Advances in Data sciences Final Project Report

http://hoteladvisorsystem.azurewebsites.net/

Team 3: Amitha Murali, Divyansh Srivastava, Jyoti Sharma

Table of Contents

Hotel Recommendation System

1.1 Problem statement

Expedia has provided us logs of customer behavior. These include what customers searched for, how they interacted with search results (click/book), whether or not the search result was a travel package. The data is a random selection from Expedia and is not representative of the overall statistics.

We are interested in predicting which hotel group a user is going to book. Expedia has in-house algorithms to form hotel clusters, where similar hotels for a search (based on historical price, customer star ratings, geographical locations relative to city center, etc) are grouped together. These hotel clusters serve as good identifiers to which types of hotels people are going to book, while avoiding outliers such as new hotels that don't have historical data.

Our goal is to predict the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event.

Recommending hotels to users based on hotel properties and user behavior. Sentiment Analysis of the reviews given by the user.

The train and test datasets are split based on time: training data from 2013 and 2014, while test data are from 2015. Training data includes all the users in the logs, including both click events and booking events. We only use training dataset.

destinations.csv data consists of features extracted from hotel reviews text.

train.csv

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

user_location_region	The ID of the region the customer is located	int
user_location_city	The ID of the city the customer is located	int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	double
user_id	ID of user	int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	int
Channel	ID of a marketing channel	int
srch_ci	Checkin date	string
srch_co	Checkout date	string
srch_adults_cnt	The number of adults specified in the hotel room	int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	int
srch_rm_cnt	The number of hotel rooms specified in the search	int
srch_destination_id	ID of the destination where the hotel search was performed	int
srch_destination_type_id	Type of destination	int
hotel_continent	Hotel continent	int
hotel_country	Hotel country	int
hotel_market	Hotel market	int
is_booking	1 if a booking, 0 if a click	tinyint
Cnt	Numer of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

destinations.csv

Column name	Description	Data type
srch_destination_id	ID of the destination where the hotel search was performed	int

Column name	Description	Data type
d1-d149	latent description of search regions	double

"Hotel" refers to hotels, apartments, B&Bs, hostels and other properties appearing on Expedia's websites. Room types are not distinguished and the data can be assumed to apply to the least expensive room type.

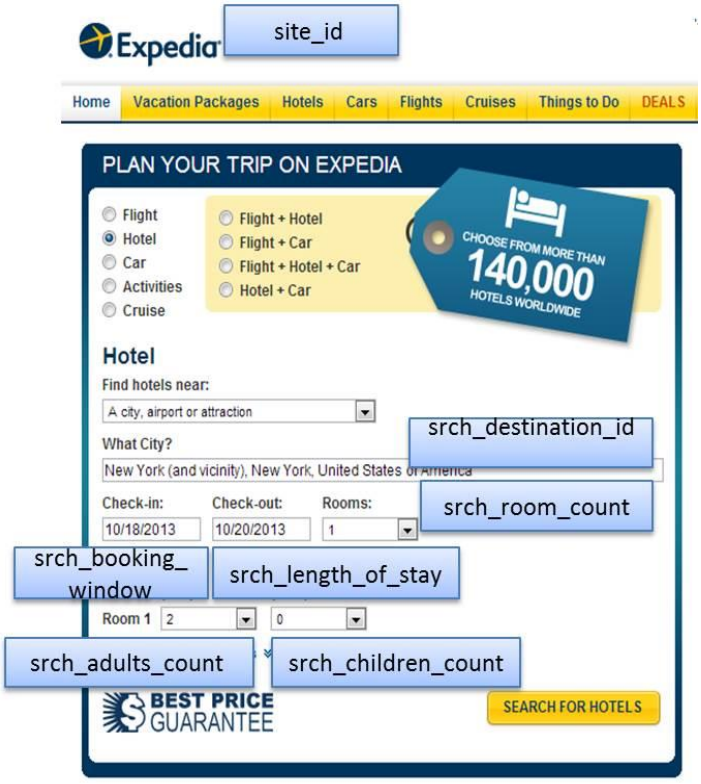
Most of the data are for searches that resulted in a purchase, but a small proportion are for searches not leading to a purchase.

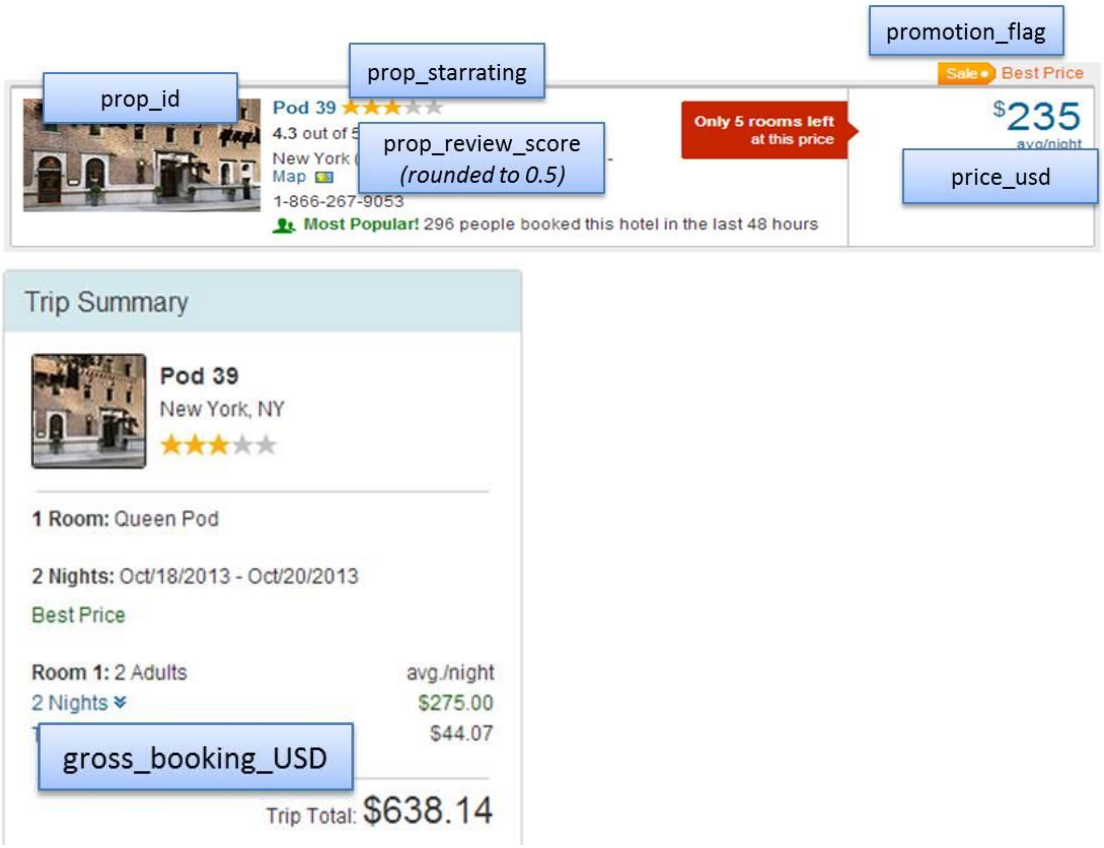
Column Name	Data Type	Description
srch_id	Integer	The ID of the search
date_time	Date/time	Date and time of the search
site_id	Integer	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp,)
visitor_location_country_id	Integer	The ID of the country the customer is located
visitor_hist_starrating	Float	The mean star rating of hotels the customer has previously purchased; null signifies there is no purchase history on the customer
visitor_hist_adr_usd	Float	The mean price per night (in US$) of the hotels the customer has previously purchased; null signifies there is no purchase history on the customer
prop_country_id	Integer	The ID of the country the hotel is located in
prop_id	Integer	The ID of the hotel
prop_starrating	Integer	The star rating of the hotel, from 1 to 5, in increments of 1. A 0 indicates the property has no stars, the star rating is not known or cannot be publicized.
prop_review_score	Float	The mean customer review score for the hotel on a scale out of 5, rounded to 0.5 increments. A 0 means there have been no reviews, null that the information is not available.
prop_brand_bool	Integer	+1 if the hotel is part of a major hotel chain; 0 if it is an independent hotel

prop_location_score1	Float	A (first) score outlining the desirability of a hotel's location
prop_location_score2	Float	A (second) score outlining the desirability of the hotel's location
prop_log_historical_price position	Float Integer	The logarithm of the mean price of the hotel over the last trading period. A 0 will occur if the hotel was not sold in that period. Hotel position on Expedia's search results page. This is only provided for the training data, but not the test data.
price_usd	Float	Displayed price of the hotel for the given search. Note that different countries have different conventions regarding displaying taxes and fees and the value may be per night or for the whole stay
promotion_flag	Integer	+1 if the hotel had a sale price promotion specifically displayed
gross_booking_usd srch_destination_id	Float Integer	Total value of the transaction. This can differ from the price_usd due to taxes, fees, conventions on multiple day bookings and purchase of a room type other than the one shown in the search ID of the destination where the hotel search was performed
srch_length_of_stay	Integer	Number of nights stay that was searched
srch_booking_window	Integer	Number of days in the future the hotel stay started from the search date
srch_adults_count	Integer	The number of adults specified in the hotel room
srch_children_count	Integer	The number of (extra occupancy) children specified in the hotel room
srch_room_count	Integer	Number of hotel rooms specified in the search
srch_saturday_night_bool	Boolean	+1 if the stay includes a Saturday night, starts from Thursday with a length of stay is less than or equal to 4 nights (i.e. weekend); otherwise 0
srch_query_affinity_score	Float	The log of the probability a hotel will be clicked on in Internet searches (hence the values are negative) A null signifies there are

		no data (i.e. hotel did not register in any searches)
orig_destination_distance	Float	Physical distance between the hotel and the customer at the time of search. A null means the distance could not be calculated.
random_bool	Boolean	+1 when the displayed sort was random, 0 when the normal sort order was displayed
comp1_rate	Integer	+1 if Expedia has a lower price than competitor 1 for the hotel; 0 if the same; -1 if Expedia's price is higher than competitor 1; null signifies there is no competitive data
comp1_inv	Integer	+1 if competitor 1 does not have availability in the hotel; 0 if both Expedia and competitor 1 have availability; null signifies there is no competitive data
comp1_rate_percent_diff	Float	The absolute percentage difference (if one exists) between Expedia and competitor 1's price (Expedia's price the denominator); null signifies there is no competitive data
comp2_rate		
comp2_inv		(same, for competitor 2 through 8)
comp2_rate_percent_diff		
comp3_rate		
comp3_inv		
comp3_rate_percent_diff		
comp4_rate		
comp4_inv		
comp4_rate_percent_diff		
comp5_rate		
comp5_inv		
comp5_rate_percent_diff		
comp6_rate		
comp6_inv		
comp6_rate_percent_diff		
comp7_rate		
comp7_inv		
Comp7_rate_percent_diff		

Comp8_rate		
Comp8_inv		
Comp8_rate_percent_diff		





1.2 Data Visualization

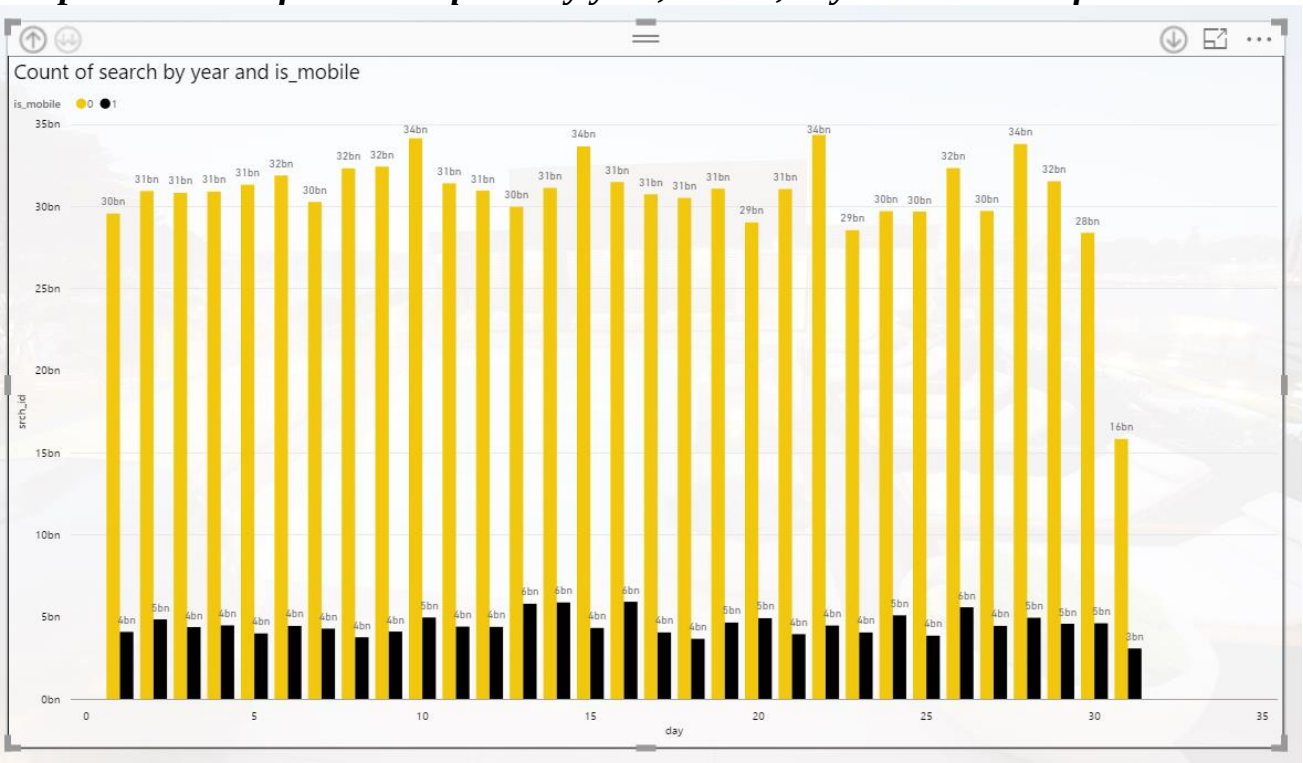
Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software.

1.2.1 Power BI Analysis:

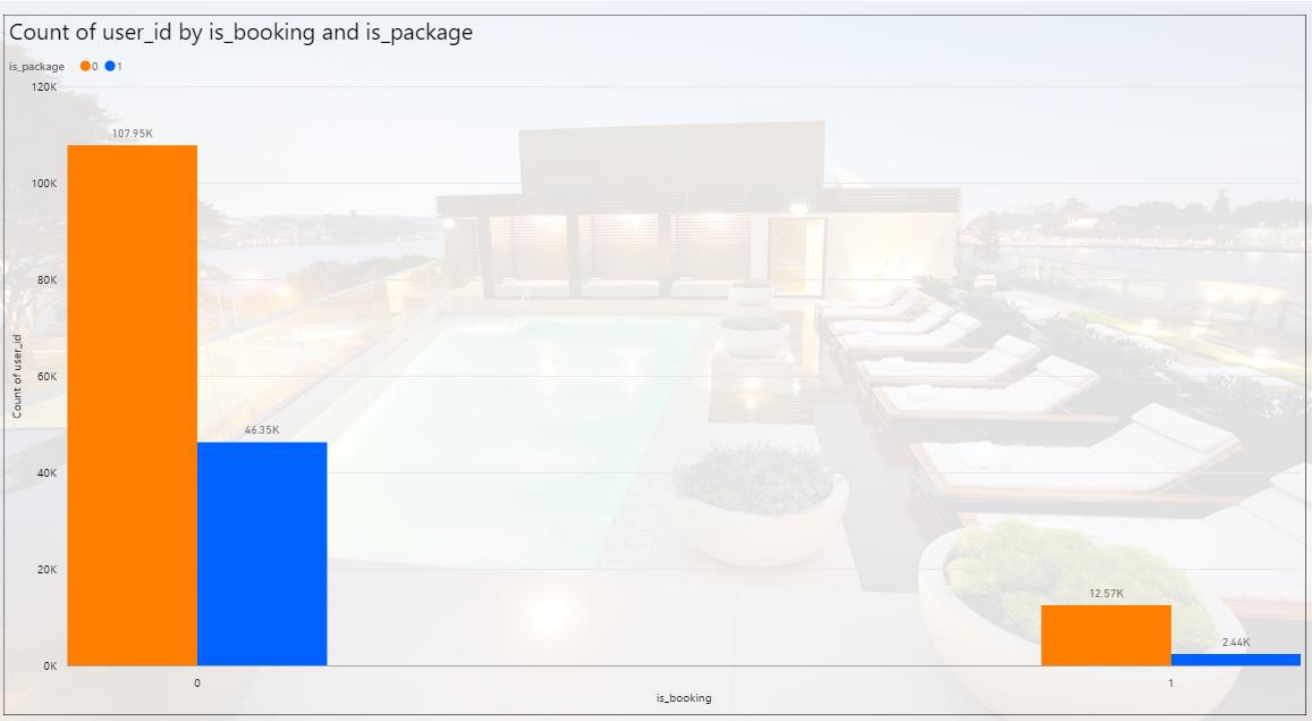
We have used the cleansed data to visualize and draw patterns about the hotel cluster data.

Graph1 – Count of hotel id by hotel cluster and hotel country

Graph2 – Count of search requests by year, month, day and is mobile factor.



Graph3 – Count of user id by booking and package.



Graph4 – Average of gross bookings by hotel cluster.

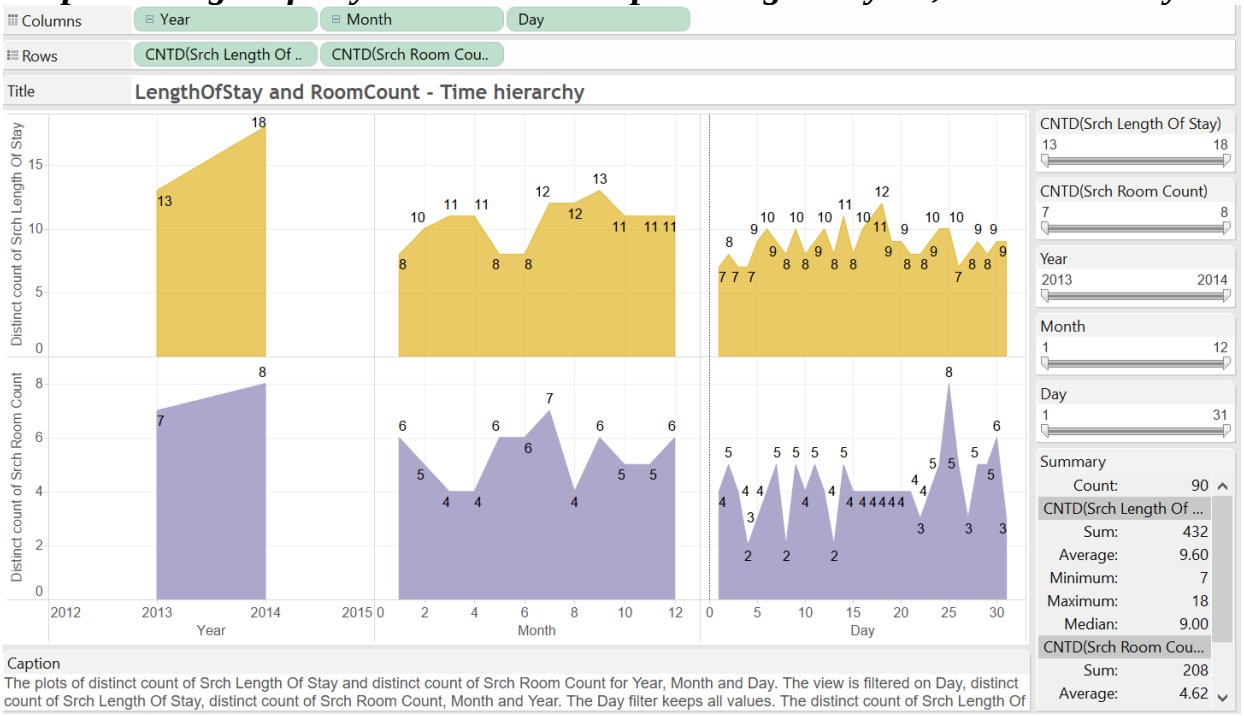
Graph5 – Count of hotel id by hotel star rating and hotel brand bool

Graph6 – Count of user id by search length of stay and search room count

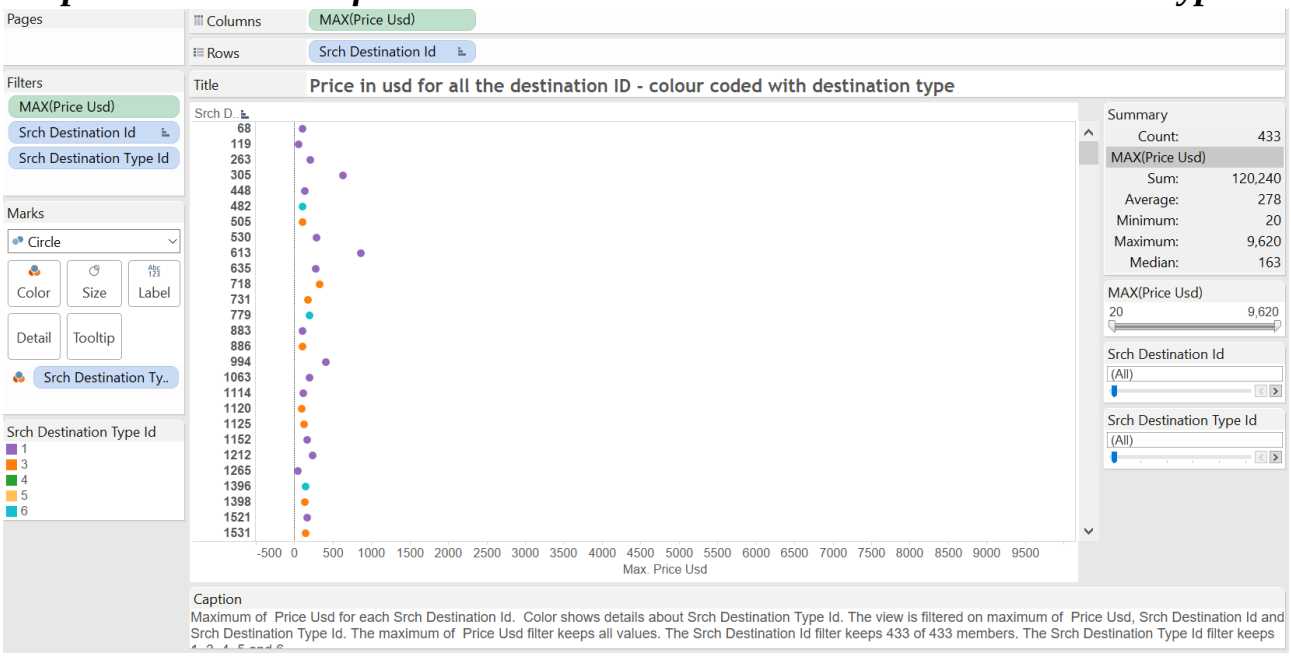
1.2.1 Tableau Analysis:

We have used the cleansed data to visualize and draw patterns about the hotel cluster data.

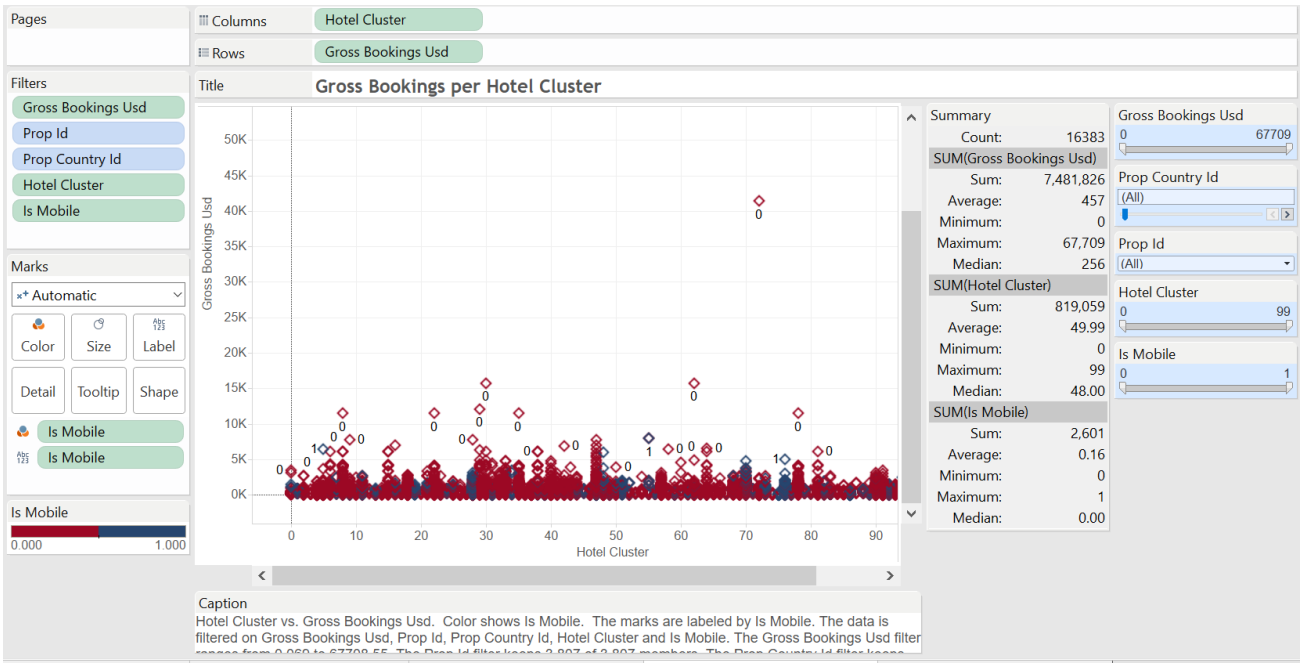
Graph1 –Length of stay and room count plotted against year, month and day



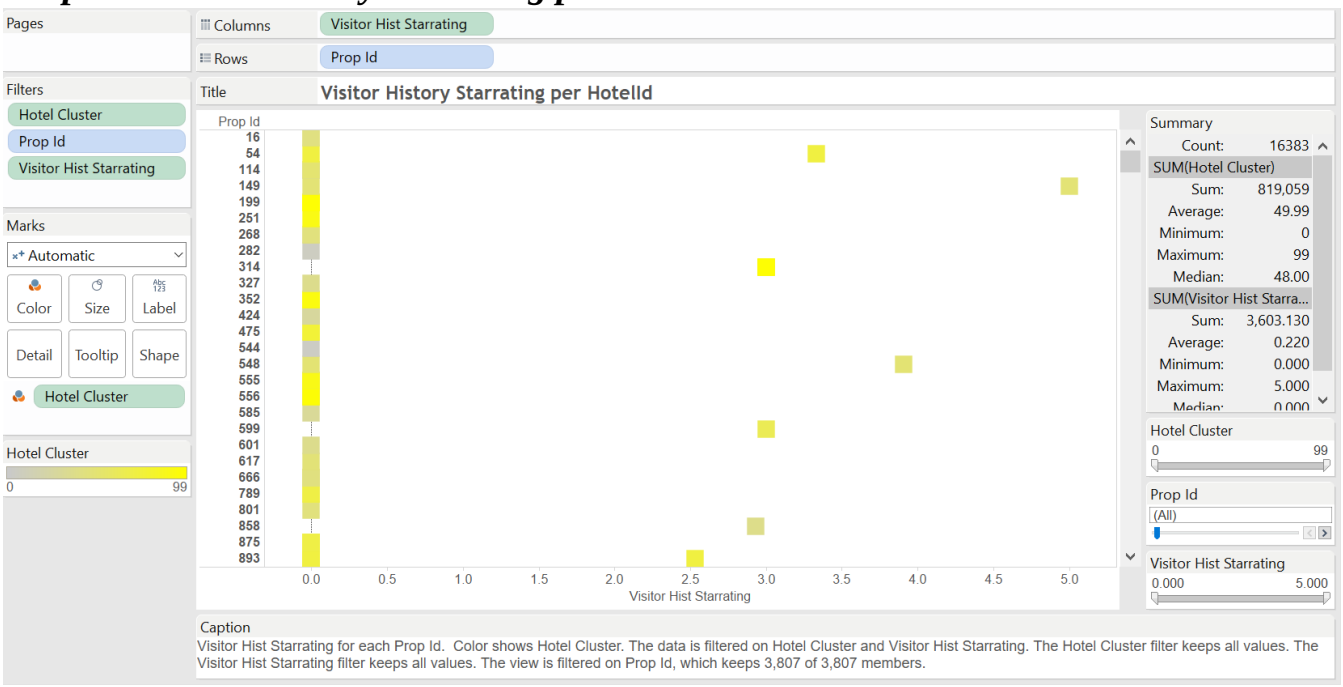
Graph2 – Price in usd for all destination Id – colour coded with destination type



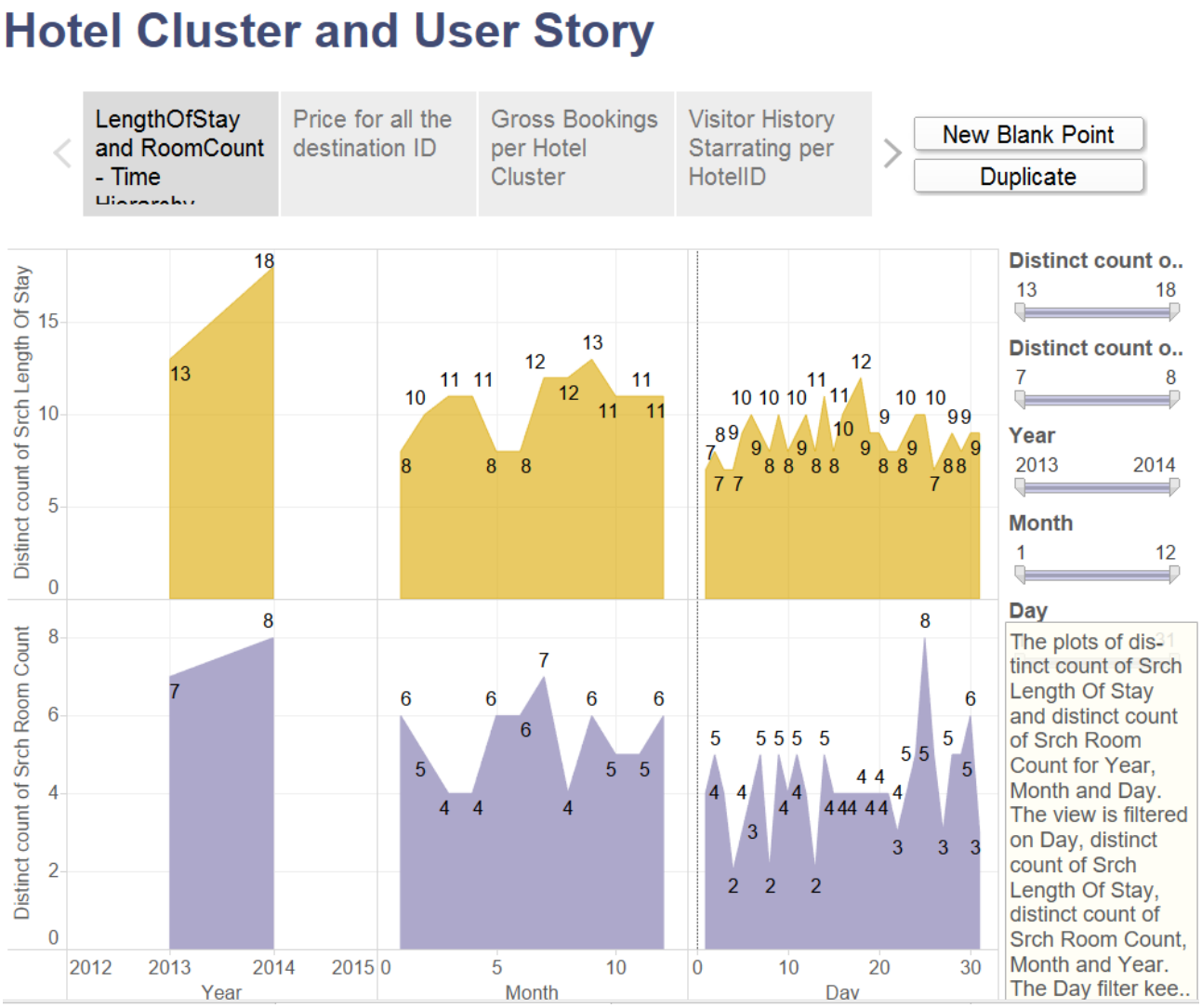
Graph3–Gross bookings per hotel cluster



Graph4 – Visitor history star rating per hotel id

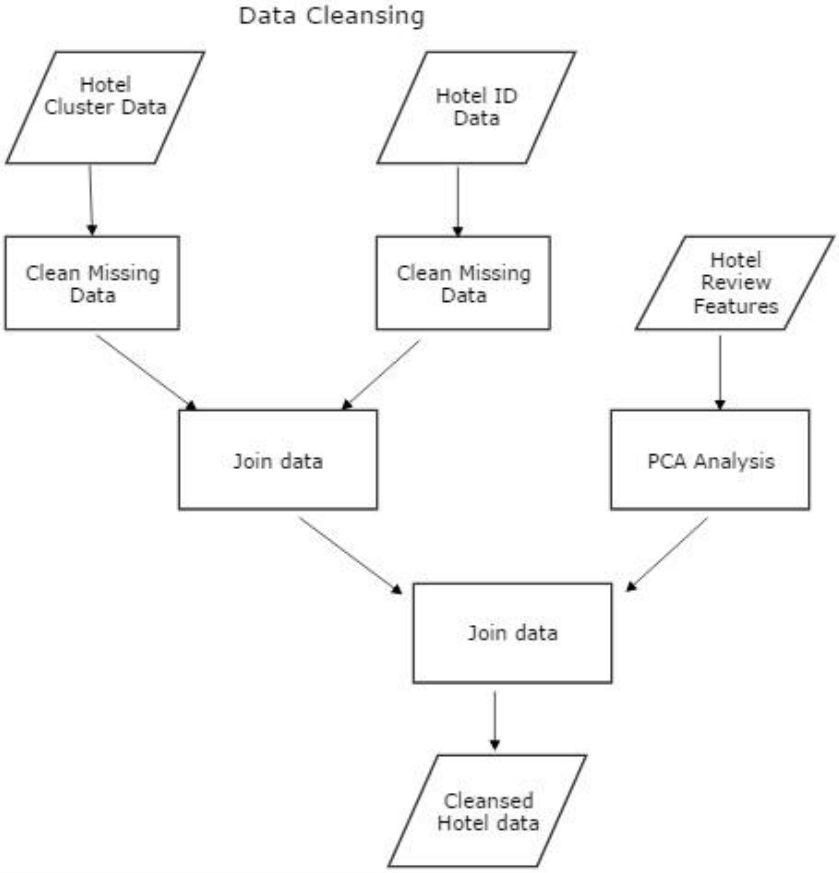


Graph5 – Tableau story

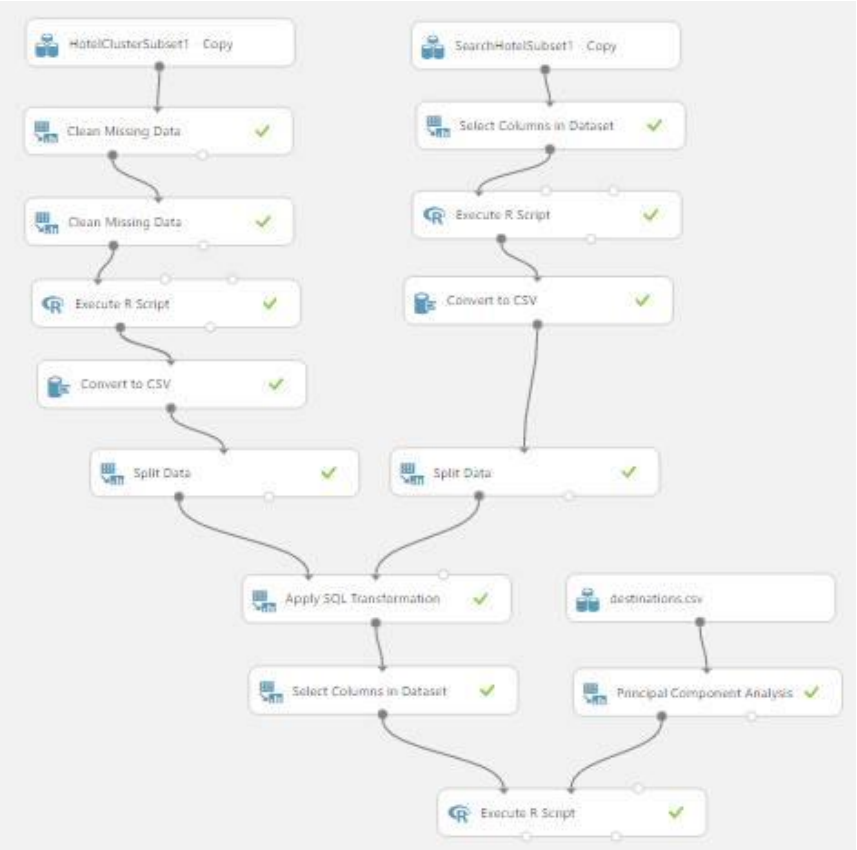


1.3 Data Pre-Processing

Hotel cluster and hotel Id are present in separate files which are merged together based on destination Id.



Azure Data cleansing model:



1.3.1 Missing Data Treatment

- In HotelCluster dataset approximately 0.12% data is missing for checkin and checkout data. So remove the rows with missing data.
- Column Orgi_destination_distance has 35% data as missing rows. One of the reasons it could be because the distance could not be calculated which can be because of presence of oceans in the origination and destination. Hence we decided to replace the missing values with the largest possible value present in the dataset.
- In visitor_hist_starrating and visitor_hist_adr_usd we have replaced all the NULL with '0' and ignored it later in our models.
- In search Hotel Dataset, column gross_booking_usd has rows with NULL value. To treat it calculate the formula for gross booking, which we found out as:

  (Price_usd * No. of rooms * length of stay) + 15% tax

```
#replace orig_destination_distance null values with largest distance present in the column
maximumDistance <- max(searchHotelData$orig_destination_distance)
searchHotelData$orig_destination_distance <- ifelse((searchHotelData$orig_destination_distance=='NULL'),maximumDistance,searchHotelDat

searchHotelData$gross_bookings_usd <- ifelse((searchHotelData$gross_bookings_usd=='NULL'),(
    ((searchHotelData$price_usd * searchHotelData$srch_room_count) + (searchHotelData$price_usd * 0.15)
        ) * searchHotelData$srch_length_of_stay),searchHotelData$gross_bookings_usd)
```

- Extract date, year, month, day and hour values for each date
- Create a column for Day of the week by assuming sun to Sat is represented in 0-6 format
- Create a column for IsWeekday by assuming 1-yes, 0-No
- Create a column for Booking_window by calculating the time difference between checkin date and booking date
- Create a column for stay_length by calculating the time difference between checkout date and checkin date

R Script

```
HotelClusterData$year  <- as.numeric(format(as.Date(HotelClusterData$date_time,format ='%m/%d/%Y'), format= "%Y"))
HotelClusterData$month  <- as.numeric(format(as.Date(HotelClusterData$date_time,format ='%m/%d/%Y'), format= "%m"))
HotelClusterData$day  <- as.numeric(format(as.Date(HotelClusterData$date_time,format ='%m/%d/%Y'), format= "%d"))
HotelClusterData$hour <- format(as.POSIXct(HotelClusterData$date_time),'%H')
#monthlyNetworkData$hour  <- as.numeric(format(as.Date(monthlyNetworkData$time,format ='%H:%M:%S'), format= "%H"))

#Get Day of the week--- sun to Sat in 0-6 format
require(lubridate)
HotelClusterData$DayOfWeek = wday(as.Date(HotelClusterData$date_time,format ='%Y-%m-%d'))-1

#IsWeekday ----- 1-yes, 0-No
require(timeDate)
HotelClusterData$Weekday <- ifelse(isWeekday(as.Date(HotelClusterData$date_time,format ='%Y-%m-%d'), wday = 1:5), 1, 0)

#HotelClusterData$Booking_Window <- difftime(HotelClusterData$srch_ci, HotelClusterData$date_time, units = "days") # days

HotelClusterData$Booking_Window <- as.Date(as.character(HotelClusterData$srch_ci), format="%Y-%m-%d")-
                  as.Date(as.character(HotelClusterData$date_time), format="%Y-%m-%d")
HotelClusterData$Stay_Length <- as.Date(as.character(HotelClusterData$srch_co), format="%Y-%m-%d")-
                  as.Date(as.character(HotelClusterData$srch_ci), format="%Y-%m-%d")
```

- Join the two dataset on the basis of destination ID.

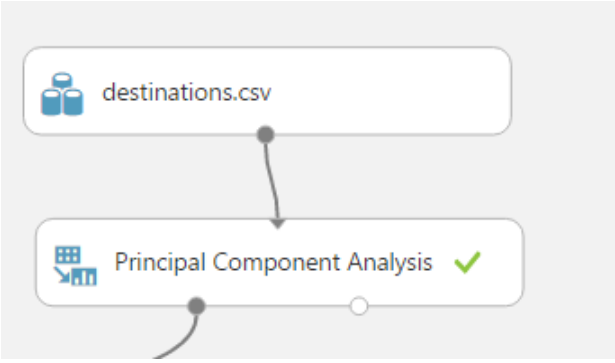
```
select * from t1 join t2 on t1.srch_destination_id = t2.srch_destination_id;
```

1.3.2 Feature Selection

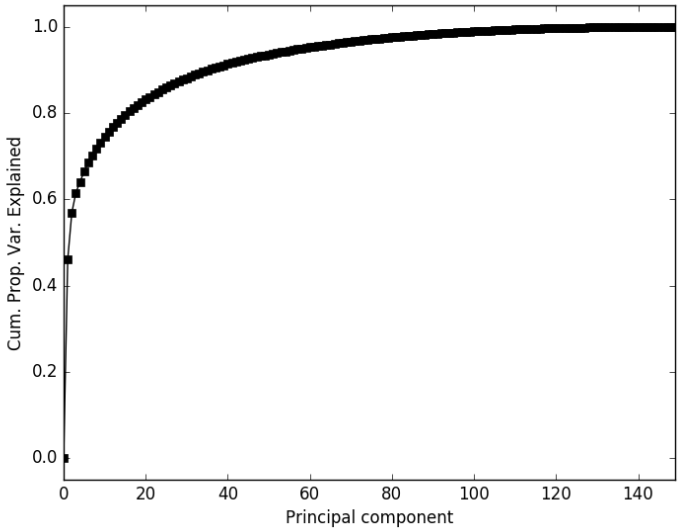
- Each of the features of the destination data set is actually a column from “d1” to “d149,” which means we don’t really know what the data within those columns mean. Furthermore, adding 149 columns of data to our training set for prediction will not be

particularly useful and will, more than likely, result in overfitting if we try to use them raw.

- One way to deal with that is to identify a small number of components–say, 5 of them–that explain a large amount of the variance among the 149 features.
- Apply **Principal component analysis (PCA)** on the destination file which contains the data on hotel rating features.



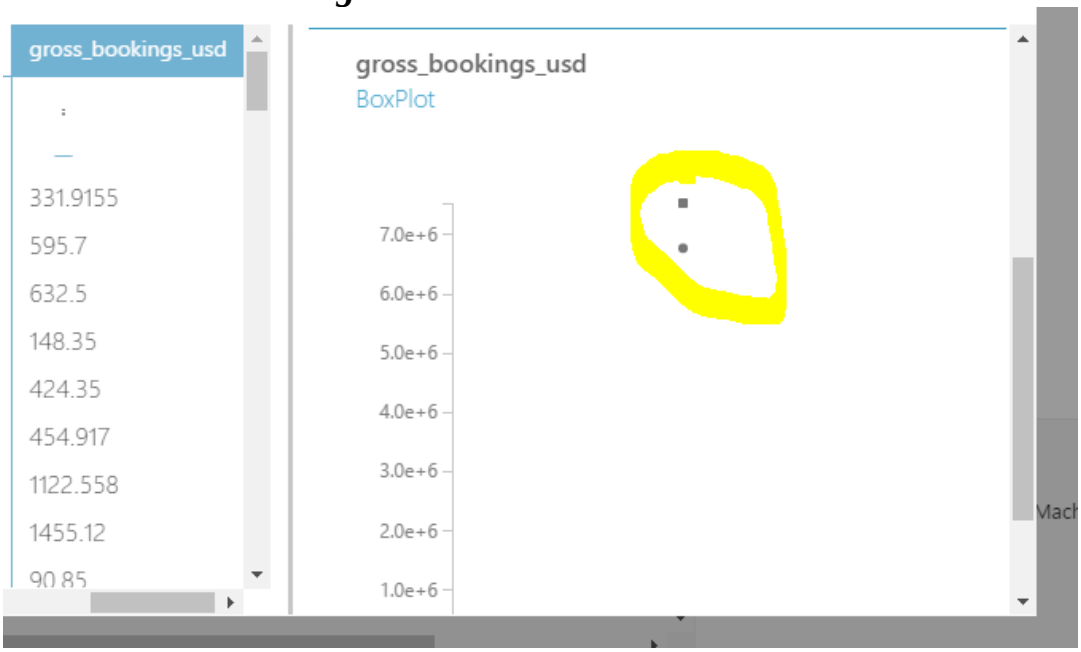
- Join the PCA analysed dataset to the hotel dataset.



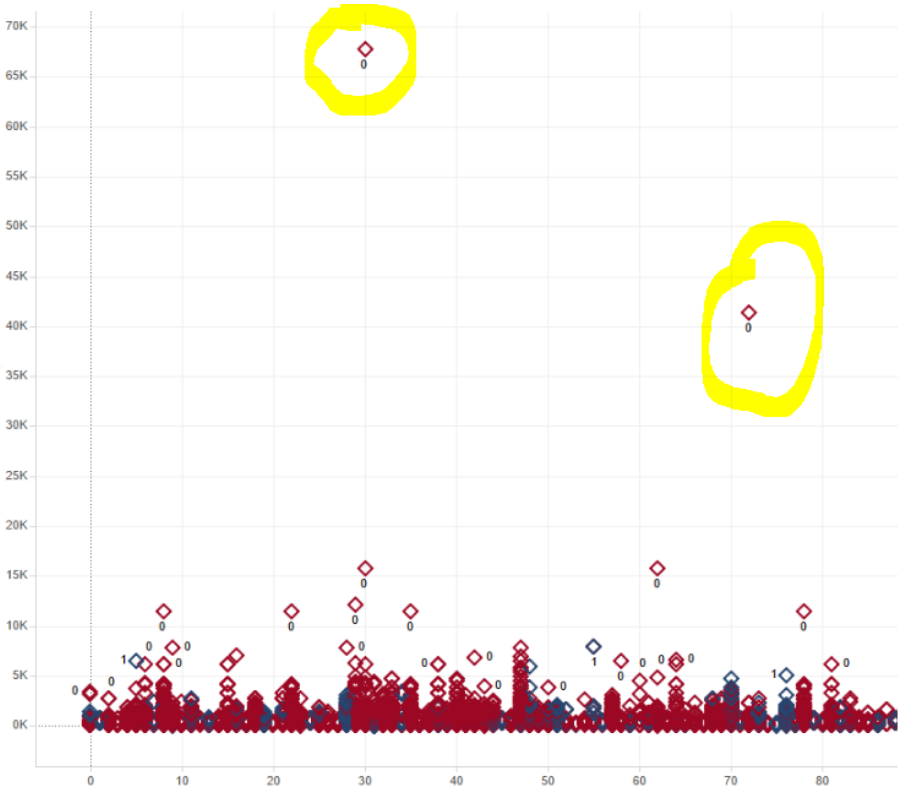
1.3.3 Outliers detection and removal

- An **outlier** is an observation that appears to deviate markedly from other observations in the sample. Identification of potential **outliers** is important.
- Perform **outlier detection** on the combined dataset using Azure boxplot and tableu.

Outlier detection using Azure ML visualization:



Outlier detection using Tableau visualization:



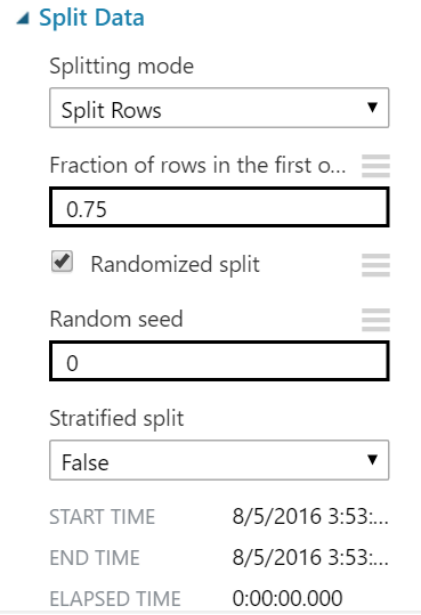
- Remove the outliers using execute R Script in Azure.

R Script

```
outlier.dataset <- c();
for(i in 1:nrow(hotelData)) {
  if(((hotelData$gross_bookings_usd[i]) < (mean(hotelData$gross_bookings_usd) - (1.5)*sd(hotelData$gross_bookings_usd)) |
      (hotelData$gross_bookings_usd[i]) > (mean(hotelData$gross_bookings_usd) + (1.5)*sd(hotelData$gross_bookings_usd))) &
     ((hotelData$srch_adults_cnt[i]) < (mean(hotelData$srch_adults_cnt) - (1.5)*sd(hotelData$srch_adults_cnt)) |
      (hotelData$srch_adults_cnt[i]) > (mean(hotelData$srch_adults_cnt) + (1.5)*sd(hotelData$srch_adults_cnt))) ) {
      #&
     #((hotelData$srch_children_cnt[i]) < (mean(hotelData$srch_children_cnt) - (1.5)*sd(hotelData$srch_children_cnt)) |
     # (hotelData$srch_children_cnt[i]) > (mean(hotelData$srch_children_cnt) + (1.5)*sd(hotelData$srch_children_cnt))) &
     #((hotelData$srch_rm_cnt[i]) < (mean(hotelData$srch_rm_cnt) - (1.5)*sd(hotelData$srch_rm_cnt)) |
      #(hotelData$srch_rm_cnt[i]) > (mean(hotelData$srch_rm_cnt) + (1.5)*sd(hotelData$srch_rm_cnt))) &
     #((hotelData$prop_country_id[i]) < (mean(hotelData$prop_country_id) - (1.5)*sd(hotelData$prop_country_id)) |
      #(hotelData$prop_country_id[i]) > (mean(hotelData$prop_country_id) + (1.5)*sd(hotelData$prop_country_id)))) {
    count = count + 1;
    outlier.dataset <- c(outlier.dataset, i)
  }
}
#outliers detcted - 11
#removing the outlier from cleaned dataset
hotelDataNew <- data.frame(hotelData[-outlier.dataset,])
```

Azure Modules:

- **Split Data -** To split a dataset into two equal parts, just add the Split Data module after the dataset without no other changes. By default, the module splits the dataset in two equal parts. For data with an odd number of rows, the second output gets the remainder.



- **Train Model** - Training a classification or regression model is a kind of *supervised machine learning*. That means you must provide a dataset that contains historical data from which to learn patterns. The data should contain both the outcome you are trying to predict,

and related factors (variables). The machine learning model uses the data to extract statistical patterns and build a model.

When you configure Train Model**,** you must also connect an already configured model, such as a regression algorithm, decision tree model, or other machine learning module.

- **Score Model -** Score Model is used to generate predictions using a trained classification or regression model. The predicted value can be in many different formats, depending on the model and your input data: If you are using a classification model to create the scores, Score Model outputs a predicted value for the class, as well as the probability of the predicted value. For regression models, Score Model generates just the predicted numeric value.

- **Evaluate Model** - Evaluate Model is used to measure the accuracy of a trained classification model or regression model. You provide a dataset containing scores generated from a trained model, and the Evaluate Model module computes a set of industry-standard evaluation metrics. The metrics returned by Evaluate Model depend on the type of model that you are evaluating

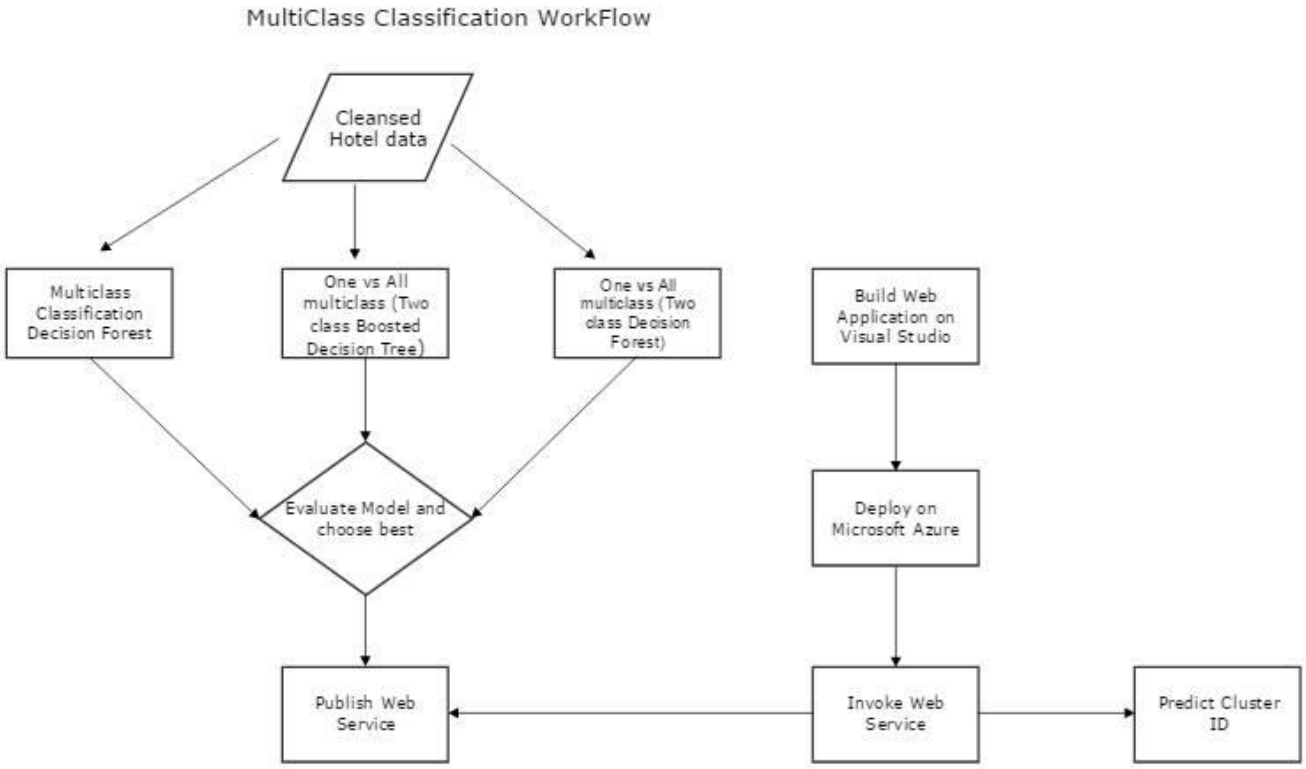
1.4 Classification Models

We build classification model to predict the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event.

1.4.1Overall Design

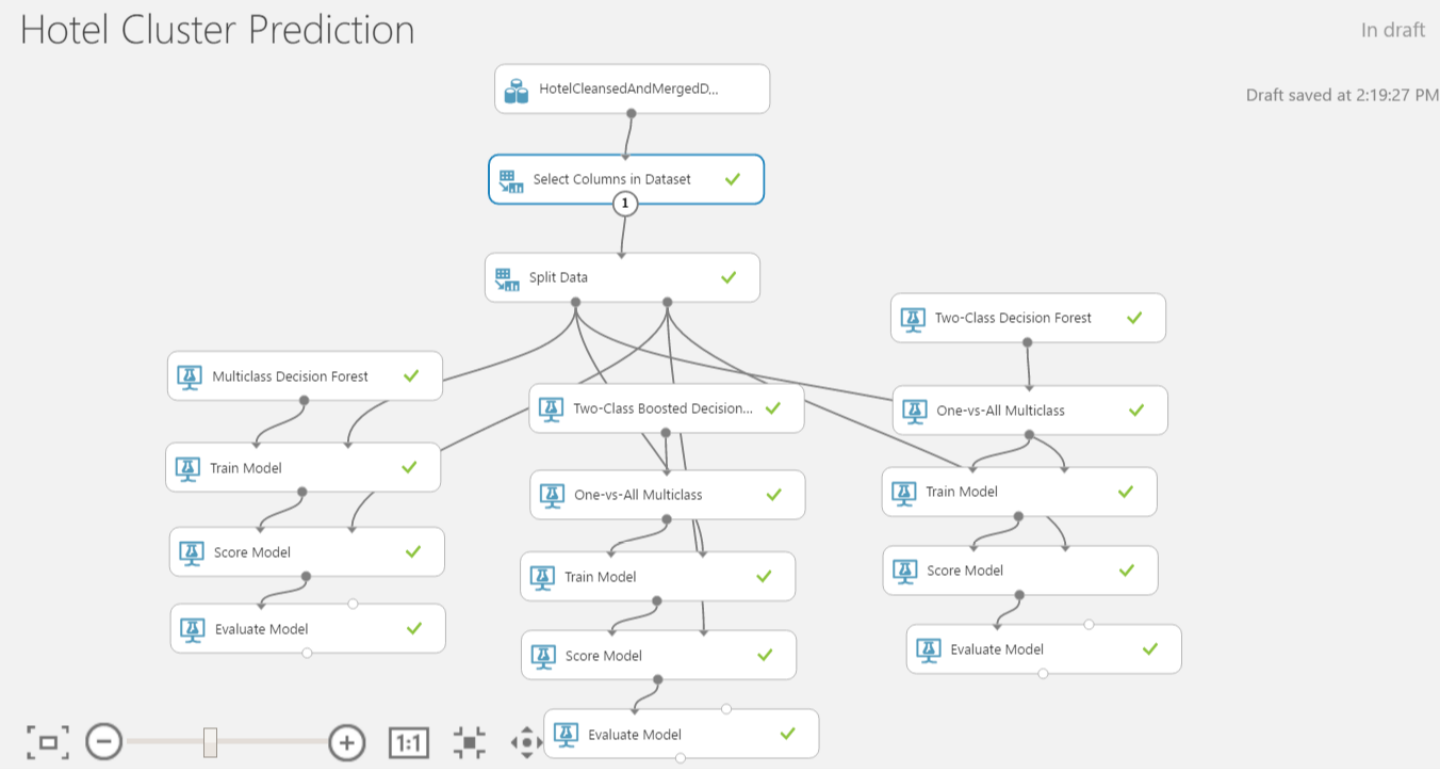
- Read the hotel recommendation cleansed and merged data.
- Select only the columns that have high coefficient value and exclude the rest.
- Split the data into train and test by 75% and 25% respectively.
- Implement Multi class Decision Forest, Two class Decision Forest and One-vs- All Multiclass, Two class Boosted Decision Tree and One-vs- All Multiclass.
- Parameters used in each model is elaborated in the further module below.
- Compare the models through the overall accuracy, average accuracy, macro-averaged precision, macro-averaged recall and choose the best model.
- Publish the best classification model created for this data as a web service.
- Build a web application using visual studio and deploy it on Microsoft Azure.
- This web application is used as an interface to invoke the web service expose through Azure.

- Predictions are performed through this web application to the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event.



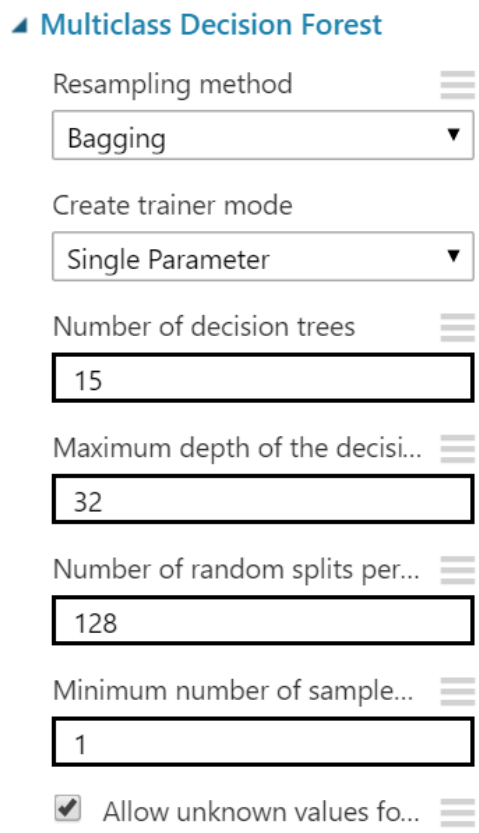
1.4.2 Azure models

Below is the hotel recommendation system classification model created for predicting the hotel cluster based on user event.



1.4.3 Multiclass Decision Forest

A machine learning model based on the decision forest algorithm. The decision forest algorithm is an ensemble learning method for classification. The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the "probabilities" for each label. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble. Decision trees in general are non-parametric models, meaning they support data with varied distributions. In each tree, a sequence of simple tests is run for each class, increasing the levels of a tree structure until a leaf node (decision) is reached. Decision trees have many advantages: They can represent non-linear decision boundaries. They are efficient in computation and memory usage during training and prediction. They perform integrated feature selection and classification. They are resilient in the presence of noisy features. The decision forest classifier in Azure Machine Learning Studio consists of an ensemble of decision trees. Generally, ensemble models provide better coverage and accuracy than single decision trees.



Confusion Matrix and other performance metrics :

The following statistics are shown for our model:

Precision: Given all the predicted labels (for a given class X), how many instances were correctly predicted?
Recall: For all instances that should have a label X, how many of these were correctly captured?

The generalization to multi-class problems is to sum over rows / columns of the confusion matrix. Given that the matrix is oriented as above, i.e., that a given row of the matrix corresponds to specific value for the "truth", we have:

Precision $i = M_{ii} / \sum_j M_{ji}$

Recall $i = M_{ii} / \sum_j M_{ij}$

That is, precision is the fraction of events where we *correctly* declared ii out of all instances where the algorithm declared ii. Conversely, recall is the fraction of events where we correctly declared i out of all of the cases where the true of state of the world is i

Hotel Cluster Prediction › Evaluate Model › Evaluation results

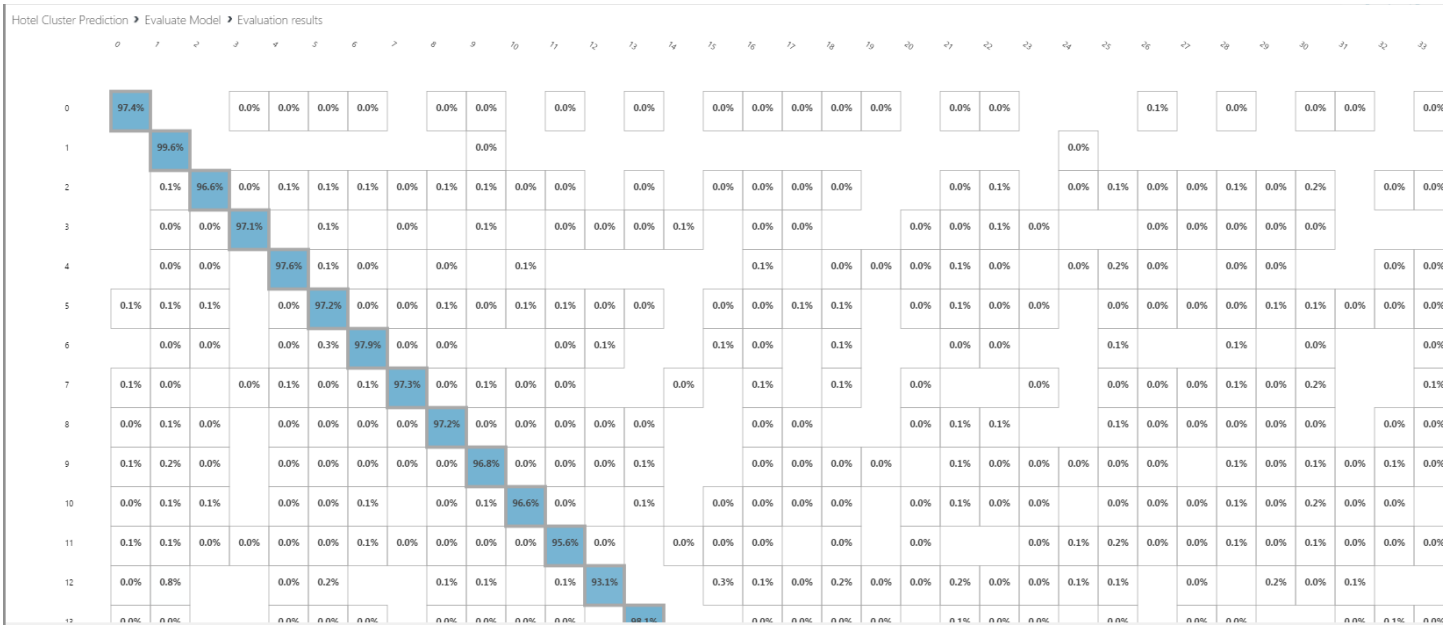
Metrics

Overall accuracy	0.970708
Average accuracy	0.999414
Micro-averaged precision	0.970708
Macro-averaged precision	0.966281
Micro-averaged recall	0.970708
Macro-averaged recall	0.966178

The Need for a Confusion Matrix

Apart from helping with computing precision and recall, it is always important to look at the confusion matrix to analyze your results as it also gives you very strong clues as to where your classifier is going wrong.

Confusion Matrix



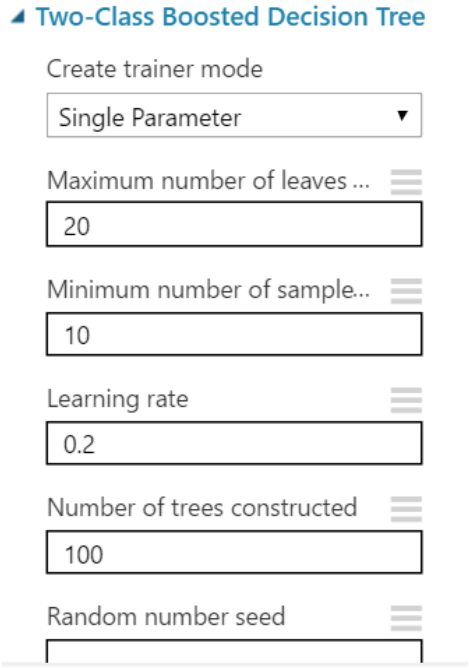
1.4.4 One – vs – All Multiclass

Azure ML Studio also provides a module called One-vs-All Multiclass which can use any binary classifier as an input to solve a multi-class classification problem, based on this one-vs-

all method. Therefore, as the second model for comparison, we used a binary classification module, **Two-Class Boosted Decision Tree**, and connected it to the **One-vs-All Multiclass** module. Third model for comparison, we used a binary classification module, **Two-Class Decision Forest**, and connected it to the **One-vs-All Multiclass** module.

1.4.4.1 Two class Boosted Decision Tree

A boosted decision tree is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth. Predictions are based on the entire ensemble of trees together that makes the prediction. Generally, when properly configured, boosted decision trees are the easiest methods with which to get top performance on a wide variety of machine learning tasks. However, they are also one of the more memory-intensive learners, and the current implementation holds everything in memory; therefore, a boosted decision tree model might not be able to process the very large datasets that some linear learners can handle.



Confusion Matrix and other performance metrics :

The following statistics are shown for our model:

Precision: Given all the predicted labels (for a given class X), how many instances were correctly predicted?
Recall: For all instances that should have a label X, how many of these were correctly captured?

The generalization to multi-class problems is to sum over rows / columns of the confusion matrix. Given that the matrix is oriented as above, i.e., that a given row of the matrix corresponds to specific value for the "truth", we have:

Precision i=Mii / ∑jMji

Recall i=Mii / ∑jMij

That is, precision is the fraction of events where we *correctly* declared ii out of all instances where the algorithm declared ii. Conversely, recall is the fraction of events where we correctly declared iiout of all of the cases where the true of state of the world is i

Hotel Cluster Prediction › Evaluate Model › Evaluation results

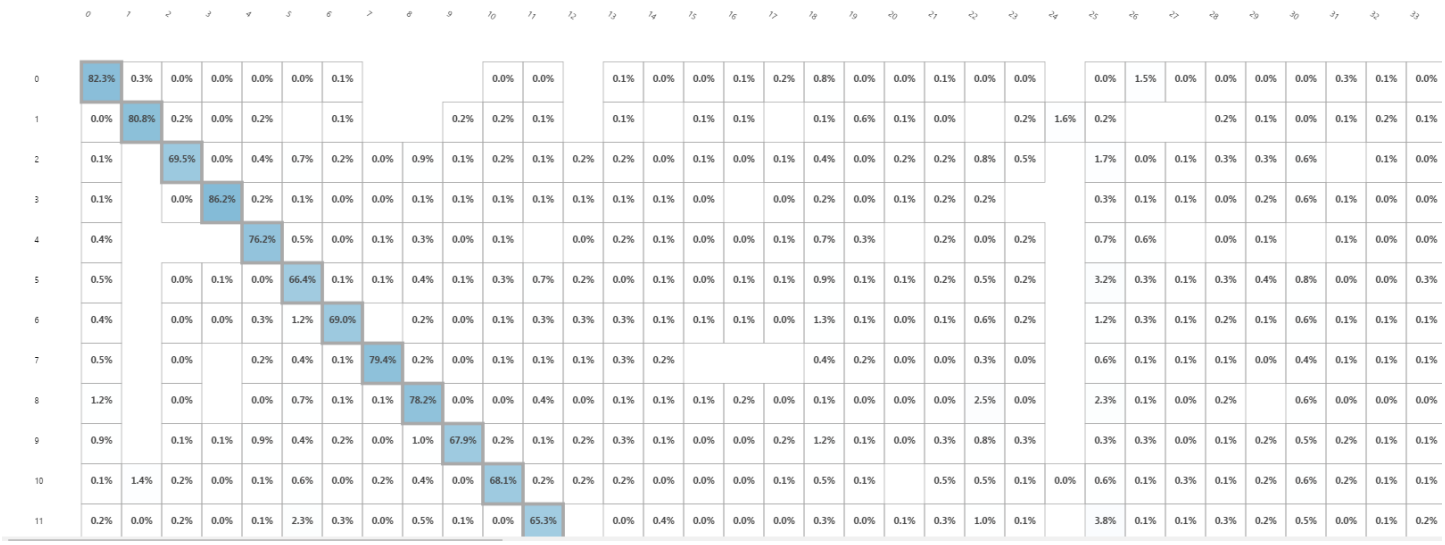
Metrics

Overall accuracy	0.7463
Average accuracy	0.994926
Micro-averaged precision	0.7463
Macro-averaged precision	0.787456
Micro-averaged recall	0.7463
Macro-averaged recall	0.731311

The Need for a Confusion Matrix

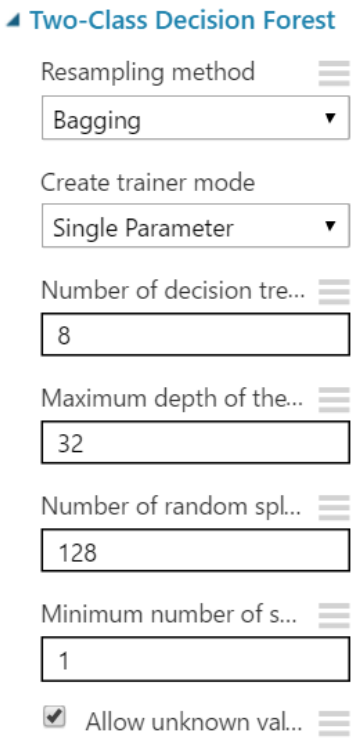
Apart from helping with computing precision and recall, it is always important to look at the confusion matrix to analyze your results as it also gives you very strong clues as to where your classifier is going wrong.

Confusion Matrix



1.4.4.2 Two class Decision Forest

The decision forest algorithm is an ensemble learning method for classification. The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the "probabilities" for each label. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble.



Confusion Matrix and other performance metrics :

The following statistics are shown for our model:

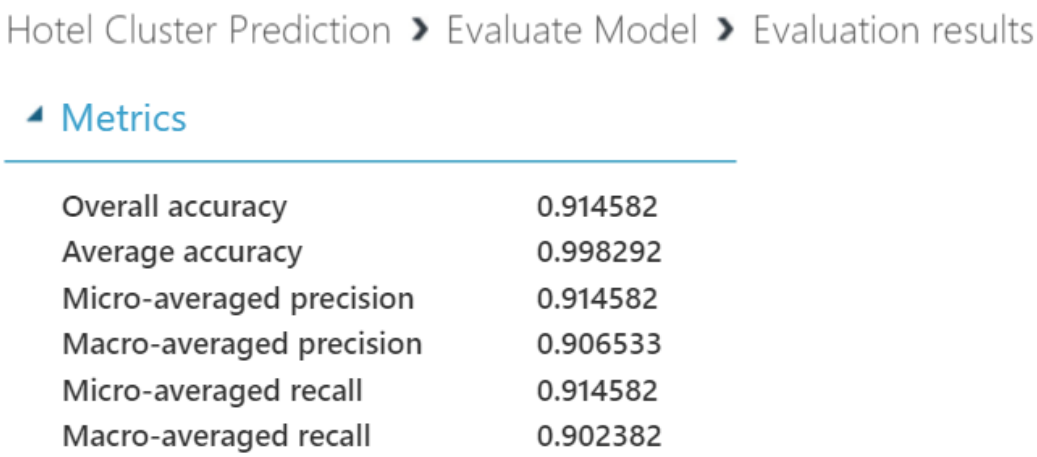
Precision: Given all the predicted labels (for a given class X), how many instances were correctly predicted?
Recall: For all instances that should have a label X, how many of these were correctly captured?

The generalization to multi-class problems is to sum over rows / columns of the confusion matrix. Given that the matrix is oriented as above, i.e., that a given row of the matrix corresponds to specific value for the "truth", we have:

Precision i=Mii / ∑jMji

Recall i=Mii / ∑jMij

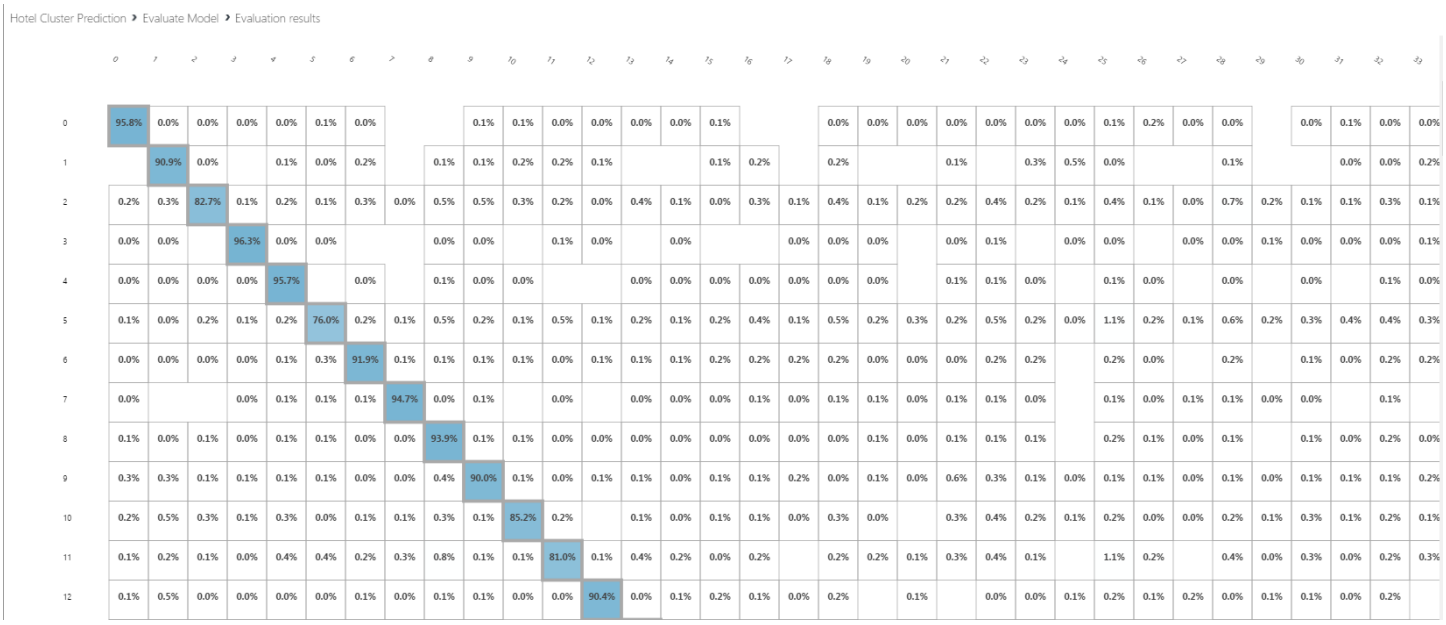
That is, precision is the fraction of events where we *correctly* declared ii out of all instances where the algorithm declared ii. Conversely, recall is the fraction of events where we correctly declared i out of all of the cases where the true of state of the world is i



The Need for a Confusion Matrix

Apart from helping with computing precision and recall, it is always important to look at the confusion matrix to analyze your results as it also gives you very strong clues as to where your classifier is going wrong.

Confusion Matrix



1.4.5 Classification model comparison

	ROC curve
Multiclass Decision Forest	Hotel Cluster Prediction › Evaluate Model › Evaluation results Metrics Overall accuracy 0.970708 Average accuracy 0.999414 Micro-averaged precision 0.970708 Macro-averaged precision 0.966281 Micro-averaged recall 0.970708 Macro-averaged recall 0.966178
One -vs-All Multiclass (Two class Boosted Decision Tree)	Hotel Cluster Prediction › Evaluate Model › Evaluation results Metrics Overall accuracy 0.7463 Average accuracy 0.994926 Micro-averaged precision 0.7463 Macro-averaged precision 0.787456 Micro-averaged recall 0.7463 Macro-averaged recall 0.731311

One -vs-All Multiclass (Two class Decision Forest)	Hotel Cluster Prediction › Evaluate Model › Evaluation results Metrics Overall accuracy 0.914582 Average accuracy 0.998292 Micro-averaged precision 0.914582 Macro-averaged precision 0.906533 Micro-averaged recall 0.914582 Macro-averaged recall 0.902382

Conclusion:

In evaluating multi-class classification problems, we often think that the only way to evaluate performance is by computing the accuracy which is the proportion or percentage of correctly predicted labels over all predictions.

However, we can always compute precision and recall for each class label and analyze the individual performance on class labels or average the values to get the overall precision and recall.

Accuracy alone is sometimes quite misleading as you may have a model with relatively 'high' accuracy with the model predicting the ***'not so important'*** class labels fairly accurately (e.g. "unknown bucket") but the model may be making all sorts of mistakes on the classes that are actually critical to the application.

Precision: Given all the predicted labels (for a given class X), how many instances were correctly predicted?
Recall: For all instances that should have a label X, how many of these were correctly captured?

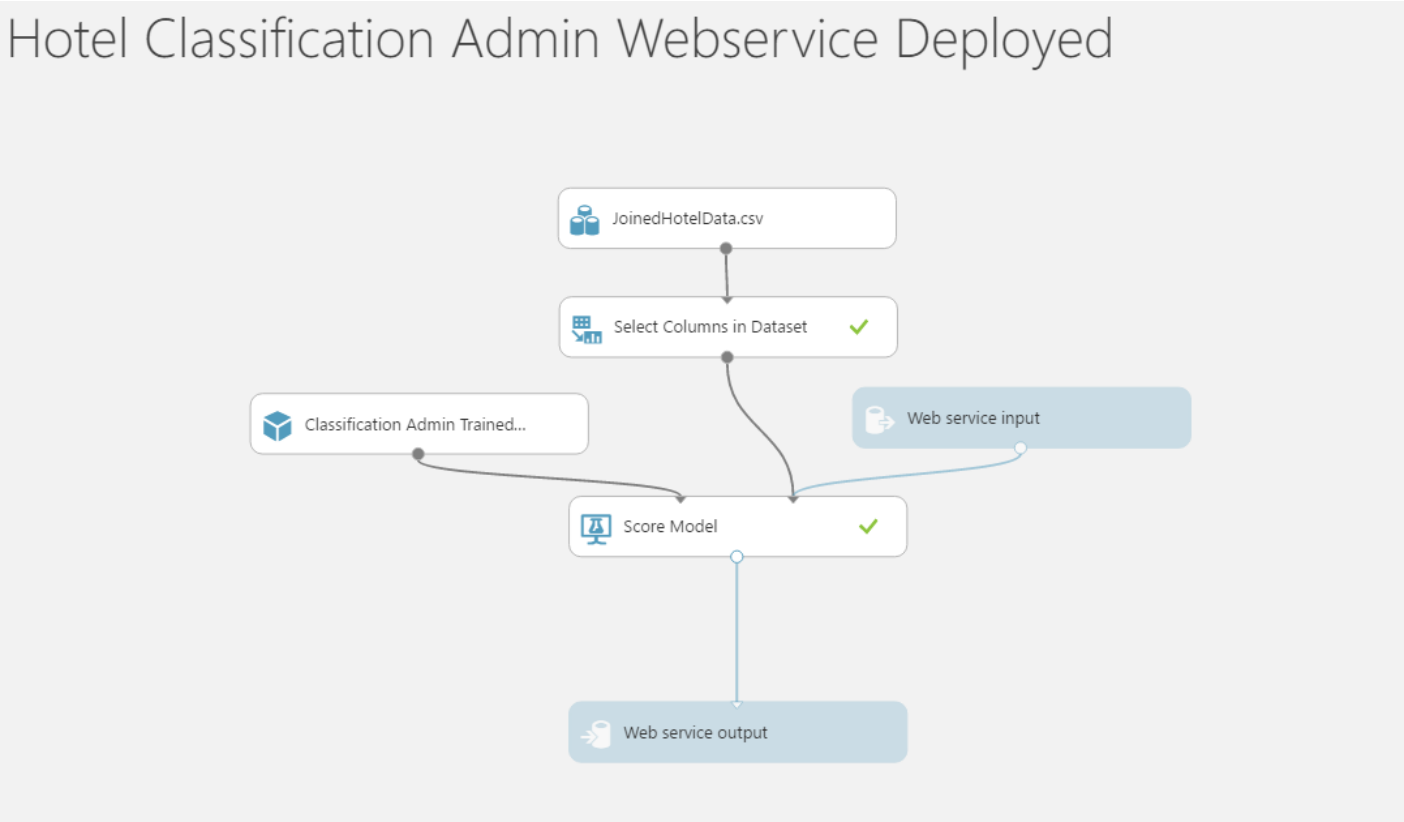
We choose Multiclass Decision Forest, which means that for precision, out of the times label '1' was predicted, 99.6% of the time the system was in fact correct. And for recall, it means that out of all the times label '1' should have been predicted only 99.6% of the labels were correctly predicted. (Please refer confusion matrix)

In essence, the more zeroes or smaller the numbers on all cells in the confusion but the diagonal, the better your classifier is doing.

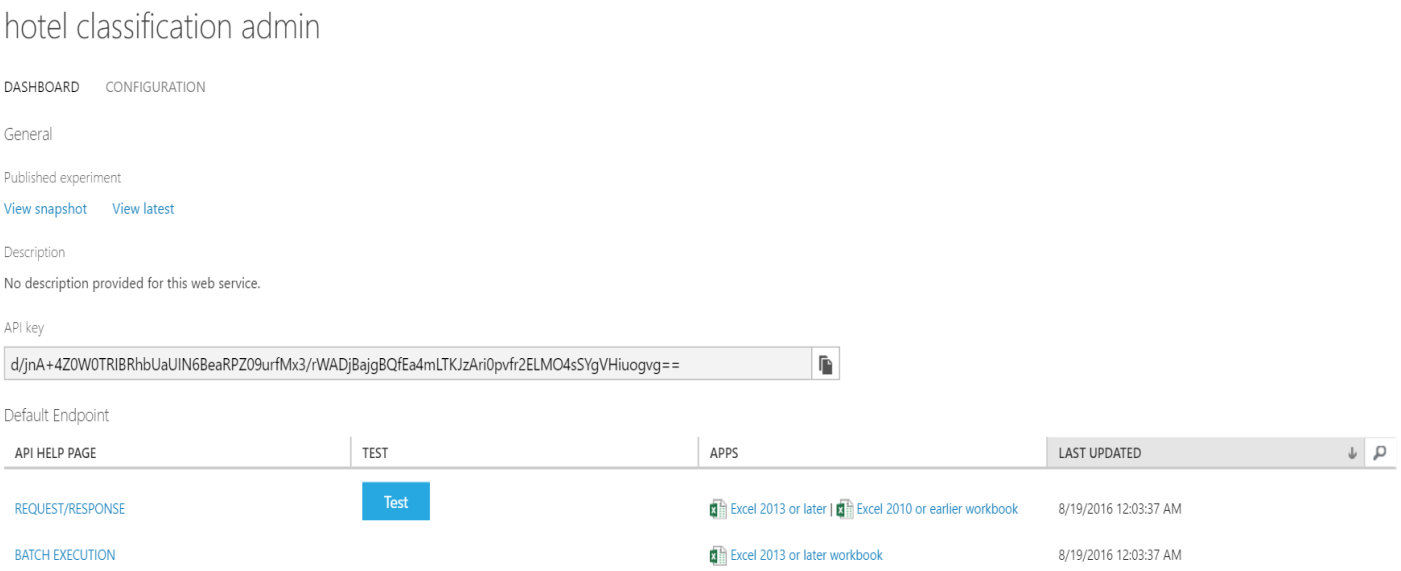
Hence, Multiclass Decision Forest has better accuracy, precision and recall rate than all other model.

1.4.6 Web Service

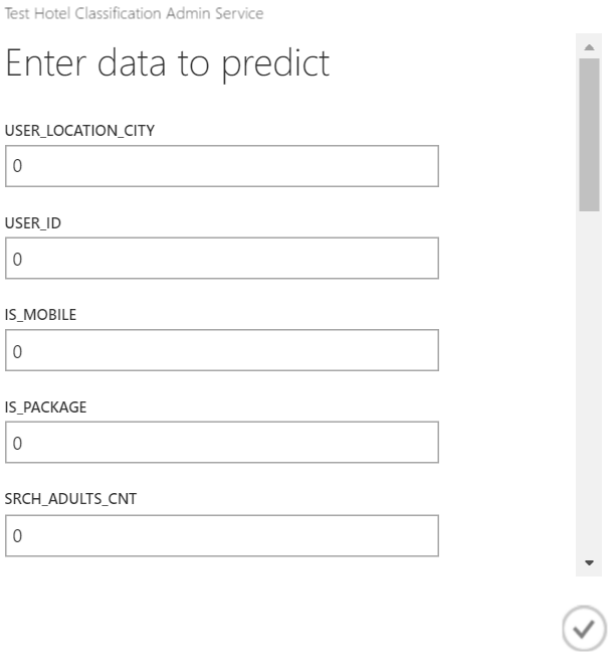
- Once the classification model is ready, we set up **Web Service**.
- The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**).
- Modules that were used for training are removed. Specifically:
    - Multi class decision forest
    - Train Model
    - Split Data
- Then we added the saved trained model back into the experiment.
- **Web service input** and **Web service output** modules are added.



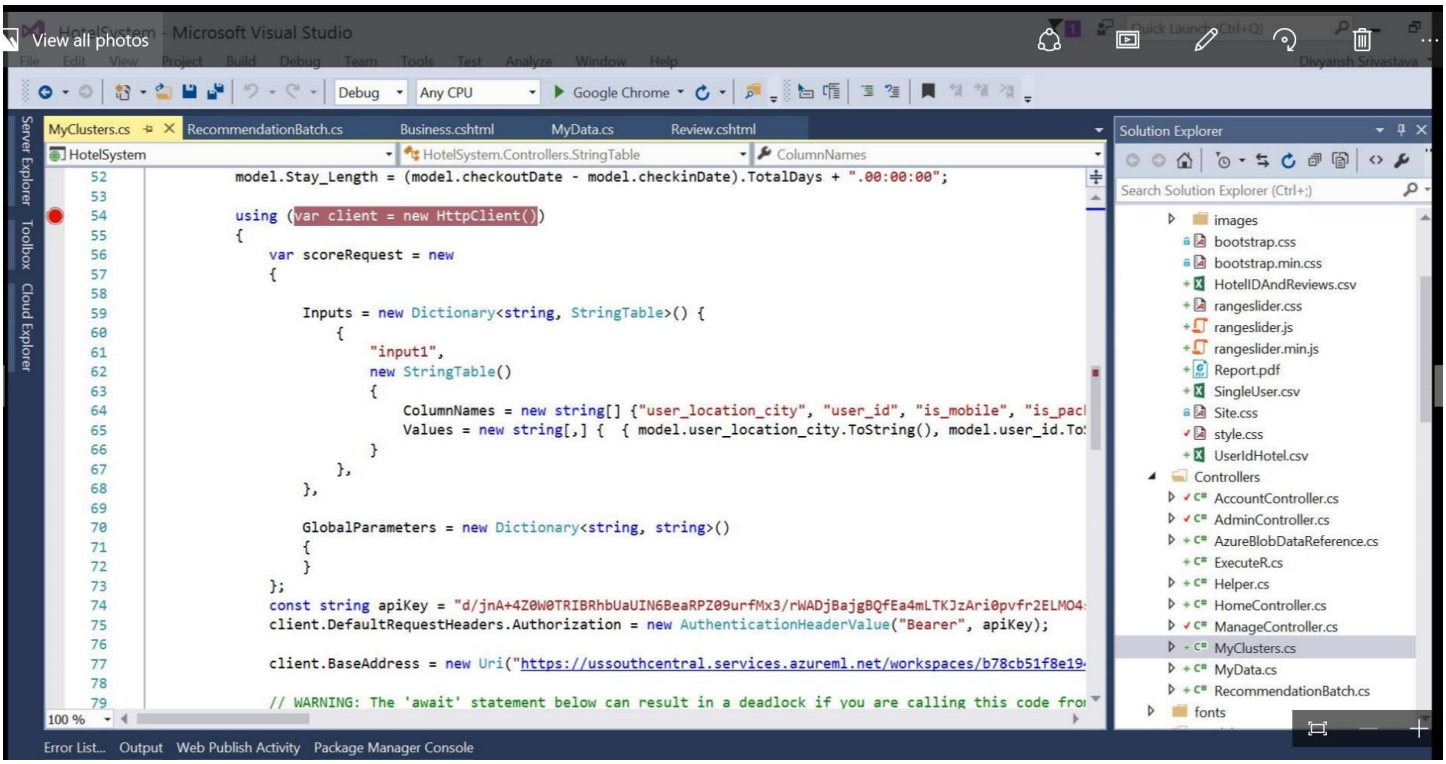
- Now run the model and publish the web service.



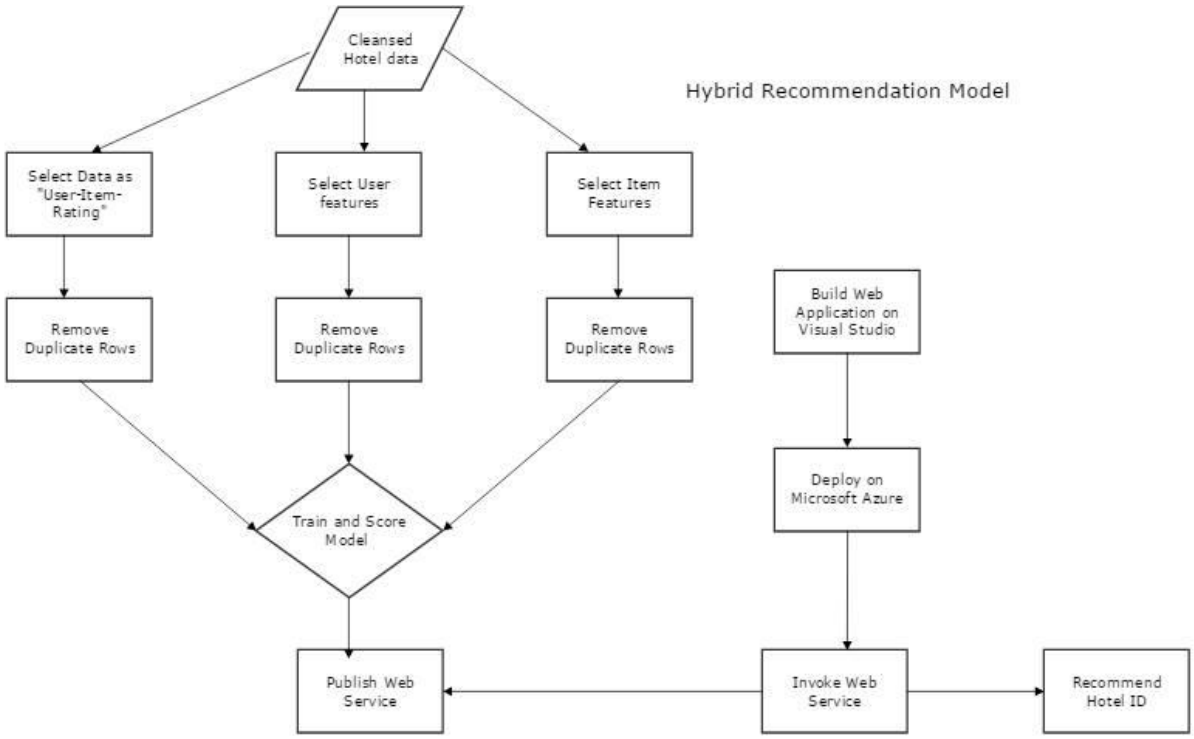
- On running the web service, we get the following form which can be used to invoke the web service and do prediction.



- Web application making call to the API and invoking the cluster web service.


View all photos
HotelSystem - Microsoft Visual Studio
Debug
Any CPU
Google Chrome
MyClusters.cs
RecommendationBatch.cs
Business.cshtml
MyData.cs
Review.cshtml
HotelSystem
HotelSystem.Controllers.StringTable
ColumnNames
model.Stay_Length = (model.checkoutDate - model.checkinDate).TotalDays + ".00:00:00";
using (var client = new HttpClient())
{
var scoreRequest = new
{
Inputs = new Dictionary<string, StringTable>() {
{
"input1",
new StringTable()
{
ColumnNames = new string[] {"user_location_city", "user_id", "is_mobile", "is_pac
Values = new string[,] { { model.user_location_city.ToString(), model.user_id.To
}
},
},
GlobalParameters = new Dictionary<string, string>()
{
}
};
const string apiKey = "d/jnA+4Z0W0TRIBRhbUaUIN6BeaRPZ09urfMx3/rWADjBajgBQfEa4mLTKJzAri0pvfr2ELMO4
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/b78cb51f8e19
// WARNING: The 'await' statement below can result in a deadlock if you are calling this code fro
Solution Explorer
Search Solution Explorer (Ctrl+;)
images
bootstrap.css
bootstrap.min.css
HotelIDAndReviews.csv
rangeslider.css
rangeslider.js
rangeslider.min.js
Report.pdf
SingleUser.csv
Site.css
style.css
UserIdHotel.csv
Controllers
AccountController.cs
AdminController.cs
AzureBlobDataReference.cs
ExecuteR.cs
Helper.cs
HomeController.cs
ManageController.cs
MyClusters.cs
MyData.cs
RecommendationBatch.cs
fonts
Error List...
Output
Web Publish Activity
Package Manager Console

1.5 Recommendation System



1.5.1 Overall Design

- ❖ This section explains the implementation and the application of a Recommendation system built around the dataset.
- ❖ The main aim of a recommendation system is to recommend one or more items to users of the system.
- ❖ There are two approaches to recommender systems, content-based and collaborative filtering. To implement a recommender system, we used the Matchbox Recommender on ML Studio.
- ❖ The Matchbox recommender on ML Studio combines both the approaches, Collaborative filtering and Content based thus also called as a hybrid recommender.

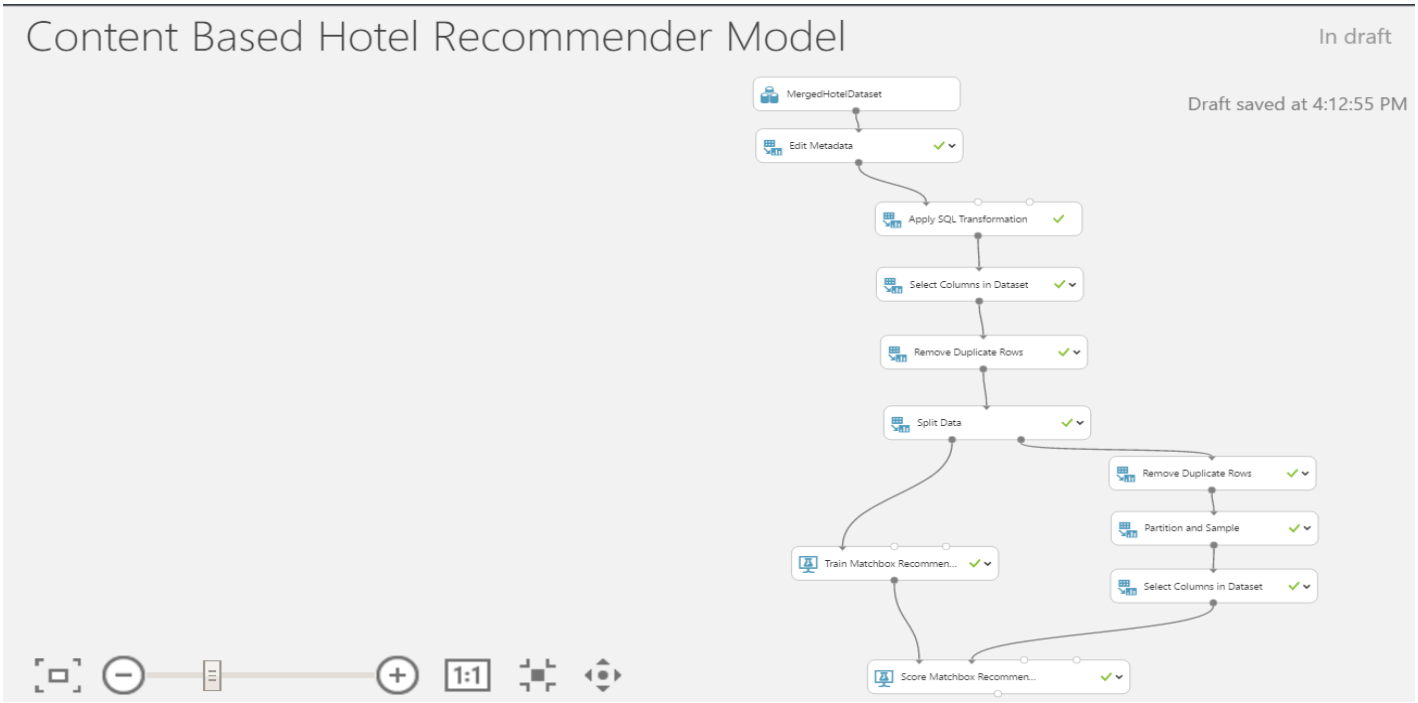
Recommender systems are classified according to the technique used to create the recommendation (fill the blanks in the utility matrix):

- Content-based systems examine properties of the items recommended and offer similar items
- Collaborative filtering (CF) systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users
- Hybrid mixing both previous approaches

1.5.2 Content Based Recommendation System

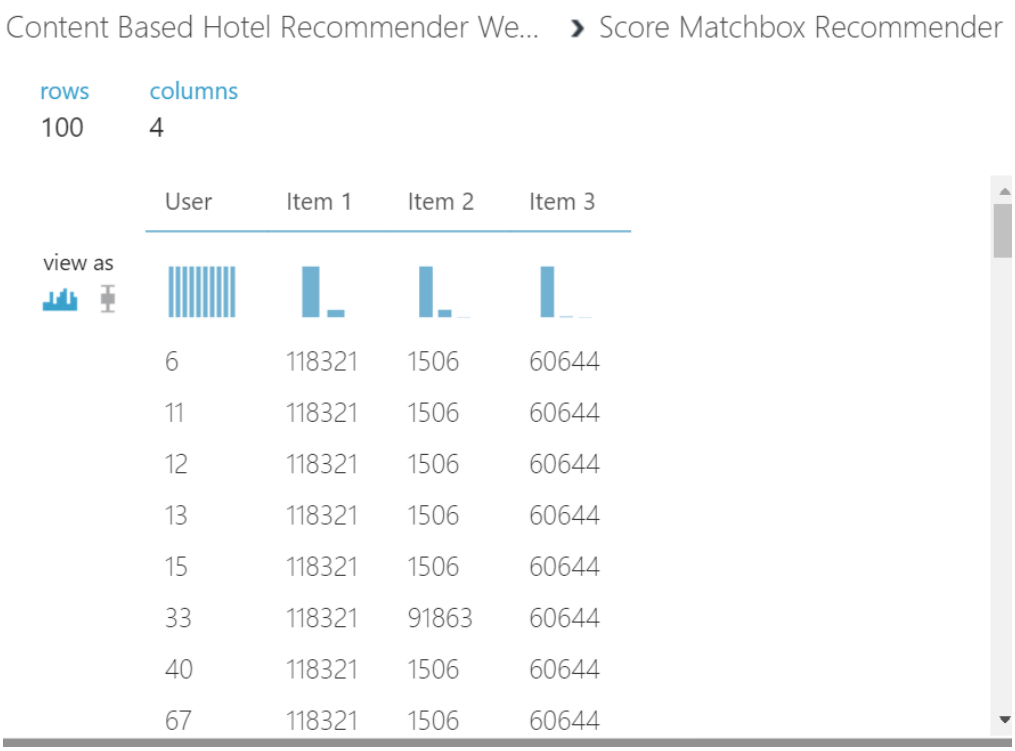
1.5.2.1 Build model

- We only want three columns, since the MatchBox recommender that we will be using for training a recommender only takes in a dataset of triples: (user, item, rating).
- Translating it into our case, we need a dataset of the following three columns: UserId, PropId and Rating.
- The matchbox recommender has a limit of 0-99 or 1-100 distribution of ratings so the model would not train.
- Thus we rounded up the ratings and thus now the range is from 1-5 in a R Script in MLStudio.



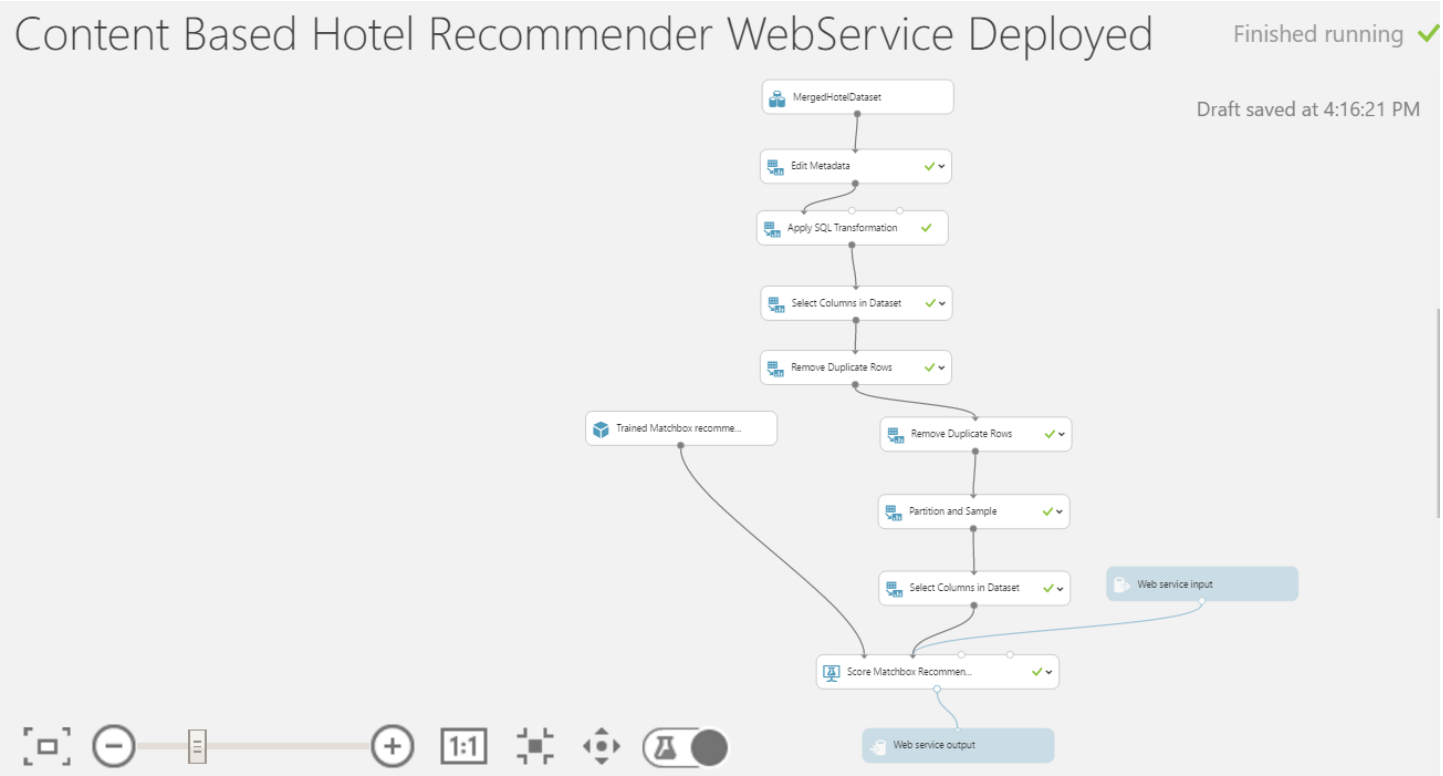
Training, Scoring and Evaluating Matchbox Recommender:

- To train the matchbox recommender, we split the data into test and train datasets using the recommender split option in the Split data module.
- Split the data in the proportion of 75% and 25%.
- Then we Add the Train MatchBox Recommender module and connect it the training data which returns a trained Matchbox recommender. We then used the Score Matchbox Recommender module, which creates recommendations for the different users.



1.5.2.2 Publish Content-Based Filtering Model as a Web Service

- The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**).
- Modules that were used for training are removed. Specifically:
  - Train Recommender model
  - Split Data
- Then we added the saved trained model back into the experiment.
- **Web service input** and **Web service output** modules are added.

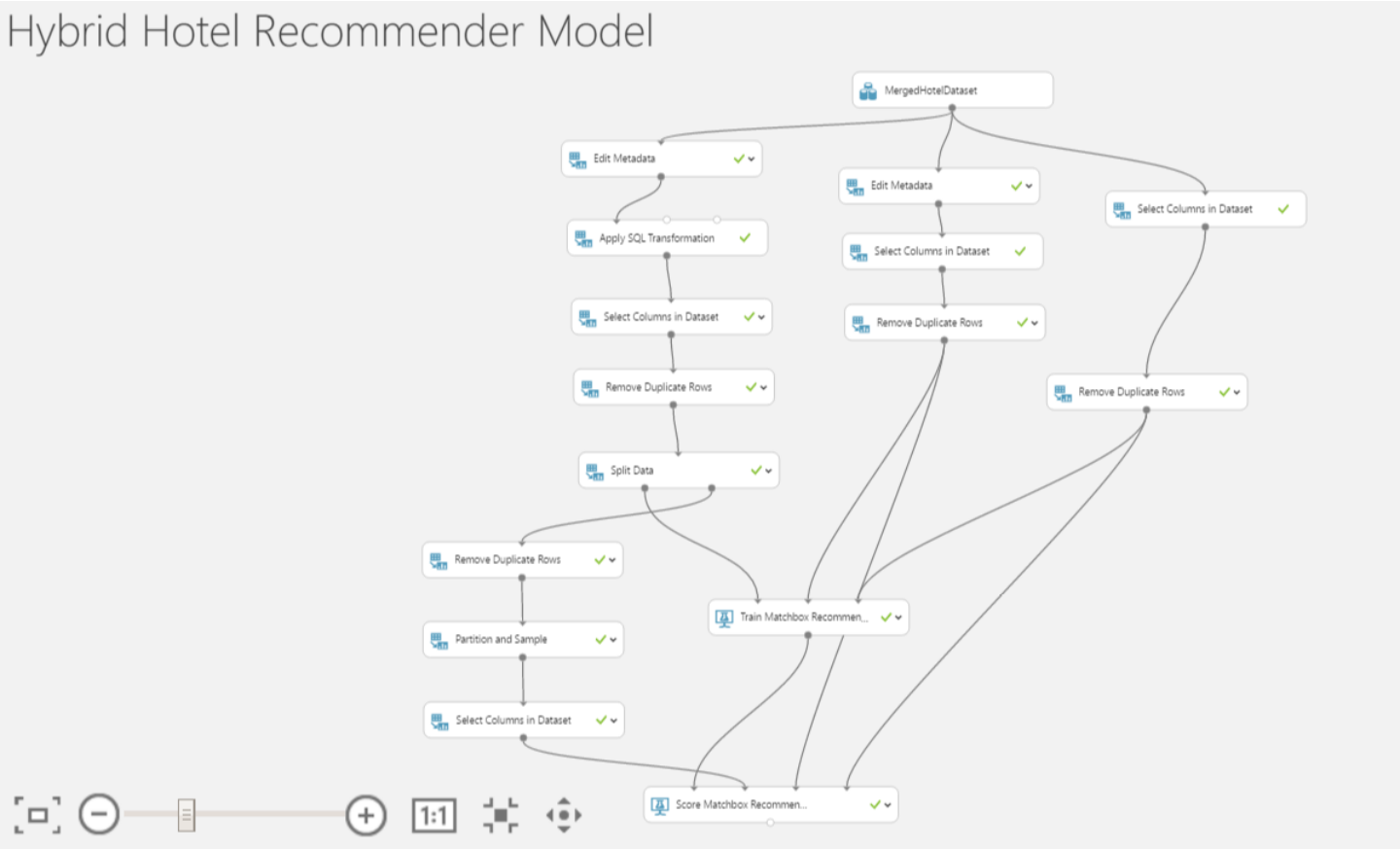


1.5.3 Hybrid Based Recommendation System

We now want to extend the content-/rating-based filtering approach to a hybrid recommender. This can be done by integrating user as well as item features - the remaining two inputs of the module Train Matchbox Recommender. User features encompass more information on the users, such as demographic information, while item features contain information on the hotels, e.g. usd, review score, destination Id

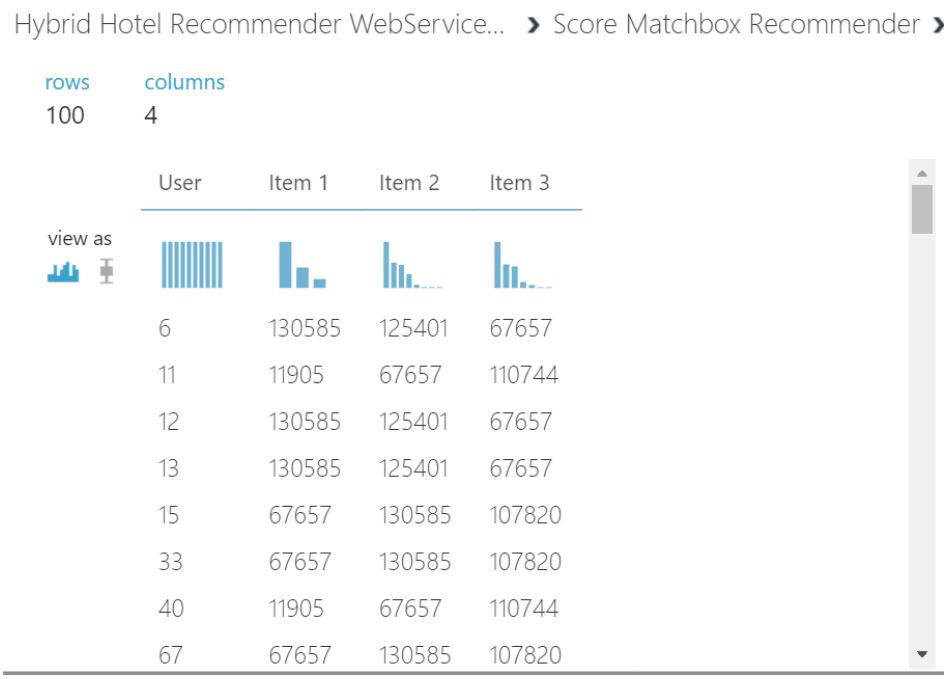
1.5.2.3 Build model

- ❖ We only want three columns, since the MatchBox recommender that we will be using for training a recommender only takes in a dataset of triples: (user, item, rating).
- ❖ Translating it into our case, we need a dataset of the following three columns: UserId, PropId and Rating.
- ❖ The matchbox recommender has a limit of 0-99 or 1-100 distribution of ratings so the model would not train.
- ❖ Thus we rounded up the ratings and thus now the range is from 1-5 in a R Script in MLStudio.



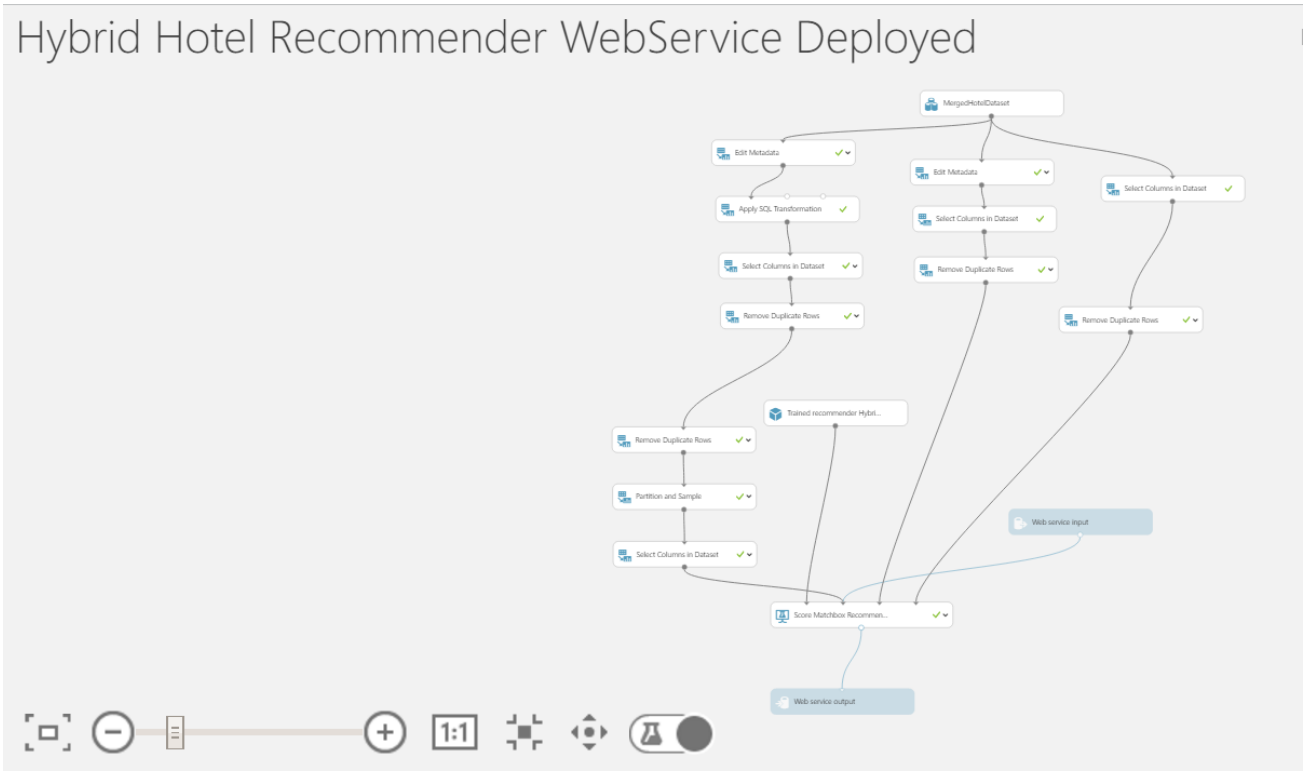
Training, Scoring and Evaluating Matchbox Recommender:

- To train the matchbox recommender, we split the data into test and train datasets using the recommender split option in the Split data module.
- Split the data in the proportion of 75% and 25%.
- Then we Add the Train MatchBox Recommender module and connect it the training data which returns a trained Matchbox recommender. We then used the Score Matchbox Recommender module, which creates recommendations for the different users.

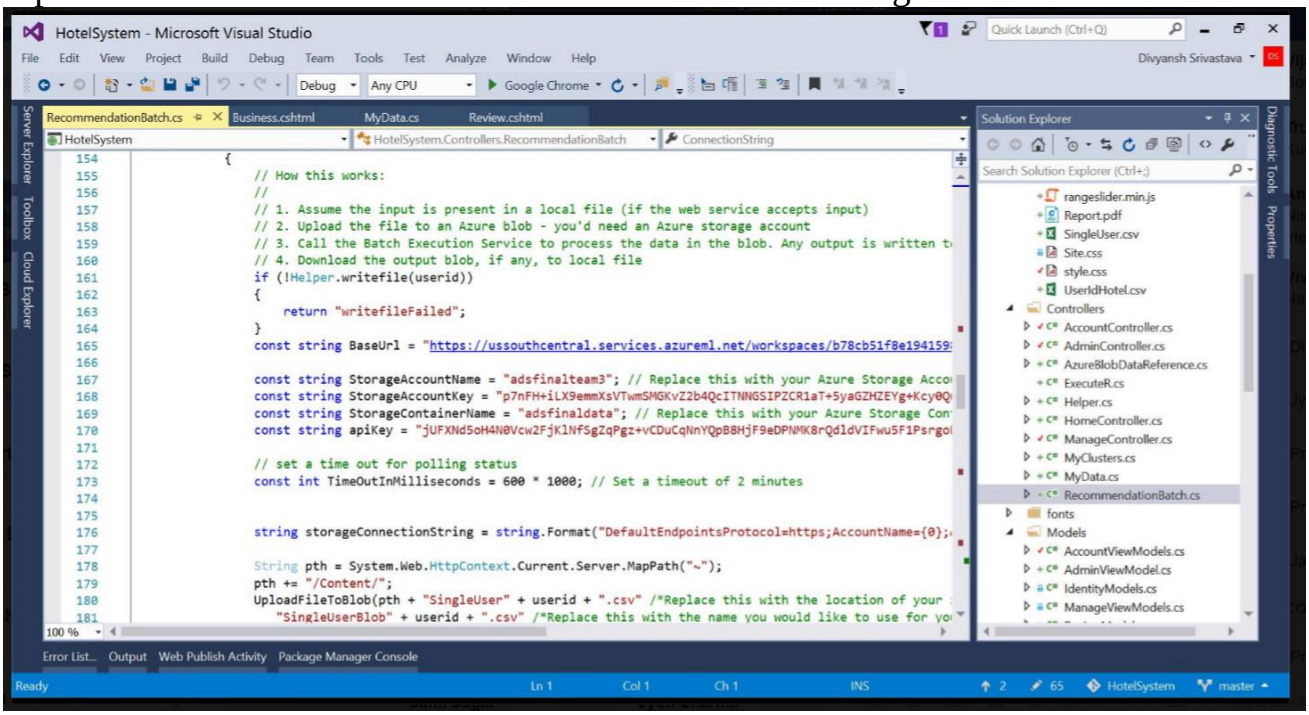


1.5.2.4 Publish Hybrid recommendation model as a Web Service

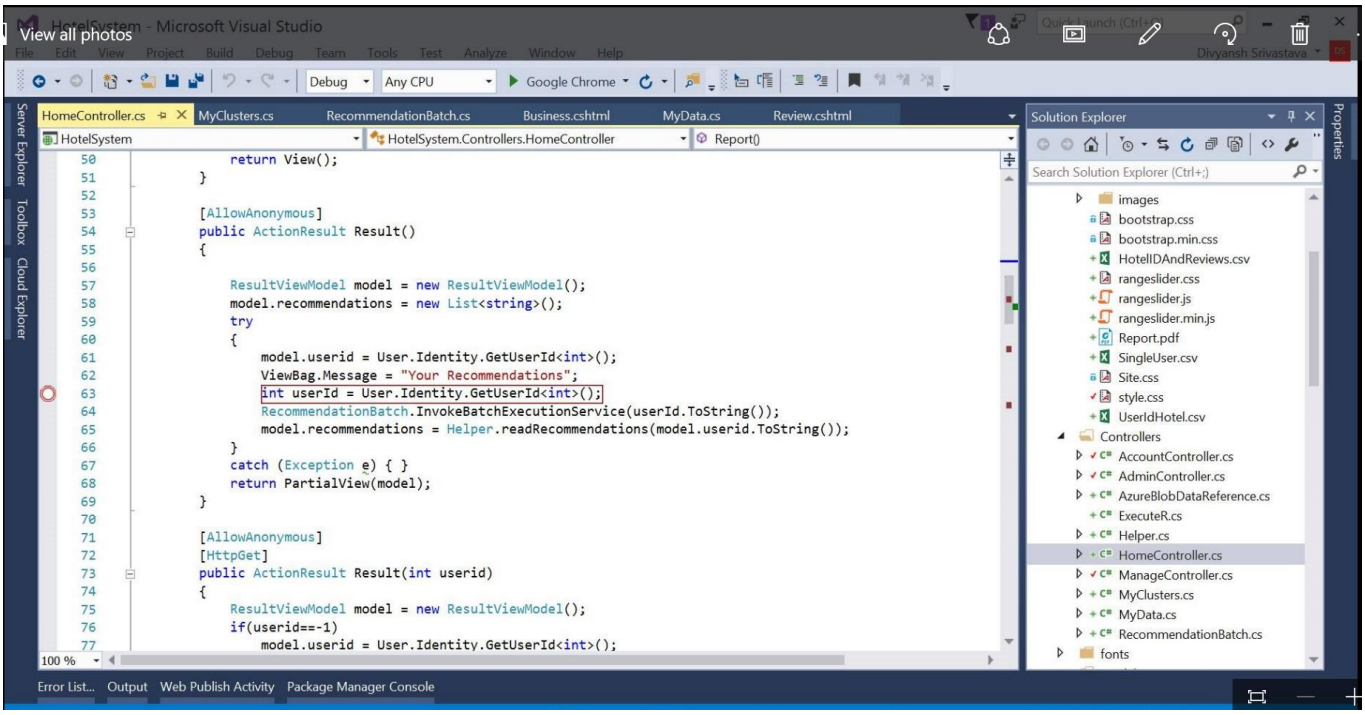
- The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**).
- Modules that were used for training are removed. Specifically:
  - Train Recommender model
  - Split Data
- Then we added the saved trained model back into the experiment.
- **Web service input** and **Web service output** modules are added.



- Input from the user is read and is stored in Azure blob storage.



- Batch execution service processes the data on blob and gives an output file which is again dumped on azure blob.
- Output is read from this file and displayed back to the user.

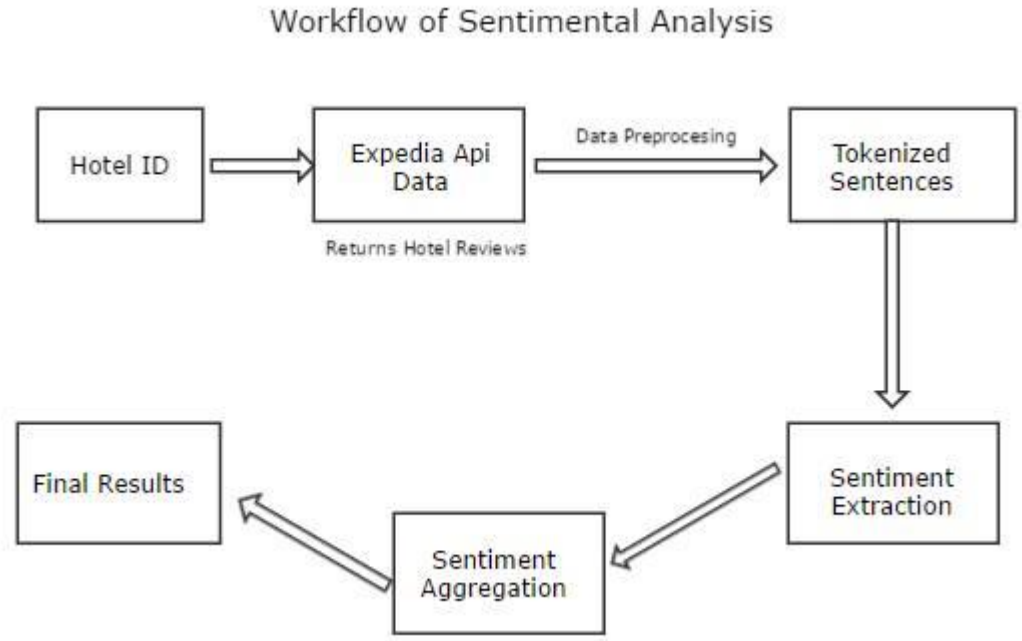


1.6 Sentimental Analysis

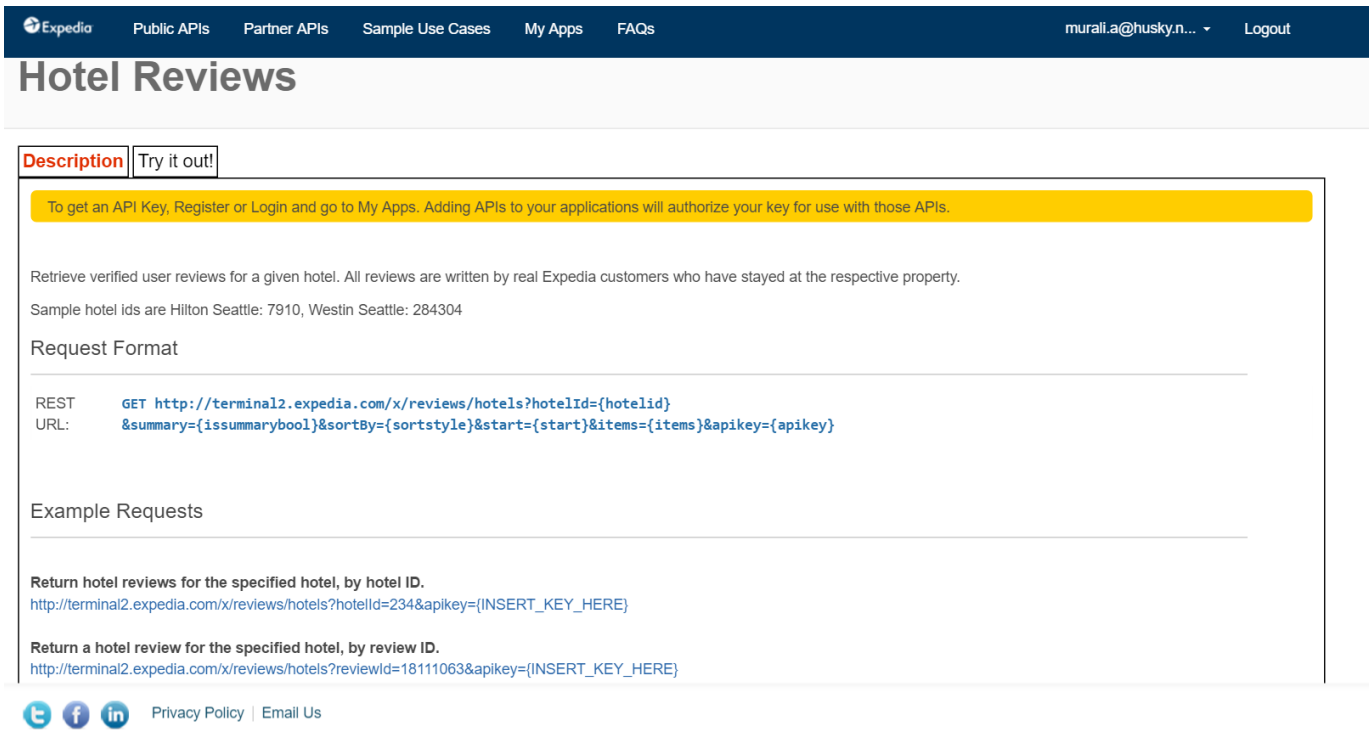
Sentiment analysis (also known as **opinion mining**) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation, affective state, or the intended emotional communication.

1.6.1Overall Design



- We use a public API from Expedia to get reviews of all those hotels.



- Code snippet to access the API and fetch all hotel reviews.

```
json_file <- paste("http://terminal2.expedia.com/x/reviews/hotels?hotelId=",hotelId,"&apikey=JvMtGQVznEaKqvkEpYFZHSYEqINUhwhH",sep="",collapse ="")
#Read data from file
json_data <- fromJSON(file=json_file)


reviewVector <- vector()
#iterate through the reviews of hotel and read the text
reviewOjectLength <- length(json_data$reviewDetails$reviewCollection$review)
for(reviewNumber in 1:reviewOjectLength)
{
  if(!(length(json_data$reviewDetails$reviewCollection$review) == 0))
  {
   # print("step1 : no review")
  if(!(is.null(json_data$reviewDetails$reviewCollection$review[[reviewNumber]])))
  {
  reviewVector <- c(reviewVector,json_data$reviewDetails$reviewCollection$review[[reviewNumber]]["reviewText"])
  }
  }
}
```

- A word cloud helps us to visualize the most common words in the reviews and have a general feeling of the reviews.

```
 if(!(length(reviewVector) == 0))
 {
# print("step2")
 #Converting vectors to matrix
hotelReviewMatrix <- matrix(reviewVector ,ncol = 1, byrow=TRUE)

# Create corpus
corpus=Corpus(VectorSource(reviewVector))

# Convert to lower-case
corpus=tm_map(corpus,tolower)

# Remove stopwords
corpus=tm_map(corpus,function(x) removeWords(x,stopwords()))

# convert corpus to a Plain Text Document
corpus=tm_map(corpus,PlainTextDocument)

col=brewer.pal(6,"Dark2")
wordcloud(corpus, min.freq=25, scale=c(5,2),rot.per = 0.25,
          random.color=T, max.word=45, random.order=F,colors=col)
```

Word cloud:

- We will use lexicon based sentiment analysis. A list of positive and negative opinion words or sentiment words for English has been attached.

```
# print("step3")
 #lexicon based sentiment analysis
 positives= readLines("positive-words.txt")
 negatives = readLines("negative-words.txt")

 reviewScore <- data.frame()
 for(row in 1:nrow(hotelReviewMatrix))
 {
   review = hotelReviewMatrix[row,1]

   review = gsub("[[:punct:]]", "", review)    # remove punctuation
   review = gsub("[[:cntrl:]]", "", review)   # remove control characters
   #review = gsub('\d+', "", review)          # remove digits
   review = str_replace_all(review, "[^[:alnum:]]", " ")

   # Let's have error handling function when trying tolower
   tryTolower = function(x){
     # create missing value
     y = NA
     # tryCatch error
     try_error = tryCatch(tolower(x), error=function(e) e)
     # if not an error
     if (!inherits(try_error, "error"))
       y = tolower(x)
     # result
     return(y)
   }
```

- We calculate the score for each word in the review by matching it with positive and negative list of words. And finally an overall score is calculated which is more of a consolidated review of all the users of that hotel.

```
# use tryTolower with sapply
review = sapply(review, tryTolower)

require(stringr)
# split sentence into words with str_split function from stringr package
word_list = str_split(review, " ")
words = unlist(word_list)

# compare words to the dictionaries of positive & negative terms
positive.matches = match(words, positives)
negative.matches = match(words, negatives)

# get the position of the matched term or NA
# we just want a TRUE/FALSE
positive_matches = !is.na(positive.matches)
negative_matches = !is.na(negative.matches)

# final score
score = sum(positive_matches) - sum(negative_matches)
reviewScore = c(reviewScore,score)
```

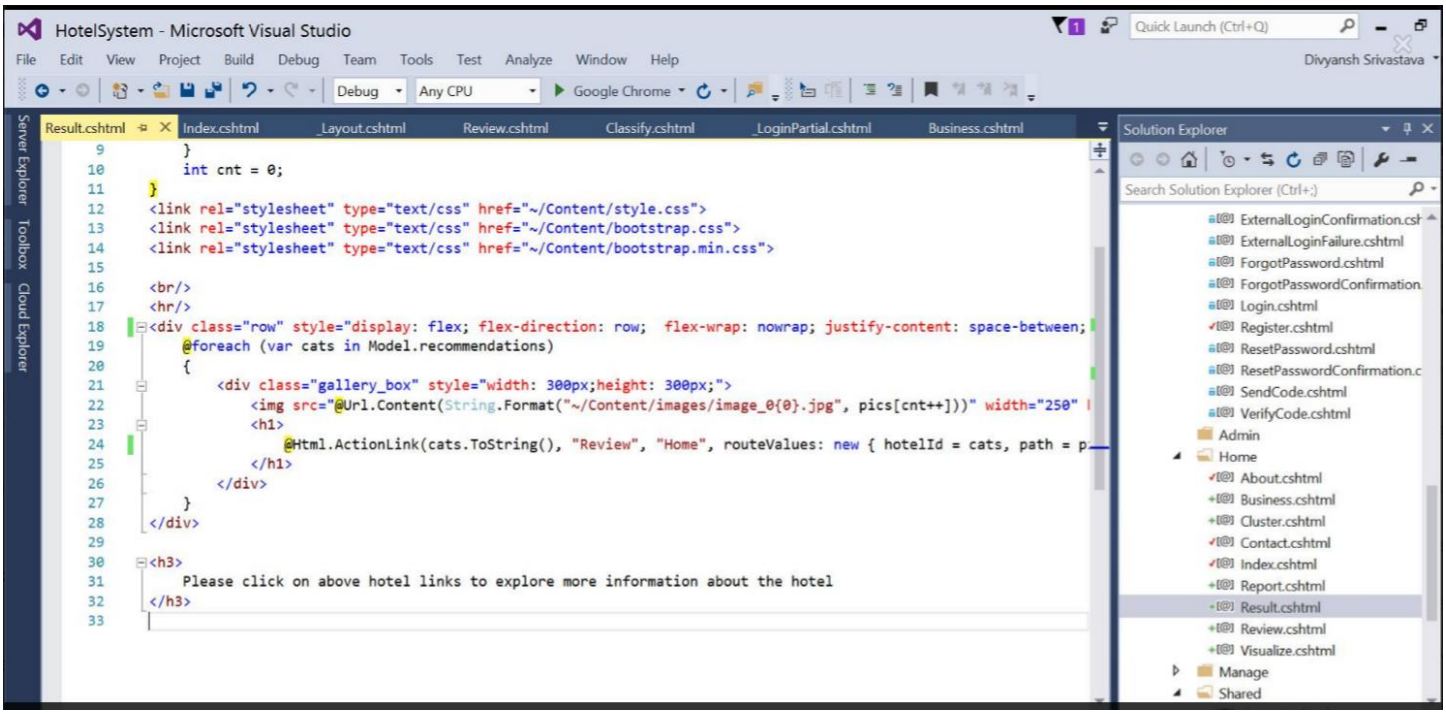
1.6.2 Integration with Web Application

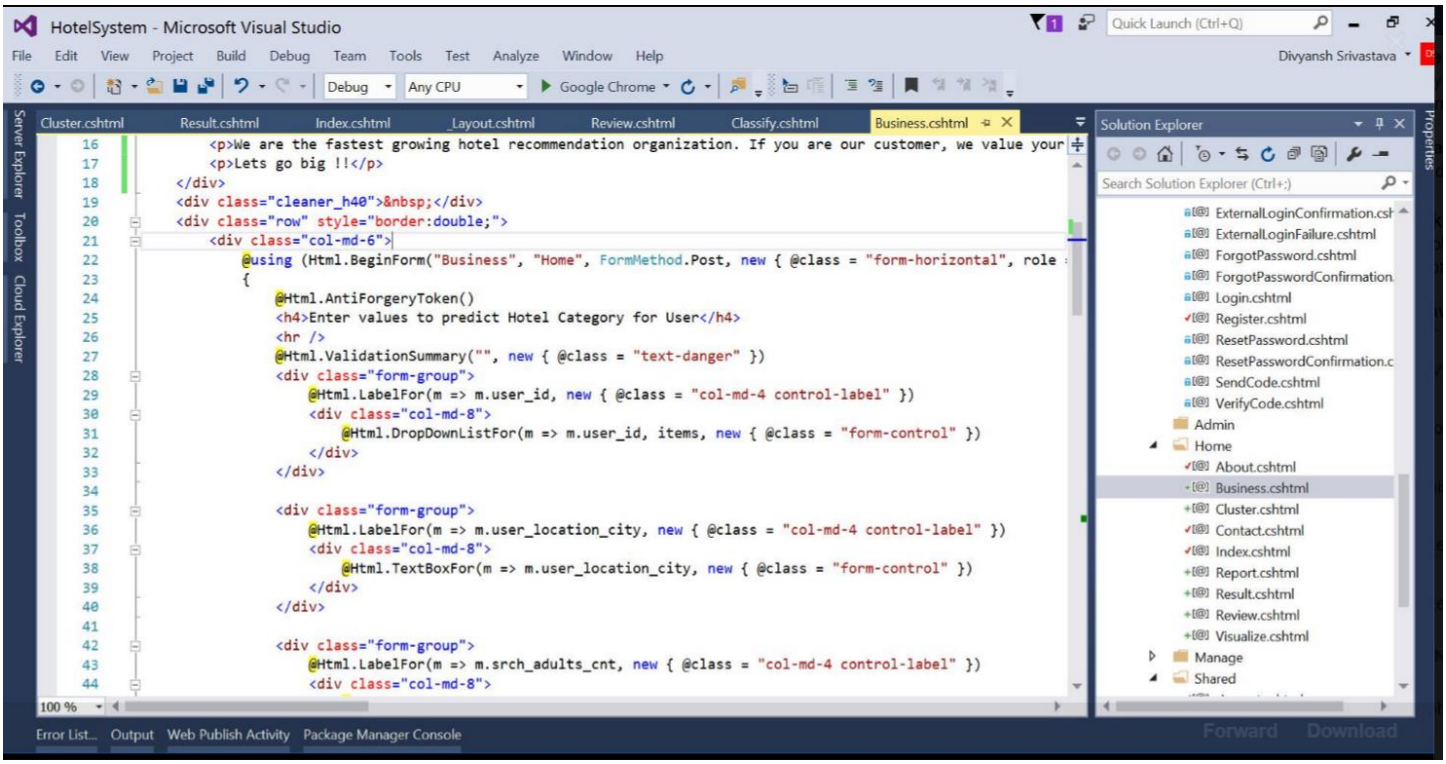
- This file with hotel Id and review score is stored on azure blob. Based on the score, a traveler rating is assigned to each hotel that the customer can view on log -in.

1.7 Web Application

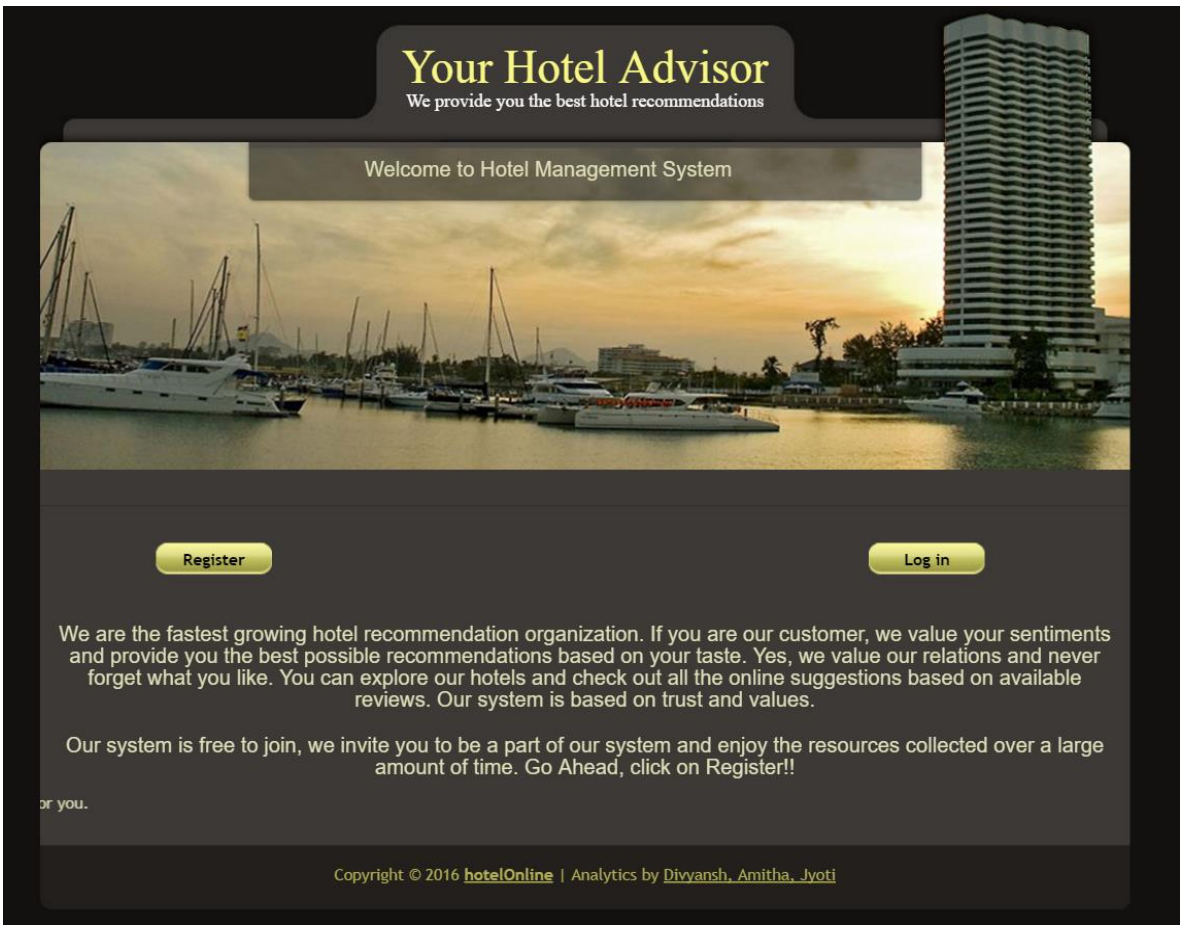
We created a MVC ASP.net application on visual studio and then deployed the web application on Microsoft Azure.

Website link: http://hoteladvisorsystem.azurewebsites.net/

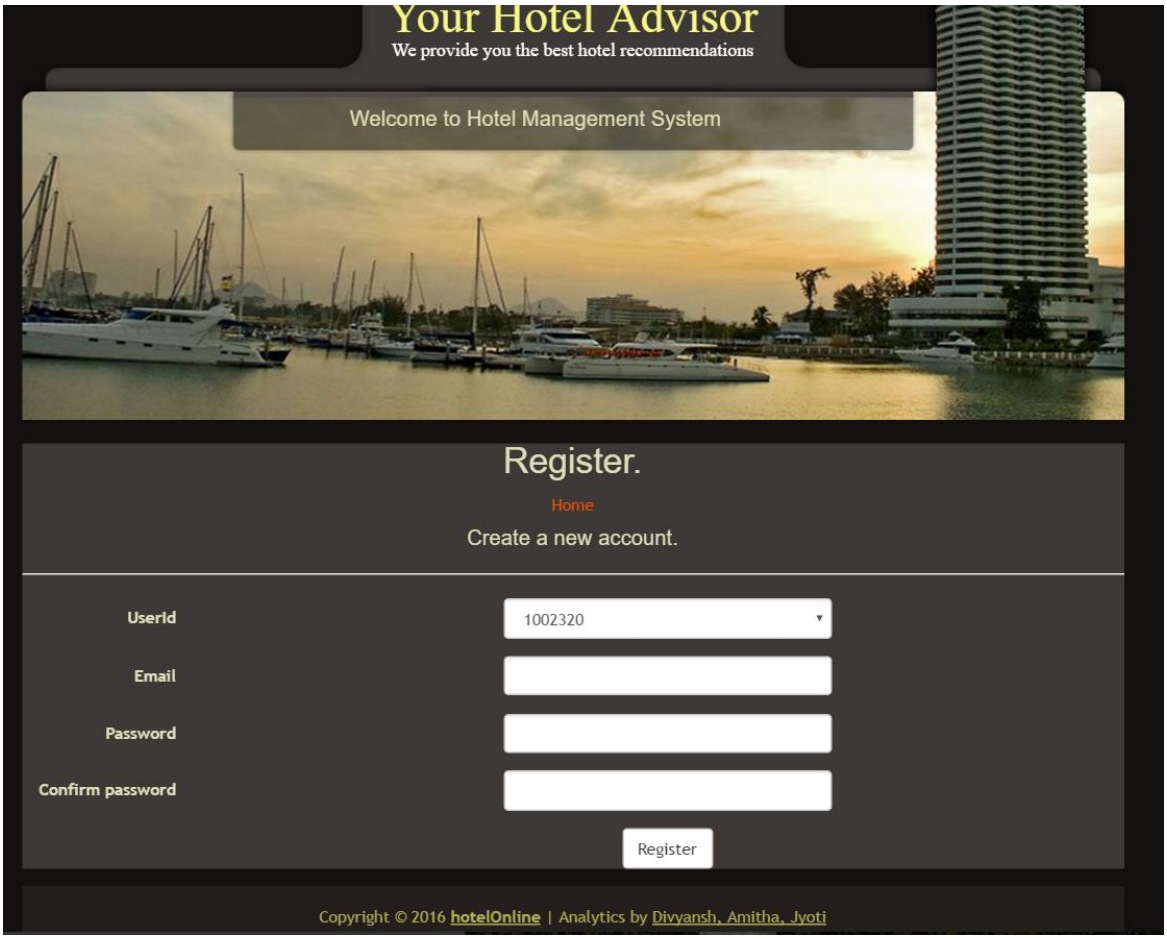

HotelSystem - Microsoft Visual Studio
Result.cshtml
Index.cshtml
_Layout.cshtml
Review.cshtml
Classify.cshtml
_LoginPartial.cshtml
Business.cshtml
Solution Explorer
Please click on above hotel links to explore more information about the hotel


HotelSystem - Microsoft Visual Studio
Cluster.cshtml
Result.cshtml
Index.cshtml
_Layout.cshtml
Review.cshtml
Classify.cshtml
Business.cshtml
Solution Explorer
Enter values to predict Hotel Category for User

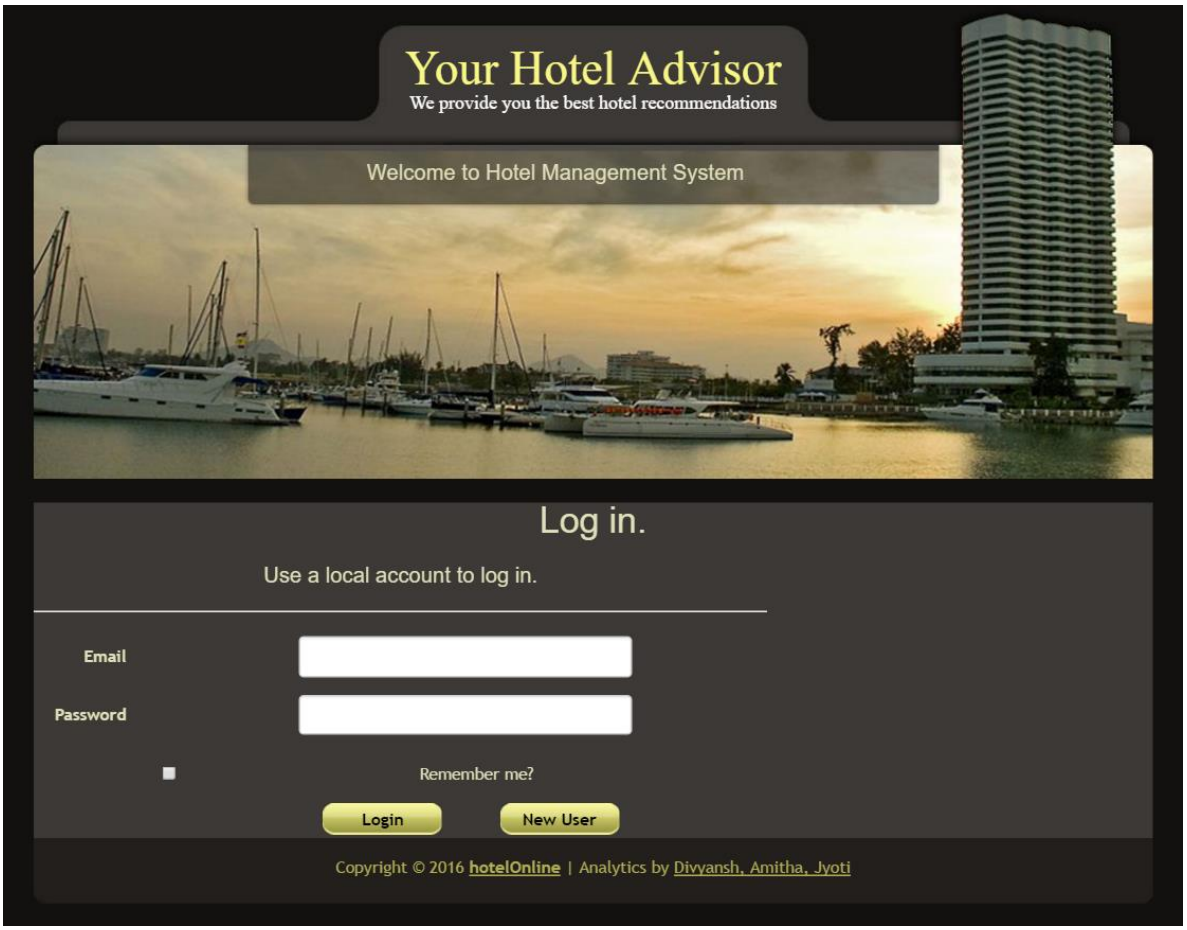
- ✓ **Hotel Advisor Website**



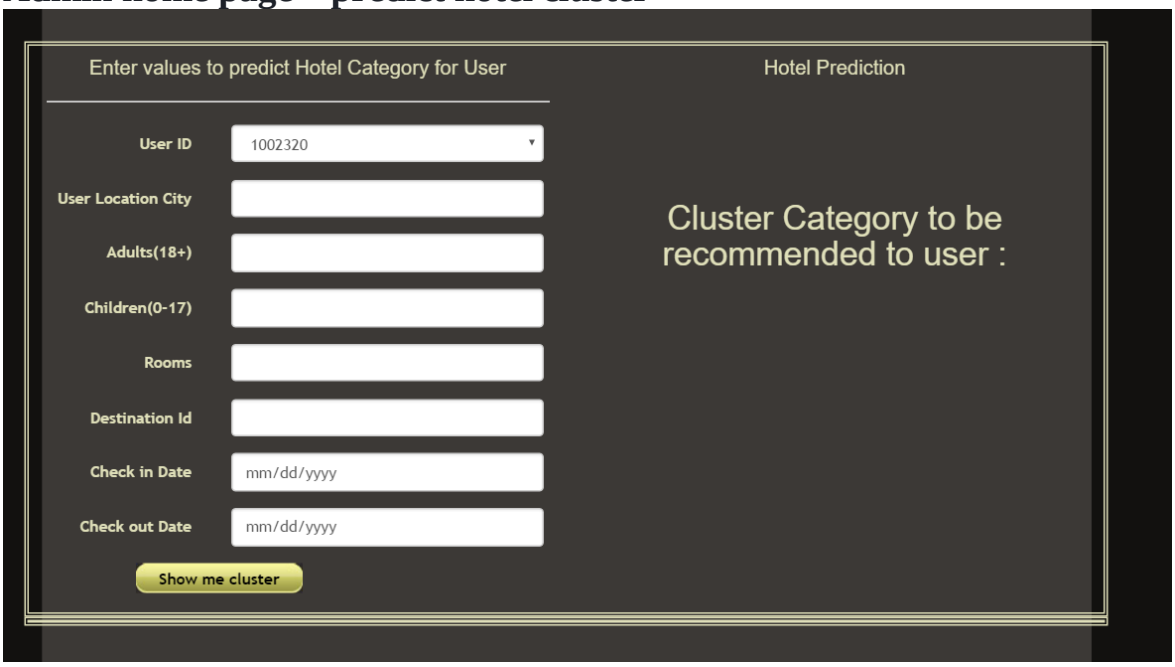
- ✓ **Register page**


Your Hotel Advisor
We provide you the best hotel recommendations
Welcome to Hotel Management System
Register.
Home
Create a new account.
UserId
1002320
Email
Password
Confirm password
Register
Copyright © 2016 hotelOnline | Analytics by Divyansh, Amitha, Jyoti

✓ **Sign-in page**



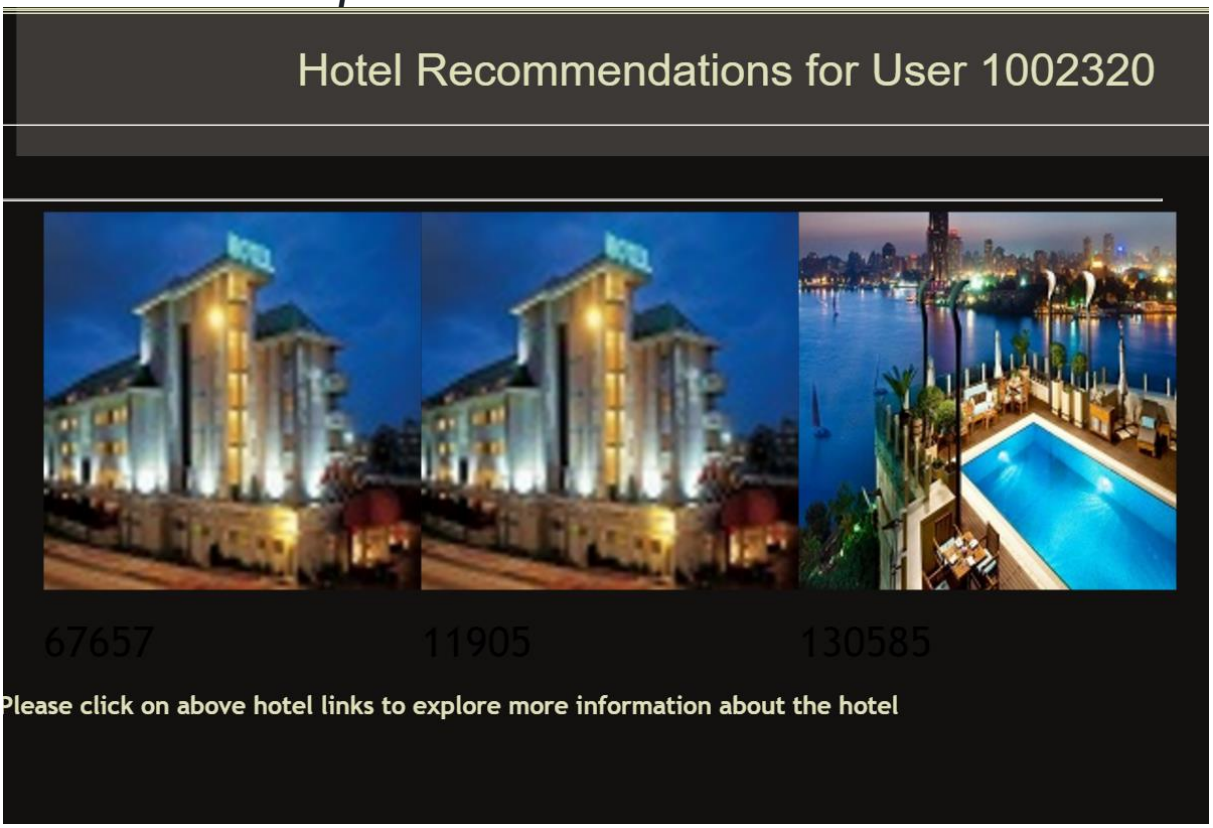
✓ **Admin home page – predict hotel cluster**



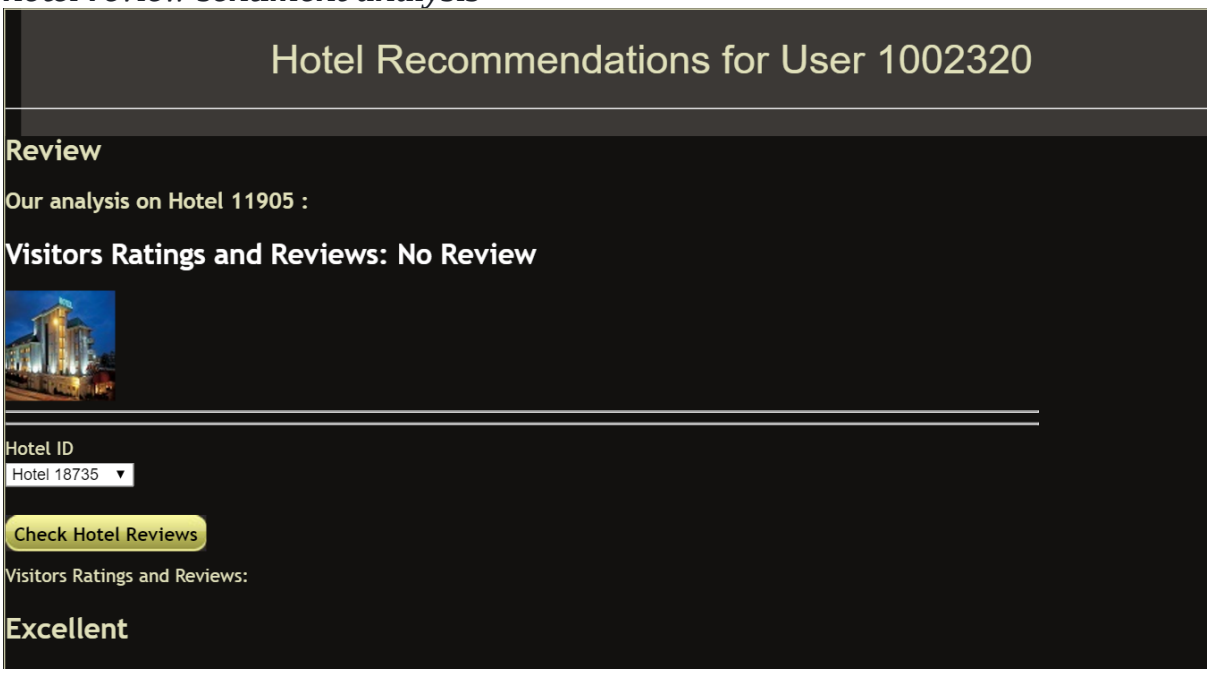
User case Data:

User ID	User Location City	Adults(18+)	Children(0-17)	Rooms	Destination ID	Check in Date	Check out Date
307065	17494	2	0	1	11835	6/27/2017 12:00:00 AM	6/30/2017 12:00:00 AM
194208	48262	2	0	1	8855	1/10/2017 12:00:00 AM	1/17/2017 12:00:00 AM
613844	27731	2	0	1	8744	5/9/2017 12:00:00 AM	5/18/2017 12:00:00 AM
1053478	53517	2	0	1	11353	8/13/2017 12:00:00 AM	8/17/2017 12:00:00 AM
936906	36086	3	0	3	5405	4/4/2017 12:00:00 AM	4/8/2017 12:00:00 AM
653198	47997	1	0	1	8260	10/6/2016 12:00:00 AM	10/10/2016 12:00:00 AM
154070	27102	2	2	2	8740	11/27/2016 12:00:00 AM	11/28/2016 12:00:00 AM
806146	4687	3	1	2	17520	6/18/2017 12:00:00 AM	6/19/2017 12:00:00 AM
720068	34868	5	0	2	23820	4/28/2017 12:00:00 AM	5/10/2017 12:00:00 AM
257191	41779	2	5	1	11939	12/13/2016 12:00:00 AM	12/16/2016 12:00:00 AM
1155959	53078	3	0	2	8289	8/19/2017 12:00:00 AM	8/27/2017 12:00:00 AM

Recommended hotels for user



hotel review sentiment analysis



- ✓ **Customer home page**

1.8 References

1. **http://www.text-analytics101.com/2014/10/computing-precision-and-recall-for.html**
   Computing Precision and Recall for Multi-Class Classification Problems

2. **https://github.com/oliviak/Recommender-in-Azure/tree/master/4%20Content-Filtering%20and%20Hybrid%20recommender**
   Content Based and Hybrid Recommendation System

3. **https://www.r-bloggers.com/sentiment-analysis-on-donald-trump-using-r-and-tableau/**
   Sentiment Analysis in R studio