CS 229/STATS 229 Spring 2016 Project Report

SUNet IDs: jcingel@stanford.edu, puzon@stanford.edu
Names: Jarrod Cingel, Liezl Puzon

Predicting Expedia Hotel Cluster Groupings with User Search Queries

Abstract

In this paper, we aim to create the optimal hotel recommendations for Expedia.com customers that are searching for a hotel to book. We will model this problem as a multiclass classification problem and build variations of classic support vector machines (SVMs) and decision tree classifiers to predict the 5 most likely hotel groupings from which a user will book a hotel. We use feature selection techniques to select optimal feature subsets, then build a unique combined SVM and decision tree model that achieves a higher precision and recall than either individual model alone. The combined model is derived using a scoring technique that is based on the supplied evaluation criteria (Mean Average Precision @ 5).

1 Introduction

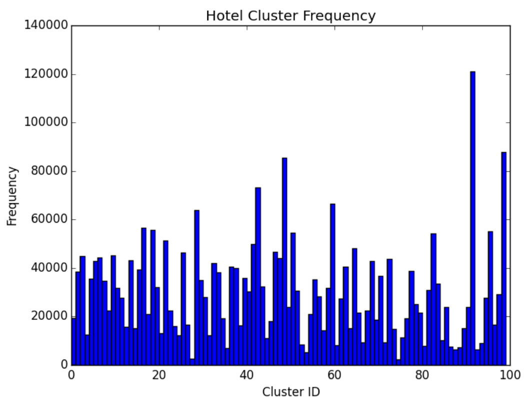
Our project is based on the Expedia Hotel Recommendations challenge on Kaggle.com (`https://www.kaggle.com/c/expedia-hotel-recommendations`). The goal of this project is to predict which of the 100 hotel clusters that a random Expedia visitor will book a hotel from. The high-level application of this project is to allow Expedia to provide the optimal personalized hotel recommendations for the user based on a user search event, which will increase the number of hotels booked through Expedia and simultaneously increase user satisfaction in the product. However, since the problem involves presenting optimal recommendations which the user is presented to choose from, this problem is not simply another multiclass classification problem. We must instead predict five hotel clusters that the user is most likely to book. The evaluation metric, mean average precision @ 5, evaluates each list of five predictions for each testing example. The MAP@5 scoring formula is as follows:

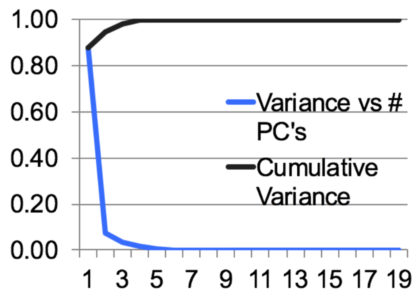
$$MAP@5 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{\min(5,n)} P(k)$$

where U is the test set, n is the number of hotel clusters predicted by the algorithm, and $P(k)$ is the precision in the list of clusters at cutoff k. With this scoring system, we are awarded a higher score for the correct cluster being as close to the front of the list of predictions as possible.

2 Data

The given dataset includes a training dataset with about 30 million examples and an evaluation dataset of about 2.6 million examples with hidden output hotel cluster. The 17 features shared by both the training set and evaluation set are listed in the table on the following page. Between each of the 17 individual features, we discovered no strong correlations. We then used principal component analysis (PCA) on the whole dataset (before appending the 3 principal components) and discovered that 99% of the variance was accounted for by 5 principal components, as pictured below.





Feature name	**Description**	**Data type**
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 int	double
is_mobile	1 when a user connected from a mobile device, 0 otherwise	int
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_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

2.1 Latent Features

The destinations file contains 149 numerically encoded features (labeled 1-149) which describe each possible destination. This makes intuitive explanation challenging because it is not given how these numbers were computed, or what the 149 latent features mean. Thus, we normalized the features then used principal component analysis (PCA) and discovered that 99% of the variance was accounted for by 5 principal components. The first three of these principal components were appended to the whole dataset via a join operation on destination_id to bring the total number of features to 20.

2.2 Downsampling

We filter out all training examples where the user did not actually book the hotel cluster that they clicked on. We do this because the actual evaluation dataset on Kaggle contains only booking events.

We begin by selecting 15,000 different user IDs out of about 500,000 unique user IDs randomly from the training dataset. We take all training instances from those user IDs and discard the others. This is because in the evaluation dataset, user events are preserved, and it would be irresponsible to arbitrarily discard certain events from the same user while keeping others.

2.3 Feature Selection

To select the optimal subset of features for our models, we chose to use a wrapper method with forward search using the average k-fold cross validation score. This allows us to maximize our models' prediction accuracy. We arrived at the same optimal subset of features for SVM and decision trees. These consisted of the last 5 elements in the table above: srch_destination_id, srch_destination_type_id, hotel_continent, hotel_country, and hotel_market.

The fact that these 5 features were selected as the optimal subset is quite surprising, since these are all related to the destination. Though the removed user-specific features - like number of rooms and length of stay - appear at first glance to be the defining characteristics of any user recommendation platform, they are actually far less indicative of booking outcome than destination-specific features. It is likely that introducing these extra features produced lower k-fold cross validation scores because of the resulting increase in model complexity. Furthermore, the principal components extracted from the latent destination features provided no performance gain and also added to model complexity.

3 Constructing a Benchmark

The expected value of the MAP-5 evaluation score for an output list of five hotel clusters produced by random guessing is one benchmark for evaluating our work. We calculate this benchmark as follows. Suppose we were randomly selecting 5 clusters for each testing example, at each position in our final 5 guesses. We get more points the closer to the front of our guesses the correct cluster is, and 0 if it is not present. This means that our MAP-5 score for random guessing would be:

$$\text{MAP-5}_{\text{random}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{5}\frac{{}^{100}P_4}{{}^{100}P_5}\text{score}_j$$

In the above, n represents the number testing examples, and score_j represents the score awarded for successfully guessing the correct cluster in the jth position. Note that the 100 comes from the fact that there are 100 possible hotel cluster assignments. We get 5 points if the correct value is in the first position,..., 1 if it is in the last position, and 0 otherwise. So we can define score_j as follows:

$$\text{score}_j = 5 - j + 1 = 6 - j$$

Substituting this into the above equation, we have:

$$\text{MAP-5}_{\text{random}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{5}\frac{{}^{100}P_4}{{}^{100}P_5}(6-j) = \sum_{j=1}^{5}\frac{{}^{100}P_4}{{}^{100}P_5}(6-j)$$

We solve the above for and we get $\text{MAP-5}_{\text{random}} \approx 0.15625$.

4 Supervised Learning

In order to generate improvement over $\text{MAP-5}_{\text{random}}$, we looked primarily to several different supervised learning techniques. These include Multinomial Naive Bayes, Multiclass Logistic Regression, Multiclass SVM, and Multiclass Decision Trees. We were able to discard the first two models based strictly on their poor training set accuracies compared to the last two. Finally, a novel convolution of the last two models allowed us to achieve a fairly high score.

4.1 Baseline Model Approaches

We first attempted to train a multinomial Naive Bayes classifier with Laplace smoothing. With the feature subset we selected, we achieve around 5% k-fold cross validation accuracy. With multinomial logistic regression, we achieve a 4.5% accuracy. multiclass AdaBoosted decision trees, multiclass gradient boosting, random forests all achieved a k-fold cross validation accuracy of around 2-3%. Although better than 1%, the random guessing benchmark of selecting predictions randomly of 100 possible hotel clusters, these models fall short of SVM and decision trees. We omitted extending these models to the case of 5 predictions per training example based on the fact that SVM and decision trees increased our k-fold cross validation accuracy by threefold and fivefold, respectively.

4.2 Multiclass SVM

The multiclass SVM is very time consuming, yet we hypothesized that given enough data, it can be a very accurate model. To test our hypothesis, we trained an SVM on a subset of the training data consisting of only one feature, the destination IDs. We use the downsampling method described in the Downsampling section above. This initial baseline was able to achieve an average of 23.7% training accuracy and an average k-fold cross validation accuracy of about 12.3%, which is very good considering that many training instances contain the same destination ID with different hotel cluster values. The average k-fold cross validation accuracy increased to 13.5% after using a subset consisting of the 5 optimal features as described in the Feature Selection section.

4.3 Decision Trees

The decision tree classifier has the advantage of being extremely efficient. Although one downside is that decision trees are subject to overfitting, our precautions of selecting an optimal subset of features counterbalanced the negative aspects of this model by reducing model complexity. Furthermore, most other models we tried required downsampling before performing k-fold cross validation; however, this model can use the partitions of the entire dataset for cross validation and still runs significantly faster than the others. Using just the destination_id feature, we achieved a training accuracy of about 17.1% and an average cross validation score of 16.3%. The average k-fold cross validation accuracy increased to 18.4% after using a subset consisting of the 5 optimal features as described in the Feature Selection section.

5 Model Modification and Combination

Once we decided which supervised learning methods to use, we needed to develop ways to 1) instead of classifying each training example with a single output label, produce a list of 5 predictions ordered by certainty to match the required submission format 2) combine their predictions to maximize the MAP5 evaluation metric by using a novel convolution of both classifiers.

5.1 Classifier Modification

First, we needed to modify both supervised learning tools to make 5 predictions, rather than 1. For the SVM classifier, this required a bit of intuition regarding the way the classifier works. The multiclass SVM we used needed to classify among 100 different possible cluster assignments. Rather than using 100 traditional one-vs-rest classifiers, we elected to use 1050 one-vs-one classifiers. Our intuition is as follows:

In order to make decide among n distinct classes, we need a total of $n\frac{n+1}{2}$ classifiers such that there exists a single classifier comparing each distinct i, j for every $i, j \in C$, where C is the set of possible classes. Once each classifier is trained, every classifier is used to make a prediction for a given test example. We initialize a dictionary containing a key for each possible class. Every time one of the $n\frac{n+1}{2}$ binary classifiers decides in favor of a single class, the value for the key corresponding to that class in the dictionary is incremented by 1. After all $n\frac{n+1}{2}$ classifiers have been tested, the dictionary is sorted in non-increasing order by value, and we retain the first 5 results in this list to use as predictions. The decision tree classifier is inherently multiclass, so only slight modifications were required to return 5 predictions.
Once this modification was done for each of the two supervised learning classifiers, we combined their outputs as follows. First, assign normalized weights to each classifier. The idea here is to give higher weight to the classifier with the smallest generalization error, assuming that its results are more likely to be correct than the results from the other classifier. We can calculate this constant for classifier i as follows:

$$c_i = \frac{1 - \text{err}_i}{\sum_{j=1}^{2}(1 - \text{err}_j)}$$

Once we have the constant for each classifier, we need to find the highest scoring 5 clusters out of the possible 10 we are given. We need to account for the fact that a cluster could be used twice, so we create a dictionary with 10 keys (1 for each cluster) whose values are initialized to 0. We iterate through each item in the list returned by each classifier, and update the cluster's dictionary value with a score calculated as follows:

$$\text{score}_i := \text{score}_i + c_i(5 - n_i)$$

In the above equation, c_i is the weighting constant for the list from the respective classifier, and n_i is the index of the element in the list, from 0 to 4, inclusive.

Once we have done this for each element, we sort the dictionary by value in non-increasing order, and then take the 5 elements with the highest values. Thus, our list is complete.

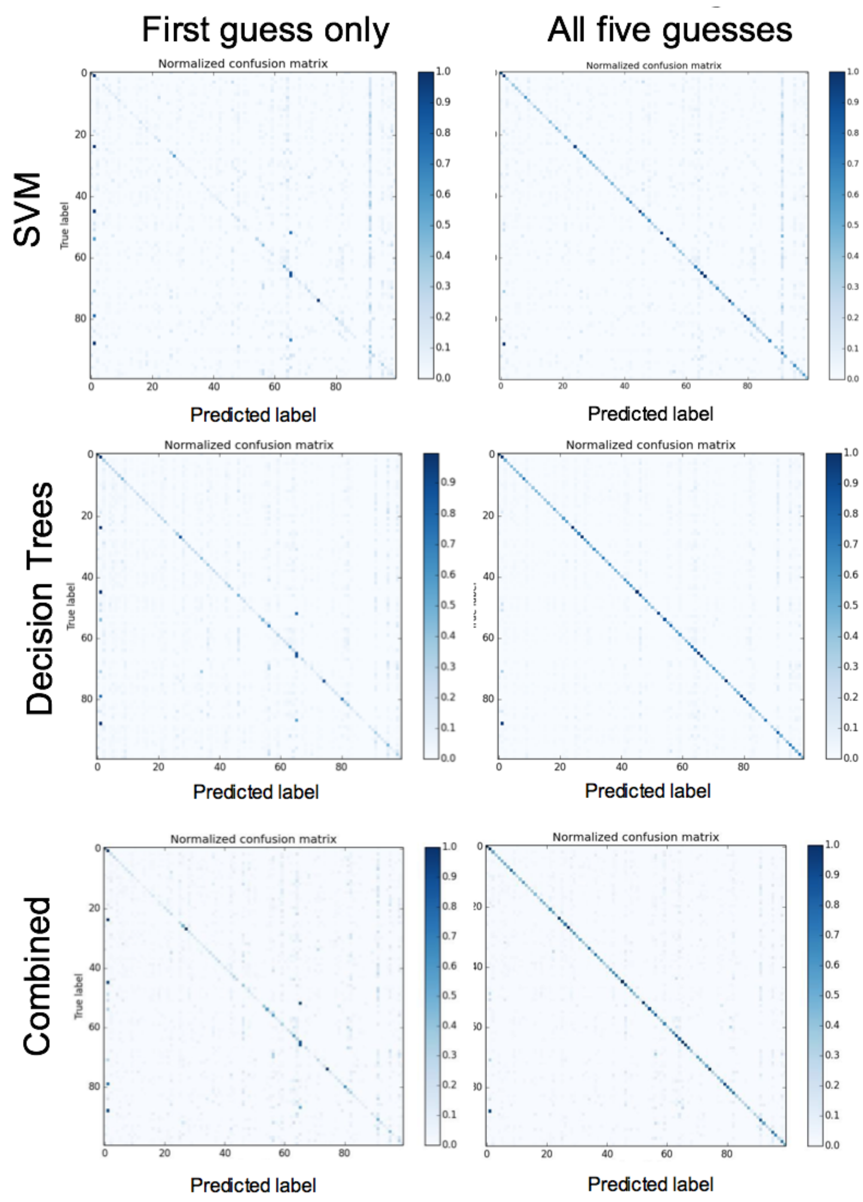
5.1.1 Accuracy on Validation Set

Before submitting to the competition by running on the entire testing set, we generated statistics to evaluate our model's success. Since the testing set does not provide the cluster values, we needed to do this using a validation subset we generated earlier from the training data. We ran trials for all three of our methods, including SVM, decision trees, and the convolution of the two described above. For each trial, we plot two normalized confusion matrices below. The first defines a classification as "correct" classification only when the true cluster value is the first value in the list. The second is more lenient, and considers a classification correct if it appears in the list at all. Here are the respective confusion matrices:

We can see that the optimal model convolution has higher average precision and recall than its two components, the SVM and the decision tree classifier, do alone. Verifying this, the next step was to run our algorithm on the test dataset and submit to Kaggle.

5.1.2 Submission Results

After running our algorithm on the entire test dataset and then submitting, we achieved a highest score of 0.38315 for the MAP@5 evaluation, with the highest score on the leaderboard being slightly over 0.5. Our results are quite good in comparison to other submissions considering that those submissions took advantage of a data leak, whereby some of the examples in the evaluation set were leaked into the training data without removing key identifying information about the example (orig_destination_distance and user_location_city could be used to exactly identify duplicates in the evaluation and training data).



Precision
Mean: 0.509
Range:
0.067 (id 91)
1.000 (id 24)

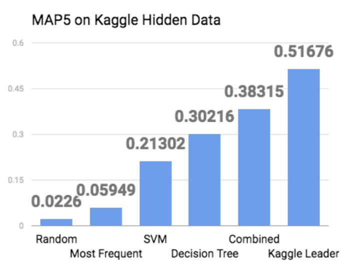
Recall
Mean: 0.413
Range:
0.029 (id 14)
1.000 (id 24)

Precision
Mean: 0.620
Range:
0.198 (id 71)
0.999 (id 24)

Recall
Mean: 0.525
Range:
0.048 (id 88)
1.000 (id 74)

Precision
Mean: 0.670
Range:
0.000 (id 24)
1.000 (id 35)

Recall
Mean: 0.533
Range:
0.000 (id 24)
1.000 (id 27)



6 Next Steps

In the future, with more time and computing power, we would like to apply other machine learning techniques, such as neural networks, and compare them against our current combined model. We plan to remove the filtering and test this method on the entire set of 30 million training examples. With such a large dataset, techniques like neural networks can be very powerful.

7 Works Consulted

1. `https://www.dataquest.io/blog/kaggle-tutorial/`
2. `http://scikit-learn.org/stable/modules/multiclass.html`
3. `https://web.stanford.edu/~hastie/Papers/samme.pdf`
4. `http://rob.schapire.net/papers/multiboost-journal.pdf`
5. `https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf`
6. `http://dml.cs.byu.edu/~cgc/docs/atdm/Readings/MLM-Overview.pdf`
7. `http://astro.temple.edu/~tud51700/pdfs/ijcai13.pdf`