Jack Cunningham

Machine Learning Report

Expedia Hotel Recommendations

12/10/21

Summary

The dataset investigated was provided by Expedia with the objective of predicting the type of hotel a potential customer would book. This problem is one that drives the travel technology industry, the sites that can give personalized deals and suggestions that match traveler desire are the ones that make money. Expedia has already broken hotels into clusters based on price, rating, and location. Clustering is used by Expedia as an approach due to hotels being similar and how suggestions are given to users. Since users scroll down and multiple hotels are being offered the use of predicting a cluster is better than predicting an individual hotel.

There are three spreadsheets provided in this competition. The first is destinations.

|  |  |  |
| --- | --- | --- |
| Feature | Description | Data Type |
| srch\_destination\_id | ID of search location | Int |
| D1-D149 | Description of region in search that are latent | Double |

The next is the training dataset which has 24 features. Ten are listed below for understanding the dataset.

|  |  |  |
| --- | --- | --- |
| Feature | Description | Data Type |
| data\_time | Time of search | String |
| site\_name | Domain ID of search | Int |
| posa\_continent | Continent ID of site\_name | Int |
| user\_location\_country | ID of customer country | Int |
| user\_id | ID of customer | Int |
| is\_package | Whether or not the sale involved a package. 1 if so, 0 if not. | Int |
| srch\_ci | Date of check in | String |
| srch\_co | Date of check out | String |
| is\_booking | Whether hotel is booked or just clicked on. 1 for the former, 0 for the latter. | Int |
| hotel\_cluster | ID of hotel cluster | Int |

The testing data is identical the training data in structure with the exception of two features. Those are hotel\_cluster and is\_booking. The objective of the testing data is to predict hotel\_cluster.

All three datasets share the srch\_destination\_id feature but there are more unique srch\_destination\_id values in the training and test sets due to newer hotels not being in the destination set.

Bench Marking of Other Solutions

The first solution investigated is by Dune\_Dweller. He begins by creating another data frame named aggs, it has srch\_destination\_id as its primary key and hotel\_cluster sum and count for initial features. They then sum the number of clicks for each duo of srch\_destination\_id and hotel\_cluster storing as a new feature called clicks. He also creates a new feature called “relevance” which is a weighted sum of clicks and bookings for each duo.

He also creates a new function called most\_popular which returns the most popular hotel clusters for all destinations, this is stored in most\_pop with srch\_destination\_id as an indexing tool. Then using the test data, he uses the srch\_destination\_id value to predict the matching most\_pop value as a prediction.

|  |  |  |
| --- | --- | --- |
| Feature | Methodology | Data Type |
| srch\_destination\_id | Taken from training data and used as the primary group formation | Int |
| hotel\_cluster | Taken from training data and paired with srch\_destination\_id as secondary group member | Int |
| bookings | Number of bookings for this srch\_destination\_id and hotel\_cluster duo | Int |
| clicks | Number of clicks for this srch\_destination\_id and hotel\_cluster duo | Int |
| relevance | Weighted sum of bookings and clicks to measure popularity of srch\_destination\_id and hotel\_cluster duos | Float |

The prediction had an accuracy of .30325.

This kernel is successful due to its simplicity but also creativity in creating new features. In the new data frame, the double grouping of srch\_destination\_id and hotel\_cluster allow for very specific metrics to be formed for each unique situation. The creation of relevance as a weighted sum is a smart decision as it allows for comparison between srch\_destination\_id and hotel\_cluster duos. Often times popularity drives end user choice. Although this method is not overly complex it does a good job of predicting and makes logical sense.

A second solution is from Ila Semenov. They stay on a similar path of using popularity for prediction but also add analysis using the feature orig\_destination\_distance in order to assist his prediction. They create a function called top\_five to find the top 5 most popular clusters based on srch\_destination\_id and hotel\_cluster pairs and also find the top 5 most popular clusters based on orgin\_destination\_distance. They then merge the two-resulting top 5s into a list of 10 and predict the value based on the most occurring value. This results in an accuracy of .3778.

|  |  |  |
| --- | --- | --- |
| Feature | Methodology | Data Type |
| dest\_top\_five | Using the top\_five function that was created this is ran by seeing the top 5 clusters for a given srch\_destination\_\_id. | List containing Int |
| dest\_top\_five1 | Using the top\_five function that was created this is ran by seeing the top 5 clusters for a given orig\_destination\_distance. | List containing int |

Accuracy is increased by the additional feature used. Since orig\_destination is an important variable to consider including it into the model will help prediction.

A third solution is from KLchang who used K nearest neighbors. The base of the model is similar to previous ones, using most popular lists for each group of srch\_destination\_id and hotel cluster and each group of orig\_destination\_distance and hotel cluster. Unlike previous models he creates a function called computeNearestNeighbor. It takes the merged dataframe of the previously mentioned popular results and training data and returns a predicted cluster. This results in an accuracy of .46987.

|  |  |  |
| --- | --- | --- |
| Feature | Methodology | Data Type |
| user | A simplified prediction input for the computeNearestNeighbor function, this helps with efficiency of the prediction | Pandas Dataframe |

This third solution combines previous ideas but adds K nearest neighbors with k=5 as the final predictor for the value. This improves accuracy quite significantly which makes sense as K nearest neighbors is certainly a better option than just grabbing the most occurring value blindly.

Data description and Initial Processing

There are 37 million data entries for training data and 2.5 million for test data. Thus, while doing discovery, it is necessary to take a sample to avoid the long processing time and still represent the population of customers ordering on Expedia.

After investigating feature meaning I produced a correlation matrix to see which features interact with each other.

Chart

Description automatically generated

One of the takeaways we can gleam from this matrix is the relationship between orig\_destination\_distance and is\_package. It is a fairly significant correlation, and it indicates that packages are more often taken for people further away, which makes sense because those people will likely need plane travel to arrive at their location. There is also a positive correlation between srch\_adults\_cnt and srch\_rm\_cnt. This makes logical sense as the larger the adult group the more rooms are required.

An interesting follow-up to that observation is the distribution of srch\_adults\_cnt.

Chart, bar chart

Description automatically generated

The vast majority of customers consist of two adults, there are much fewer data points after 4 adults. This could be helpful if couples are more likely to book certain hotels versus single adults and groups of friends.

A follow-up to the is\_package and orig\_destination\_distance is a distribution of is\_package.

Chart, bar chart

Description automatically generated

We can see that the vast majority of customers are not ordering their hotel room as a part of a package. This is useful because perhaps people who are picking packages are taking longer vacations due to their commitment to fly to the location.

The dates in this dataset are also being stored as strings, which isn’t useful for analysis. Converting dates to new features for days, months and years allow us to see a distribution on when most people are taking vacations.

Chart, bar chart

Description automatically generated

Looking at this distribution we see that hotel rooms begin to pick up during the summer month and peak again in December. This makes sense due to people’s propensity to go on vacations during the summer and again during the holidays. This could be useful to us because month of vacation could influence the clusters of hotels that a potential customer would prefer.

A final look would be beneficial at hotel\_cluster’s distribution. If certain clusters are popular, they could be a strong candidate for predicting the end results of a customer’s choice.

Chart, bar chart, histogram

Description automatically generated

We can certainly see outliers with very high amounts of people attending certain hotel clusters. This uneven distribution will be important to keep in mind when developing models as we must factor in popular and unpopular clusters as we predict values.

I then also create a new feature called days\_spent that is the difference between the srch\_in and srch\_out features.

Before modeling we must deal with NaN values present in the dataset. There were 7 dates in srch\_in and srch\_out. I simply removed these rows since the amount of NaN values was quite small and unlikely to affect model performance.

There are 3,729 NaN orig\_destination\_distance variables, this is far too many to just drop the rows. Instead, I replace the NaNs with the average orig\_destination\_distance.

Then I standardized multiple variables, orgin\_destination\_distance, days\_spent, srch\_adults\_cnt, srch\_children\_cnt, srch\_rm\_cnt and cnt. This is necessary because the differing scales of these features would affect the models.

I also took categorical features that weren’t ordinal and performed one-hot-encoding using get\_dummies, this is done because leaving it in the original state could have models attempting to use the arbitrary values in an ordinal fashion. This was performed on hotel\_continent, hotel\_country, hotel\_market, user\_location\_country, user\_location\_region, user\_location\_city, site\_name, posa\_continent check\_in\_month, date\_time\_month and channel.

Modeling

Random Forest Classifier

The first model I selected was the sklearn random forest classifier. This was selected because of the nature of prediction needed. We are predicting the hotel\_cluster for each customer so we have a categorical target. There are a lot of columns after the one hot encoding so I figured that random forest would do a solid job of dealing with it.

I create the train and test data using sklearn’s train test split. The test split used was .3. The sample used is 10,000 entries.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Train | Test |
| RandomForestClassifier | 32.1% | 70% | 30% |

The top 10 feature importance’s for the random forest are below.

|  |  |
| --- | --- |
| Feature | Importance Value |
| Orig\_destination\_distance | .189 |
| Cnt | .077 |
| Srch\_destination\_id | .046 |
| Days\_spent | .038 |
| srch\_adults\_cnt | .024 |
| srch\_destination\_type\_id | .023 |
| Is\_booking | .016 |
| Channel\_9 | .015 |
| Srch\_children\_cnt | .012 |
| Is\_package | .01 |

Comparing this model result to previous predictions it does better than the most upvoted solution that was primarily based off of creating popularity but does worse than the two other models. The reason for this is because the contest actually had a flaw, orig\_destination and user\_location\_city when paired is a data leak and is able to predict the hotel cluster to an extremely high probability. All of the high scoring models used this to their advantage so the random\_forest\_classifier does a very good job accuracy wise without abusing that data leak.

The most important feature to a successful prediction for the random forest classifier is first and foremost orig\_destination\_distance. Its importance value is two and a half times greater than the second highest feature. Days\_spent also appears on the top features which shows that creating the feature had a positive effect on prediction and was certainly worth the effort. The channel 9 feature is also in the top ten and that was created when performing one-hot-encoding. This is a testament to the power of ensuring that non-ordinal categorical data is stored as dummies as opposed to just values in a single column. This result means that the marketing channel under the id 9 is important to predicting which cluster travelers end up at. In real world terms perhaps, this channel is aggressively pushing a certain cluster and it is working on travelers.

K Nearest Neighbor

The next model I investigated was a K nearest neighbor classifier from sklearn’s library. I did this because K nearest neighbors is commonly used for the categorical data if they are well defined and scales well with additional data. Since there are 37 million training entries it could see strong performance as more and more data are added to a sample. It also makes sense from the perspective of clustering; each hotel cluster is likely to bring in similar customers with similar traits. The K nearest neighbor classifier will find the most similar customers and make the prediction based on the majority of their choices.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Train | Test |
| K Nearest Neighbors | 23.6% | 70% | 30% |

The K nearest neighbor model doesn’t perform as well as the random forest classifier model. I suspect this could be due to only have 10k samples from the original data and only 7k of those are used in the training data for the model. If more data was used it would increase the number of similar travelers and would allow better predictions to be produced. It is also a simpler model than most others so it could not capture as much information as the random forest classifier does. Another important reason why performance was worse is because K Nearest Neighbor doesn’t use the categorical data as there is no way to find the distance between categorical features. This results in the model losing a lot of information as many features of the dataset are categorical.

Decision Trees

The third model investigated was a simple decision tree algorithm from sklearn’s library. I chose this model because I was curious about the difference in performance between random forest and a basic decision tree. I expected random forest to perform better but the results were extremely close.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Train | Test |
| Decision Trees | 31.7% | 70% | 30% |

The decision tree classifier does a comparative job to random forest only faltering by a few points. The random forest is essentially a group of multiple decision trees from random selected rows that then takes the average prediction as its outcome. The regular decision tree instead creates one tree based on all the data. It is not a surprise that random forest performs better but the decision tree classifier remains competitive and does a great job compared to K Nearest Neighbors.

Neural Network

I wanted to test at least one neural network even though the sample size was relatively low just to see how it would perform. I used the MLPClassifier from sklearn with the solver ‘adam’.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Train | Test |
| MLPClassifier | 10.6% | 70% | 30% |

The neural network performs quite poorly compared to the other models tested. This is likely because neural networks require more fine tuning than the other models. It would also scale better with more data used so it would be interesting to see how it would perform as the size increases.

Modeling Takeaways

One of the biggest takeaways is the power of the random forest classifier and decision trees. Considering the fact that there are 100 different hotel clusters these accuracy percentages are pretty impressive. Randomly selecting a hotel cluster would be a 1% chance but using these methods we manage to get to 32% and likely can get even higher with more ingenuity and more fined tuned models. This taught me that the preprocessing of data is extremely important when it comes to make good predictions. A posted solution using random forest with poor preprocessing had only a 11% accuracy score. Being able to increase that by nearly three-fold through standardization, feature creation and one hot encoding is impressive.