered only 5-10 neighbors to make the final prediction. We believe that the optimal k could have improved the accuracy further considering 100 possible choices for identifying the correct class.

Regularized Logistic Regression model trained using the combination of hotel attributes and features set-D and set-pca outperformed the random forest model. It predicted about 8.8% of the test labels correctly.

In our observation, the OneVsRest strategy, a unique combination gradient decent optimization and the regular logistic regression contributed to its good performance. Naive Bayes model performed poorly compared to the above described parametric technique. Based on the results, we believe that it's assumption of feature independence doesn't hold true for this case. This may have been a factor for its poor performance.

As a logical next step, we analyzed how engineered feature contributed in searching a good predictive model. Prediction accuracy improved by 2-4% after the introduction of principal components and discretized features. We observed an additional overall 2-3% increase in prediction accuracy after incorporating the cluster ids as the features. However, this technique wasn't scalable for the larger training sample. Random forest model trained using the features discussed in section 5.3 helped us to get 7-8% prediction accuracy. It was also observed that features created using historic booking/click rate did not add to the accuracy of the model.

Table-1 : Candidate Model Performance

Model	Algorithm	5-Fold Cross Validation Score	Test Score
1	Random Forest - 1	[0.067, 0.07 , 0.070, 0.07, 0.06]	0.079
2	Random Forest - 2	[0.068, 0.074, 0.073 ,0.07, 0.06]	0.089
2	KNN	[0.06, 0.57, 0.64]	0.072
4	Naive Bayes	[0.0856, 0.088, 0.089, 0.087, 0.080]	0.069
5	Regularised LR	[0.089, 0.097, 0.096, 0.096, 0.090]	0.082
6	Adaboost	[0.082, 0.072, 0.062, 0.078, 0.074]	0.042
7	GBM	NA	0.086

Table-2 Test Accuracy of Final Ensemble

Ensemble	Test Score
RF and ERT	0.10
RF and ERT	0.089
Bagged KNN	0.072
Adaboost and NB	0.069

9 Conclusion and Future Work

During the course of this project, we applied some of the most popular supervised learning methods from the diverse family of machine learning techniques to predict the hotel booking preferences for the Expedia users. We improved our prediction accuracy by 8% using the ensemble of Random Forest, Extra Tree Classifier, Gradient Boosting Machine (GBM) and Regularized Logistic Regression techniques. It can be considered that the approach of combining such assorted week classifiers is the key to build a powerful predictive model.

The project also demonstrated the use of Principal Component Analysis to engineer new features and as an essential tool to deal with the high dimensional data. We have shown usability of hierarchical clustering for handling the high-cardinal categorical features. However, we need scalable method to apply this technique on large training sample. We found that the feature engineering component was essential for the competition supposing the atypical nature of this data. The top hundred teams spent fair amount of time on the feature engineering process. The winning team created more than 100 new features to solve this challenging classification problem. As the part of this project, we also employed combination of balanced and random sampling to create representation of the original data to train our models. We believe a good sampling technique is the key to handle massive datasets. The large size of the data used for our project posed many restriction while tuning the model parameters for the powerful machine learning techniques like GBM and Adaboost.

Many other powerful techniques such as Factorization Machines (FM) and Extreme Gradient Boosting (xgboost) were considered for use for this project but were not implemented due to the limited availability of time and computational resources. One possible direction for the future work is to explore FM to deal with the high-cardinality features. The winning team used 100 factorization machine models to learn the interaction between ID features and 100 hotel clusters to extract the useful features. According to their synopsis, this tool significantly helped them to get a high level of prediction accuracy. To take this one step forward schemes like the 'Leave-one-out' should be explored to encode the high-cardinal ID features which was suggested by Owen Zhang[22][21] and referred to as computationally efficient alternative to one-hot encoding. While reviewing the Kaggle competition briefings, we observed that many teams converted this classification problem into ranking task and used XGboost Ranking Algorithm to predict the hotel clusters. Hence, it is recommended that teams carrying out future work on this topic should investigate this unique approach to solving the challenging multi-classification problem faced on this project.

10 APPENDIX

- Python Libraries : sklearn, pandas, numpy, math, ml_metrics
- R-packages:lubridate, dplyr
- Tools and Services: Jupyter notebook for python, R-Studio, AWS

References

[1] Kaggle - Expedia Hotel Recommendations

[2] Kaggle - Data Source Expedia Hotel Recommendations

[3] Kaggle - Discussion forum Expedia Hotel Recommendations

[4] Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.

[5] Basu, Chumki et.al. ”Recommendation as classification: Using social and content-based information in recommendation.” Aaai/iaai. 1998.

[6] Hu, Ya-Han, et.al Hotel Recommendation System Based On Review and Context Information: A Collaborative Filtering Approach, June 2016

[7] Personalize Expedia Hotel Searches - ICDM 2013

[8] Jiang, Xinxing, Yao Xiao, and Shunji Li. ”Personalized Expedia Hotel Searches.” (2013).

[9] Kaggle’s Discussion Forum for ’Personalize Expedia Hotel Searches’

[10] First winner solution summary for ’Expedia Hotel Recommendation’

[11] Leaderboard Score ’Expedia Hotel Recommendation’

[12] Tutorial Expedia Hotel Recommendation

[13] Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001

[14] James, Gareth, et al. An introduction to statistical learning. Vol. 6. New York: springer, 2013.

[15] Ethem Alpaydin, An introduction to machine learning. 3^rd edition. The MIT Press, 2014.

[16] H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS

[17] scikit-learn developers, 2016, sklearn User guide

[18] Kaggle’s discussion on Random Forest model

[19] J. Friedman, Gradient Boosting Model, 1999

[20] Harrington, Peter. Machine learning in action. Vol. 5. Greenwich, CT: Manning, 2012.

[21] Leave-One-Out Encoding, Owen Zhang

[22] Owen Zhang