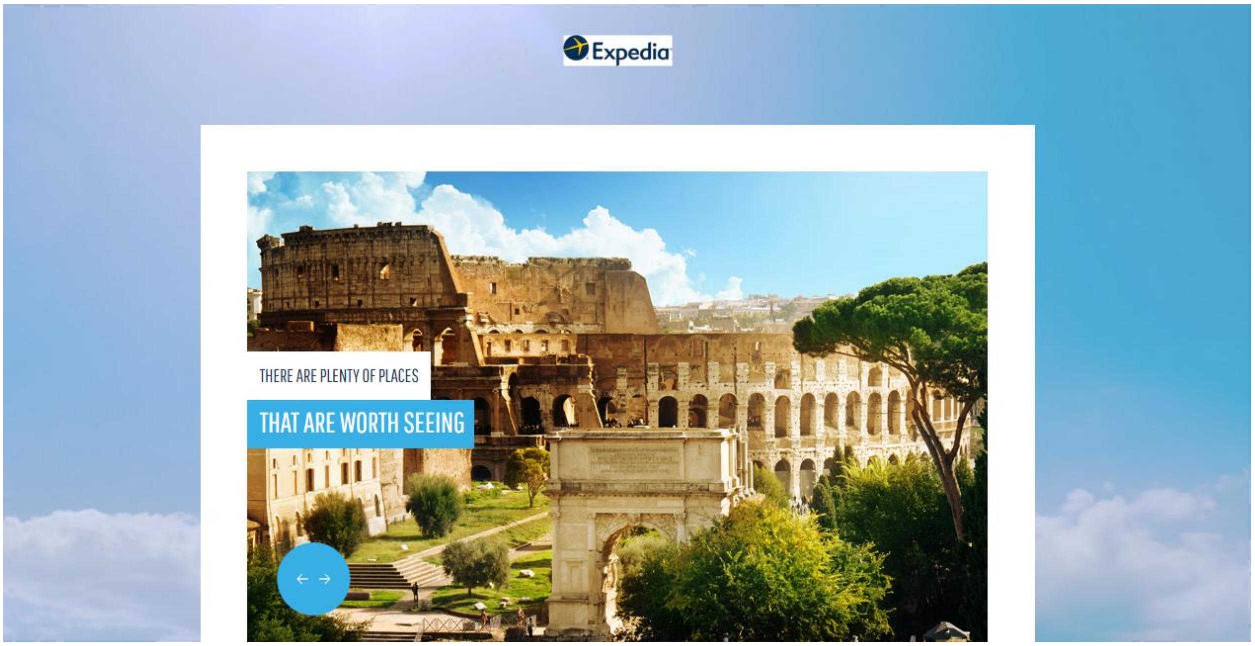


Expedia Hotel Recommendation

Website: http://www.expediahotelrecommendation.com/

Team 10
Vishwa Shah
Avinita Shetty
Jun Wu

Table of Contents

Background

Challenge:

Which hotel type will an Expedia customer book?

Expedia wants to optimize hotel search by providing personalized hotel recommendations to their users. This is no small task for a site with hundreds of millions of visitors every month!

Currently, Expedia uses search parameters to adjust their hotel recommendations, but there aren't enough customer specific data to personalize them for each user. In the competition, Expedia is expecting Kagglers to contextualize customer data and predict the likelihood a user will stay at 100 different hotel groups.

Goals

Based on the dataset provided by Expedia, we are planning to achieve the following goals by performing multiple ways of data analytics, and comparing then applying data mining models.

- Perform data exploration and extract meaningful information for business users
- Predict which group of hotels the customer performing the search would likely to book.
- Provide business users/ customers a user-friendly web access to observe data on web pages

Dataset

Data Description:

Logs of customer behavior (separate training and testing dataset)

Include customer profile(User ID, Geo location, search site, device), search event information (date, check-in/check-out day, destination info).

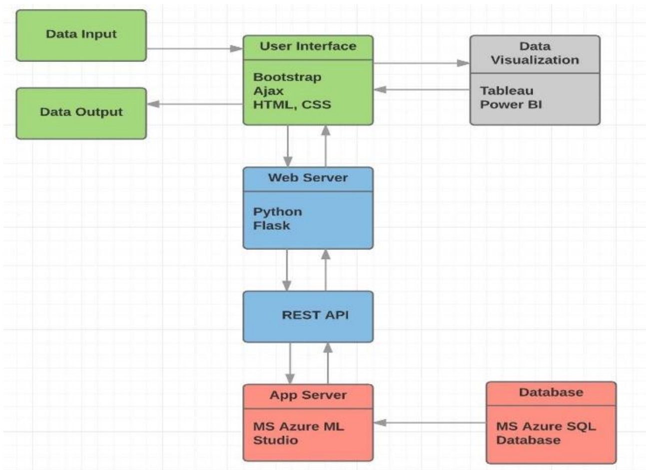
train/test.csv

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

Project Plan

The project has been developed through multiple phases. The major steps are listed below.

1. Data Exploration
2. Data Preparation
3. Data Visualization
4. Modeling
5. Web Deployment



Also, we followed our product architecture and achieved most of our functionalities when we finished our project. In the following sessions, detailed project breakdown which covers the actions and deliverables for each developing phase will be reported.

Project Report

Data Exploration

Tool: Python

By using Python and leveraging libraries such as Numpy, Pandas, Matplot and Seaborn for data summarization and visualization, we managed to evaluate the data quality and understand our business problems in different aspects.

The dataset is sized over 3.5 gb, over 36 millions of rows. We extracted the first 1 million rows for and inspect the data quality using python for the purpose of efficiency. According to the result, 21 out of 24 columns has no missing values (regardless of improper values). For column srch_ci and srch_co about 0.1% data are missing. And for column orig_destination_distance, almost 40% percent of data are missing.

```
In [4]: import numpy as np
        import pandas as pd
        train = pd.read_csv('C:/Users/vanwu/Desktop/INFO 7390 ADS/Expedia data set/train.csv', header = 0, sep = ',', nrows = 1000000)
        train.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000000 entries, 0 to 999999
Data columns (total 24 columns):
date_time                    1000000 non-null object
site_name                    1000000 non-null int64
posa_continent               1000000 non-null int64
user_location_country        1000000 non-null int64
user_location_region         1000000 non-null int64
user_location_city           1000000 non-null int64
orig_destination_distance    629753 non-null float64
user_id                      1000000 non-null int64
is_mobile                    1000000 non-null int64
is_package                   1000000 non-null int64
channel                      1000000 non-null int64
srch_ci                      999002 non-null object
srch_co                      999001 non-null object
srch_adults_cnt              1000000 non-null int64
srch_children_cnt            1000000 non-null int64
srch_rm_cnt                  1000000 non-null int64
srch_destination_id          1000000 non-null int64
srch_destination_type_id     1000000 non-null int64
is_booking                   1000000 non-null int64
cnt                          1000000 non-null int64
hotel_continent              1000000 non-null int64
hotel_country                1000000 non-null int64
hotel_market                 1000000 non-null int64
hotel_cluster                1000000 non-null int64
dtypes: float64(1), int64(20), object(3)
memory usage: 190.7+ MB
```

According to the data description, our data includes time-based customer events which may lead to a sales event (booking through Expedia website). In order to have a better understanding of their business situation, two charts has been developed for the convenience of observing the booking/clicking trends from 2013 to 2015.

```
In [9]: import matplotlib.pyplot as plt
        import seaborn as sns

        sns.set_style('whitegrid')
        sns.set(color_codes=True)

        def string_to_datetime(s, fmt='%Y-%m-%d'):
            if s != s:
                return np.nan
            year, month, day = s.split('-')
            try:
                d = pd.datetime(int(year), int(month), int(day))
            except ValueError:
                d = pd.datetime(2017, 1, 1)
            d = min([max([d, pd.datetime(2013, 1, 1)]), pd.datetime(2017, 1, 1)])
            return d
```

In [10]:
```
import numpy as np
import pandas as pd
train = pd.read_csv("C:/Users/vanwu/Desktop/INFO 7390 ADS/Expedia data set/train.csv", usecols=['date_time', 'is_booking', 'srch_ci', 'srch_co'],
                    parse_dates=['date_time'])
train['srch_ci'] = train['srch_ci'].apply(string_to_datetime)
train['srch_co'] = train['srch_co'].apply(string_to_datetime)
train.info()
train_bookings = train[train['is_booking'] == 1].drop('is_booking', axis=1)
train_clicks = train[train['is_booking'] == 0].drop('is_booking', axis=1)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37670293 entries, 0 to 37670292
Data columns (total 4 columns):
date_time     datetime64[ns]
srch_ci       datetime64[ns]
srch_co       datetime64[ns]
is_booking    int64
dtypes: datetime64[ns](3), int64(1)
memory usage: 1.4 GB
```

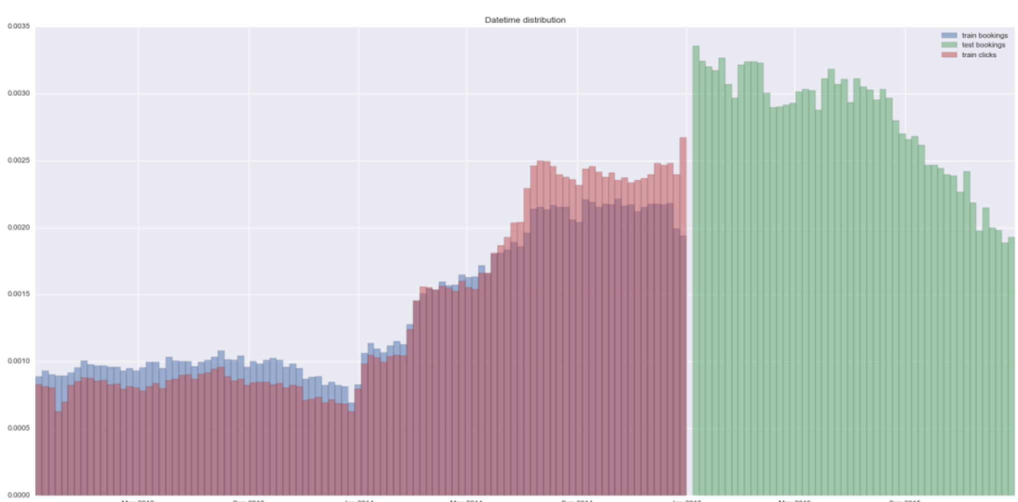
In [15]:
```
del train
test_bookings = pd.read_csv("C:/Users/vanwu/Desktop/INFO 7390 ADS/Expedia data set/test.csv", usecols=['date_time', 'srch_ci', 'srch_co'],
                    parse_dates=['date_time'])
test_bookings['srch_ci'] = test_bookings['srch_ci'].apply(string_to_datetime)
test_bookings['srch_co'] = test_bookings['srch_co'].apply(string_to_datetime)
test_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2528243 entries, 0 to 2528242
Data columns (total 3 columns):
date_time    datetime64[ns]
srch_ci      datetime64[ns]
srch_co      datetime64[ns]
dtypes: datetime64[ns](3)
memory usage: 77.2 MB
```

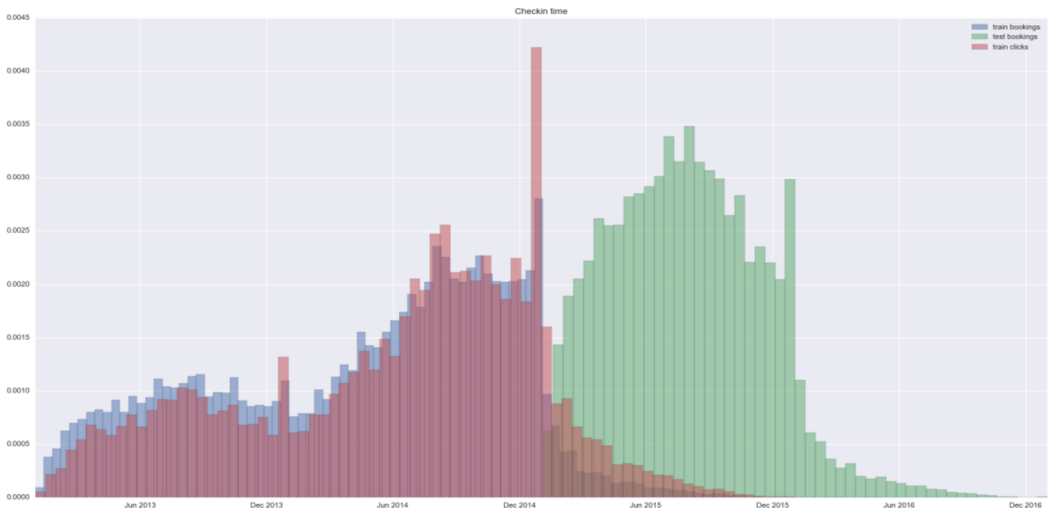
In [16]:
```
f = plt.figure()
plt.hist(train_bookings['date_time'].values, bins=100, alpha=0.5, normed=True, label='train bookings')
plt.hist(test_bookings['date_time'].values, bins=50, alpha=0.5, normed=True, label='test bookings')
plt.hist(train_clicks['date_time'].values, bins=100, alpha=0.5, normed=True, label='train clicks')
plt.title('Datetime distribution')
plt.legend(loc='best')
f.savefig('Time.png', dpi=300)
plt.show()
```

In [17]:
```
f = plt.figure()
plt.hist(train_bookings['srch_ci'].values, bins=100, alpha=0.5, normed=True, label='train bookings')
plt.hist(test_bookings['srch_ci'].dropna().values, bins=50, alpha=0.5, normed=True, label='test bookings')
plt.hist(train_clicks['srch_ci'].dropna().values, bins=100, alpha=0.5, normed=True, label='train clicks')
plt.title('Checkin time')
plt.legend(loc='best')
f.savefig('CheckinTime.png', dpi=300)
plt.show()
```

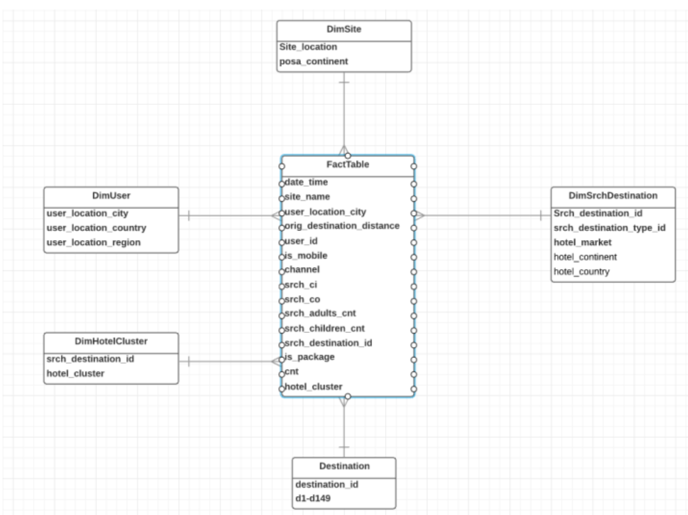
The chart below shows the booking number changes by time. The time frame was the date time when the customer events happen. Also, we observed that expedia business had a fast growth in 2014 and 2015.



The chart below shows customers' travelling trends over time. The time frame is the hotel check-in dates the customers booked or searched. We found that the majority of travelling, which is indicated by hotel check-in happens during summer and December, which fits the reality of hotel industry.



Also, by reviewing each column of the first dataset, we noticed the hierarchical relationships of five groups of variables. We decided to build a dimensional data model for two reasons: First, our dataset has a large number of rows, by reducing the columns containing redundant values, data size can be optimized, which will improve the efficiency of data storage and visualization. Second, with the variable which represents the lowest granularity of each hierarchy, we will be able to build our classification model with less features but same accuracy.



By the end of data preparation, we managed to gain a good understanding of our dataset. Also, we planned out our actions in second phase - Data Preparation.

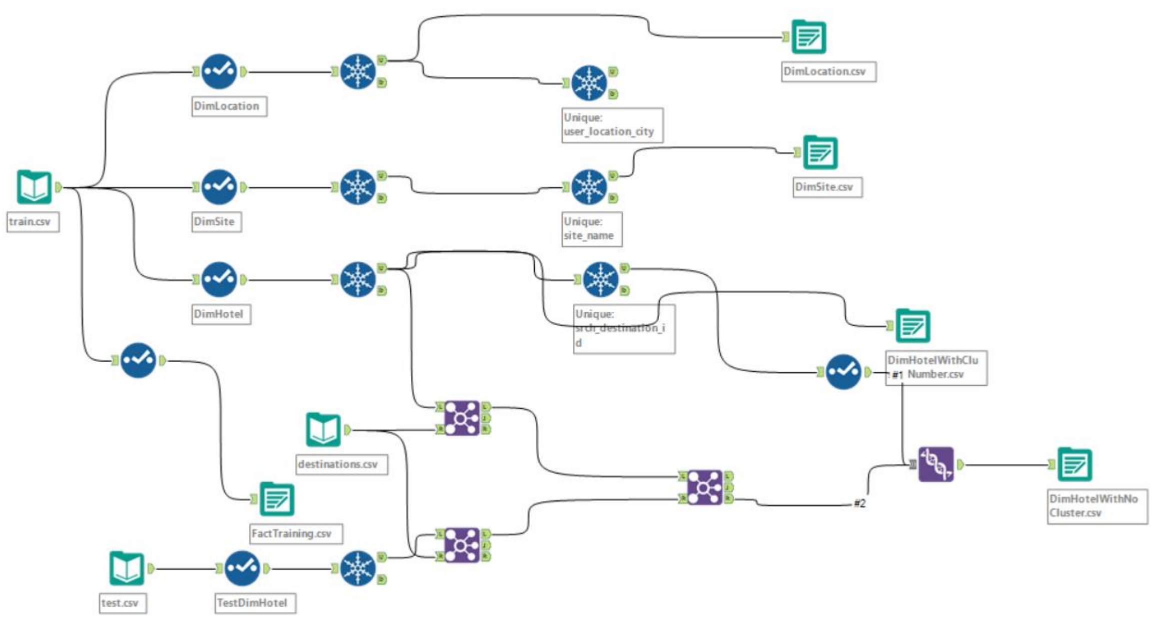
Data Preparation

Tool: Alteryx

In this step we managed to perform data cleaning and transformation to convert our dataset into a star schema for the convenience of visualization and analysis.

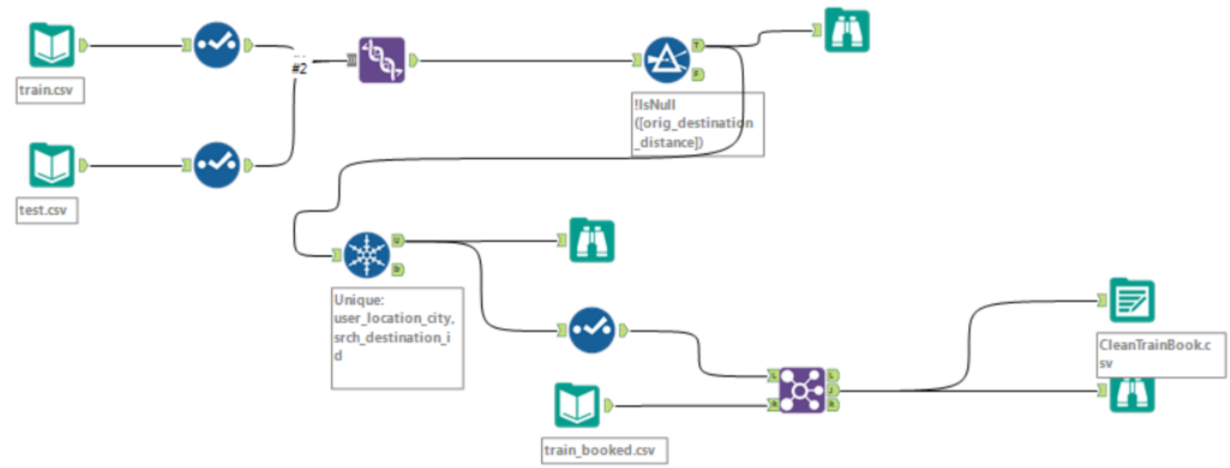
Using the data preparation tool Alteryx, we were able to build up pipelines to perform multiple data transformation, manipulation process step by step, and by the end, generate our data output.

There are a variety of widgets used in the pipeline. To build each dimension table, we first select the columns belonging to the hierarchy, followed with filtering out duplicated values. For the fact table which still contains every individual customer event, we kept the columns which act as a foreign key and dropped the columns that can be reconnected using these foreign keys. In the special use case of handling the new search_destination which cannot be found in the reference table, we managed to filter out these rows by joining the result of unique search_destination list from both training and testing dataset with their reference table.



About missing value handling, we deleted the rows with missing values in search_ci and search_co columns, the row number that has been dropped was less than 0.1% of the entire dataset, which is acceptable.

For Orig_dest_distance due to a large amount of missing values (about 40%), as well as the lack of geographical information, it was difficult to estimate missing value by using statistical methods. As a result, we decide to only fill in the values whose starting point (user_location_city) and destination point (search_dest_id) are known from our existing datasets.



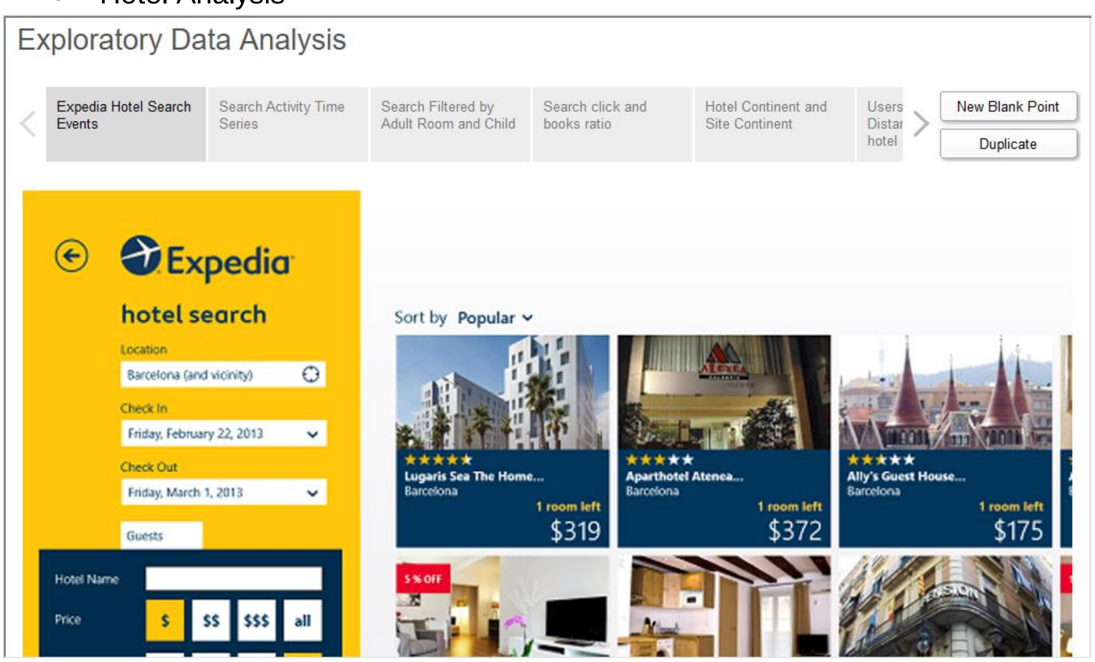
By the end, we were able to fill in about 20 percent of the missing data. For the rest of the missing values, since we have a large dataset and we expect the rows with no missing values are enough for modeling purpose, we decided to drop the rows with missing values.

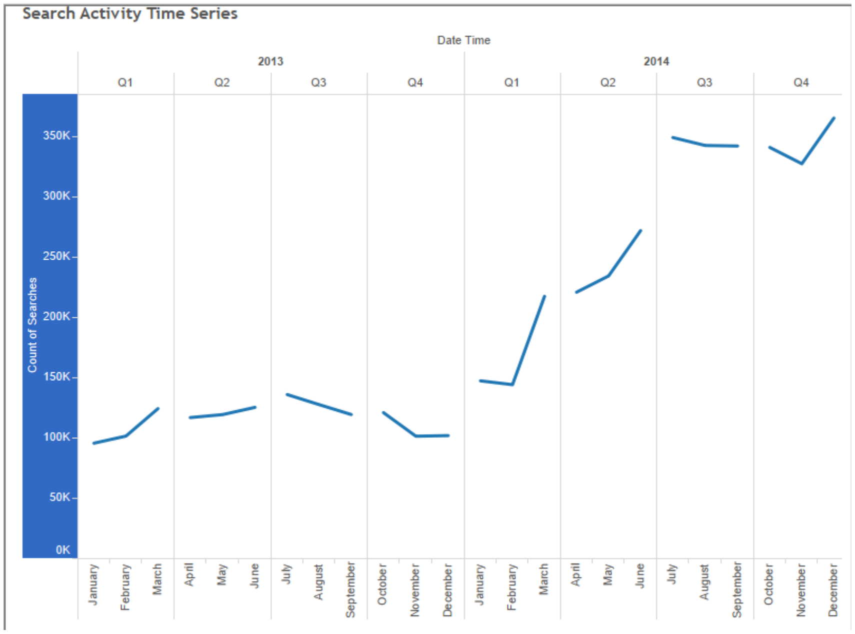
Data Analysis by Visualizations

Tools: Tableau, Power BI

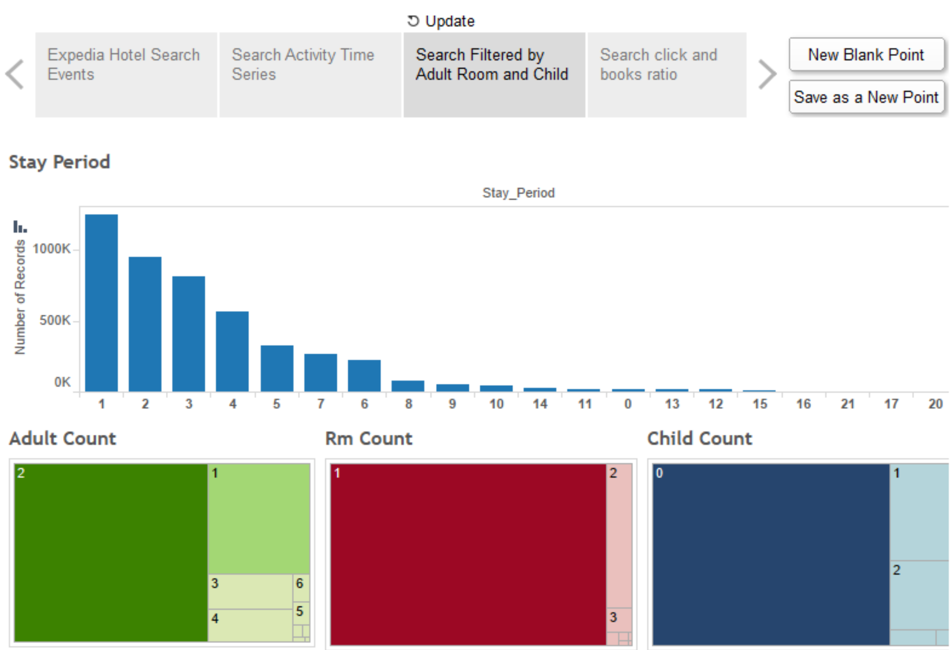
For the purpose of understanding and making sense if the Data we have done visualizations using Tableau and Power BI. In Tableau, we covers the following user stories.

- Exploratory Data Analysis
- User Analysis
- Hotel Analysis

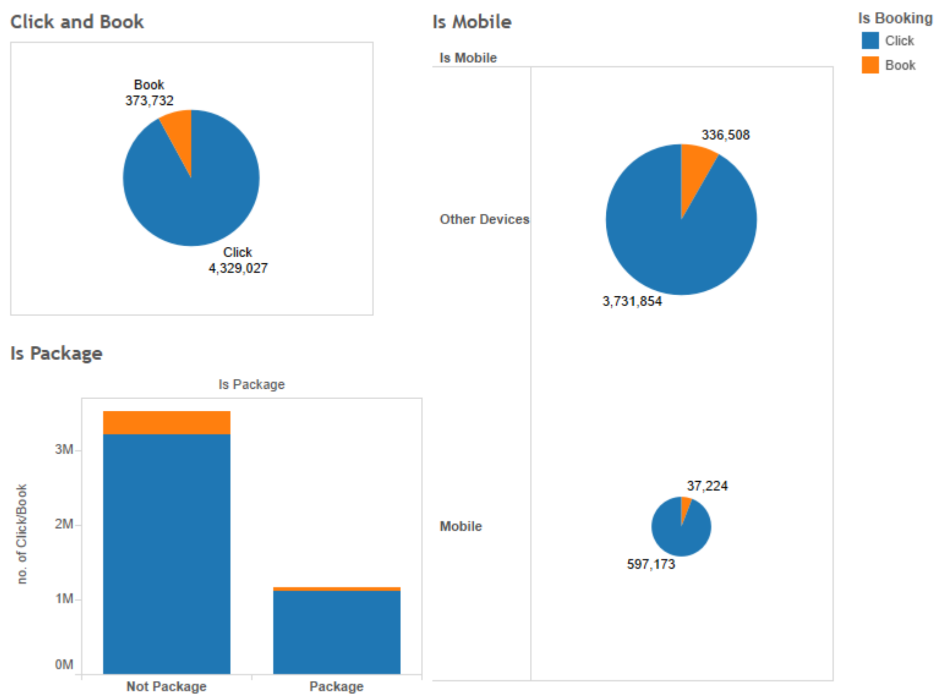




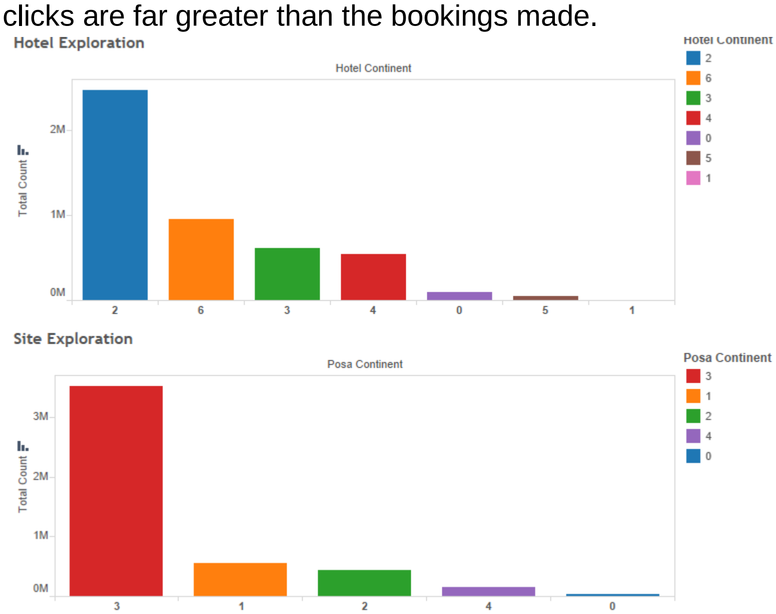
This above time series is for the overall user searches over the years 2013-14. It shows how the



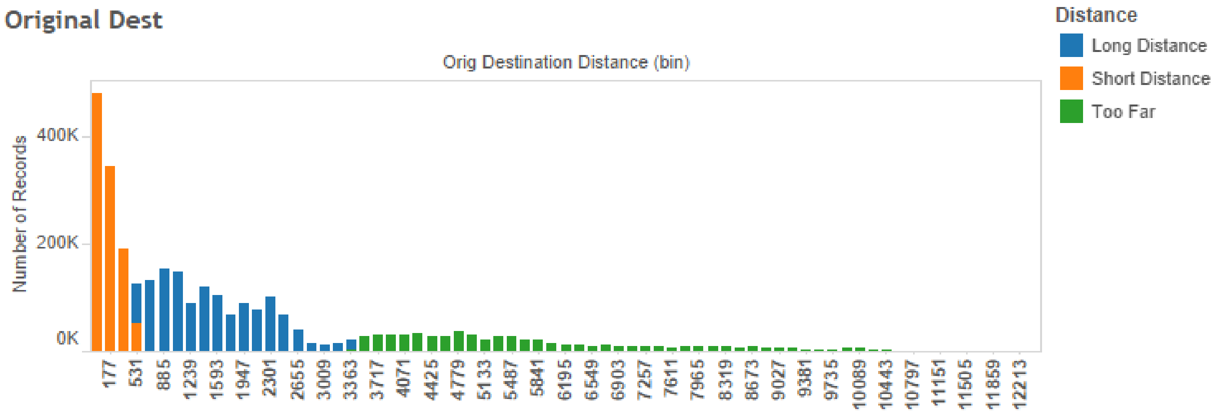
overall searches increased for the year 2014 and mostly toward the Q3 and Q4 period.
The above dashboard shows maximum no of searches according to no. of Rooms, Adult Count and Children Count. It also shows what is the most probable period of stay the user would enter(Stay Period is calculated using the check-in and check-out days the user entered while searching)
On the above analysis we find out that most users commonly recorded searches are for:
Adult count: 2 or 1 adult/adults
Room count: 1 or 2 rooms
Child count: 0 and Max no. of stay is 1-2 day/days

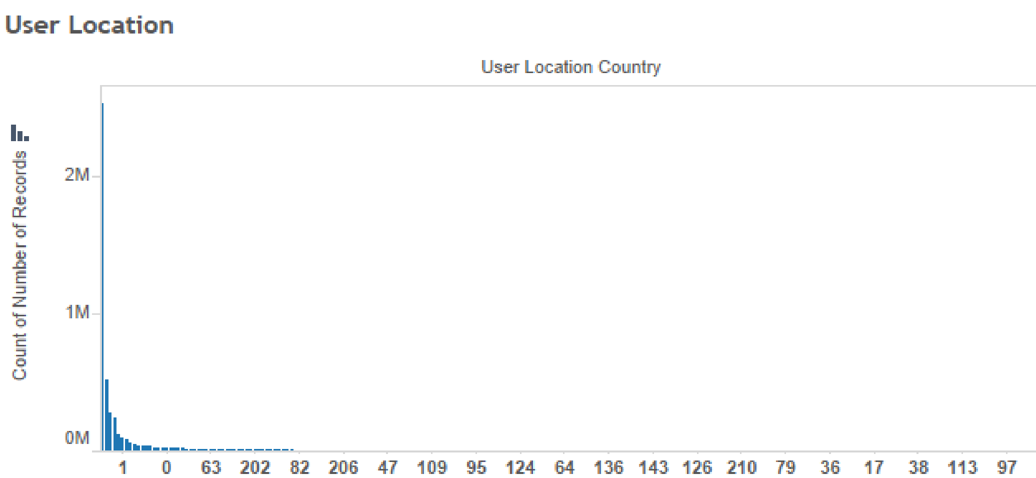


The above dashboard shows the click versus the book searches, which means which searches were made to actually book a hotel versus which were used to just browse through the hotel. Users may use Mobile vs other devices for their searches. Also the users may look for packaged deals as opposed to just hotel reservation. The above dashboard describes exactly how many users checked out packaged deals and how many actually booked it. Also the overall clicks are far greater than the bookings made.

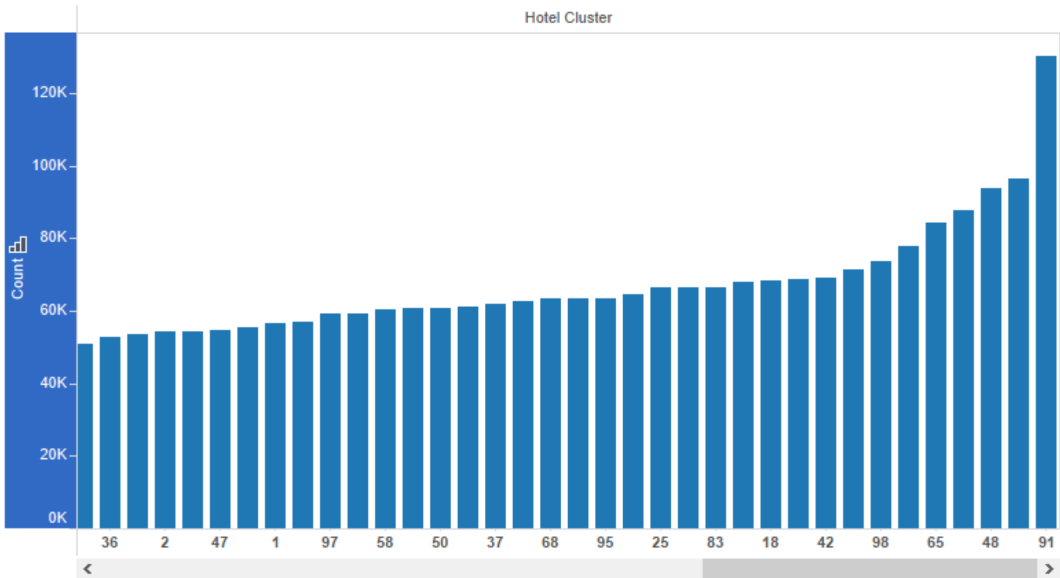


The above dashboard shows no. of records in each continent. The Hotel continent stand for the searches made for hotel in those continents. Posa Continent stands for continents for which there are different site names of Expedia via which the search has been done.



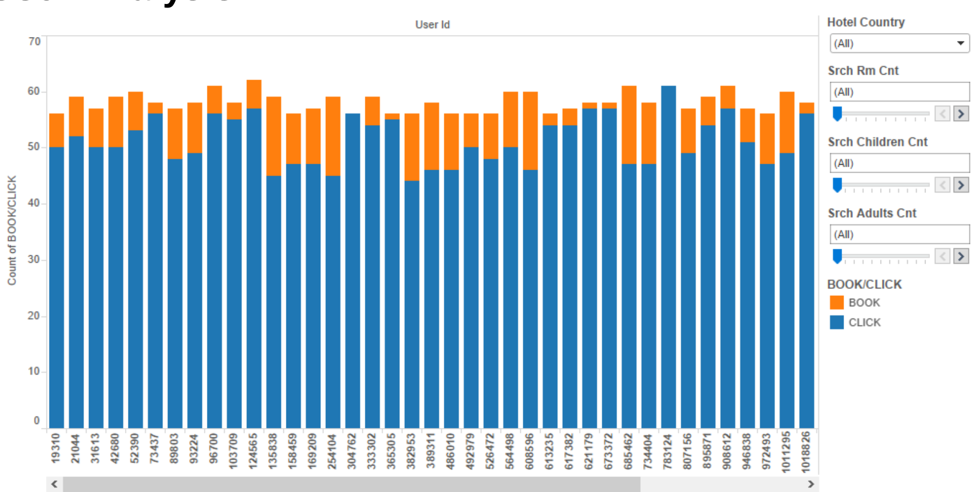


This dashboard shows the distance of the user from the searched destination. We have categorized it to 4 categories for simplicity purpose: Short Distance, Long Distance, Too far and not computable. We can see maximum no of searches by that particular user made as per the distance type and also in the second dashboard we can see which countries have maximum no. of users.

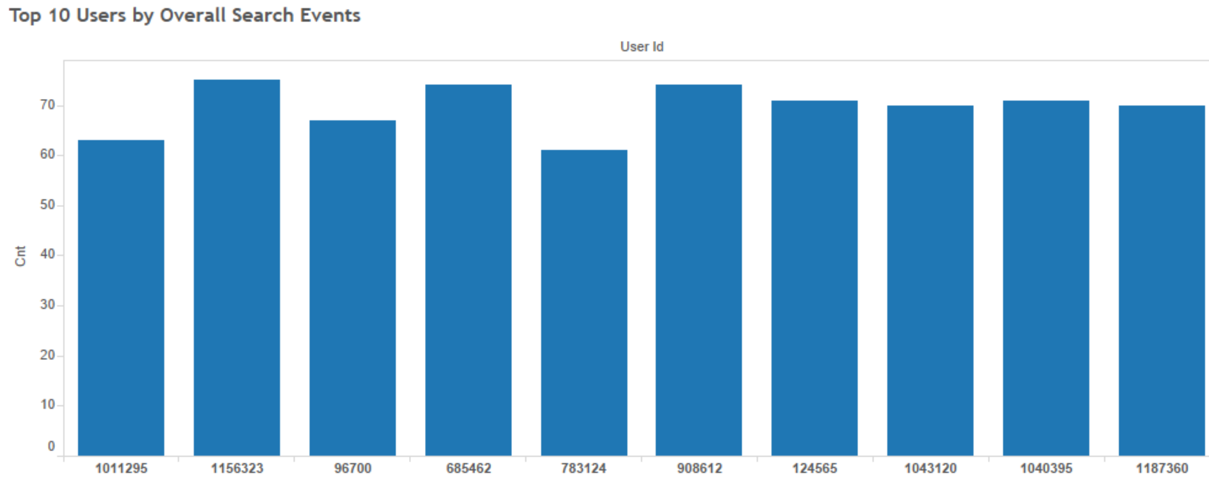


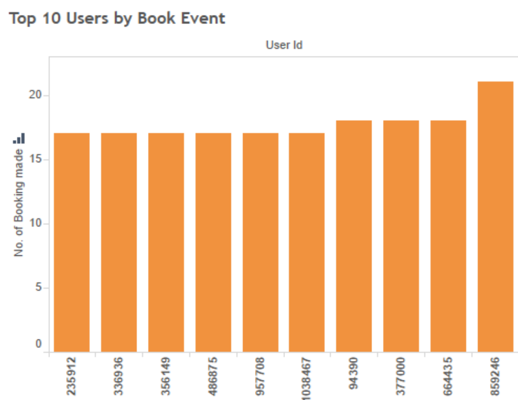
The dashboard above shows the hotel clusters by no. of searches for each.

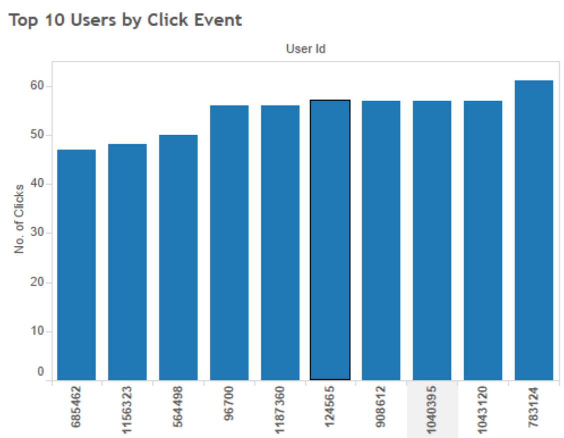
User Analysis



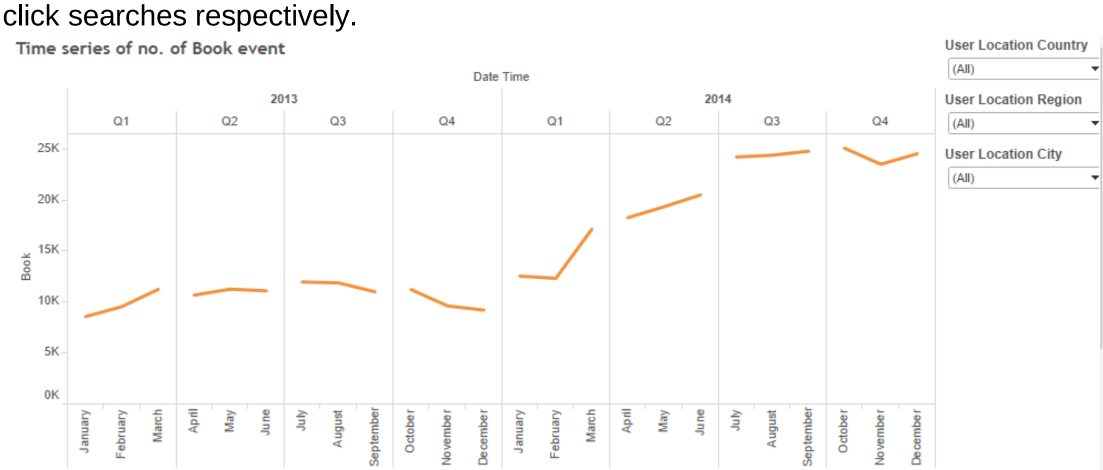
The above dashboard shows the Users activity (click/book) by various filter like the no. of Rooms they selected, the child count and the adult count.



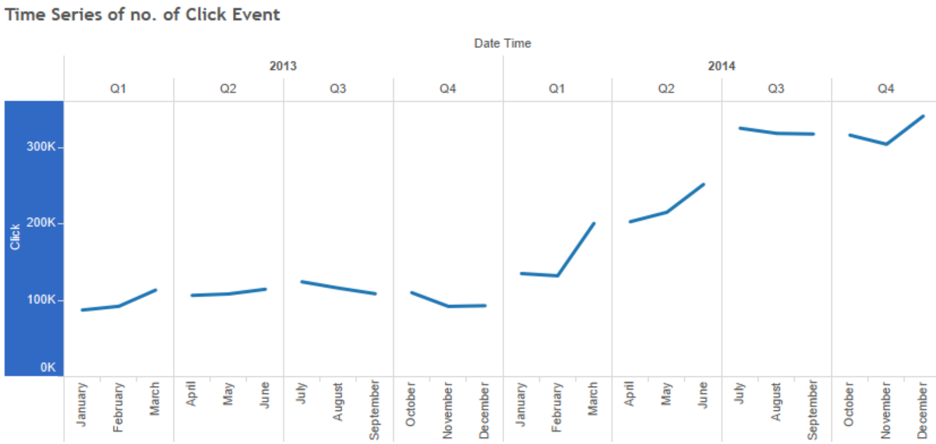




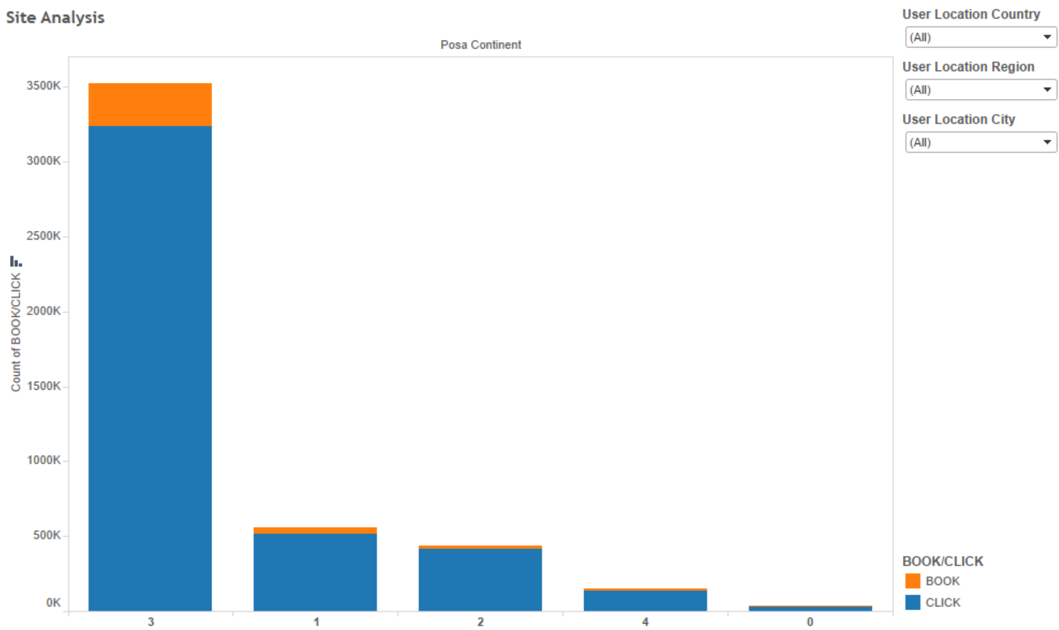
The above 3 dashboards show the top 10 users by overall searches, booking searches and click searches respectively.



The above dashboard shows the time series for the booking event and also you can drill it by users location.



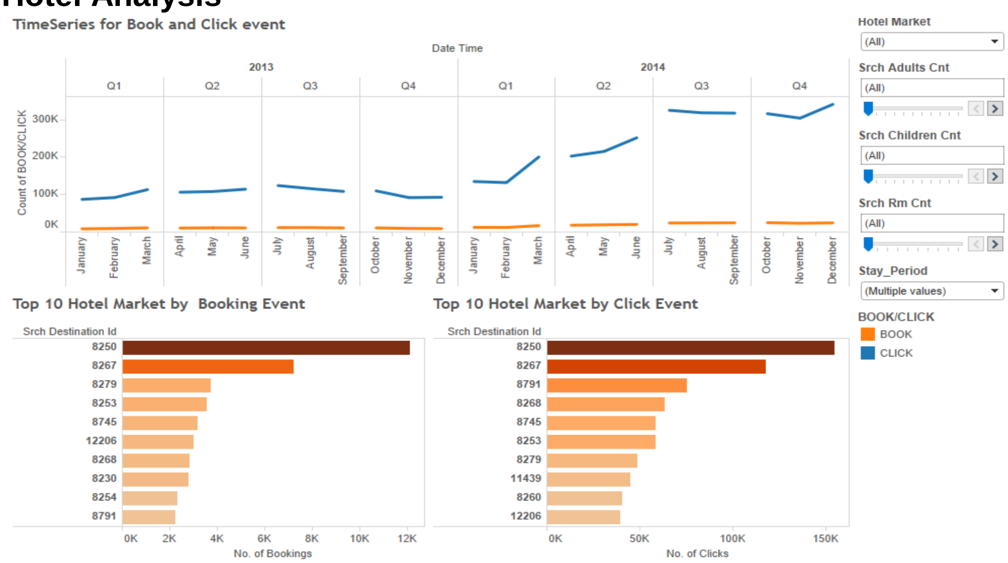
The above dashboard shows the time series for the click event and also you can drill it by users location.



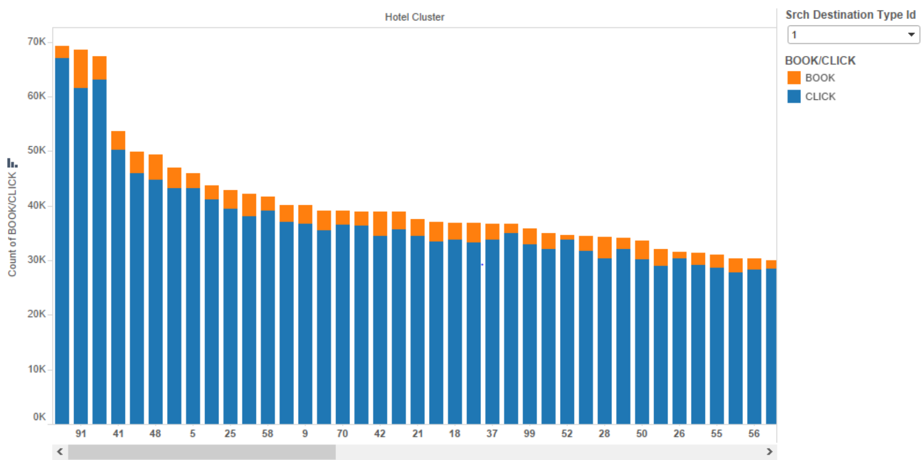
The above dashboard shows which site continent has highest searches and this can be drilled down to see which sites have highest searches and also you can filter it by the users location.

The above dashboard shows the distribution of search event by package deals or mobile usage by filtering them on basis of the site, user and user location.

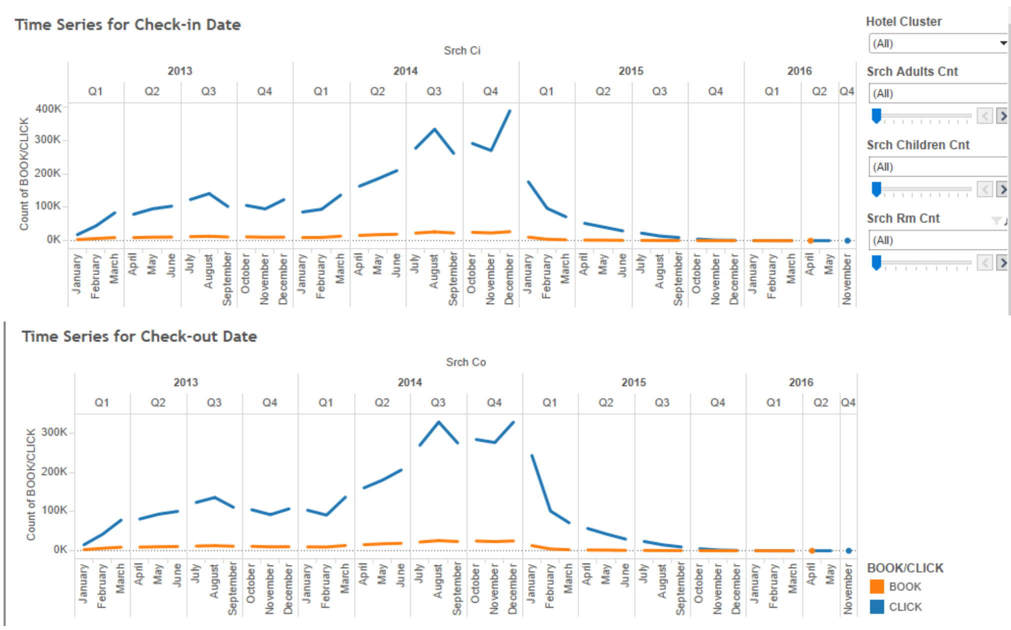
Hotel Analysis



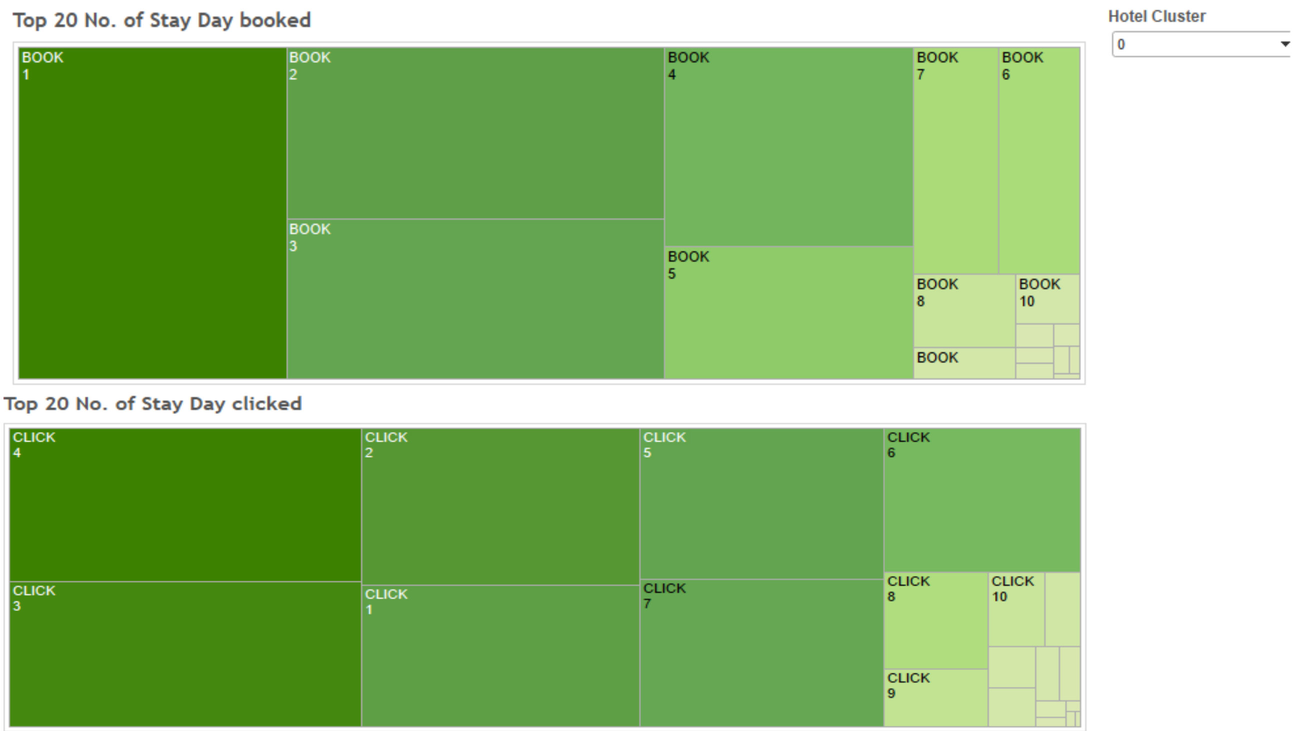
The above dashboard show the time series for event filtered by Hotel market and the various count other parameters. Also it shows the top 10 destination by book and click event which can be filtered by the filters shown.



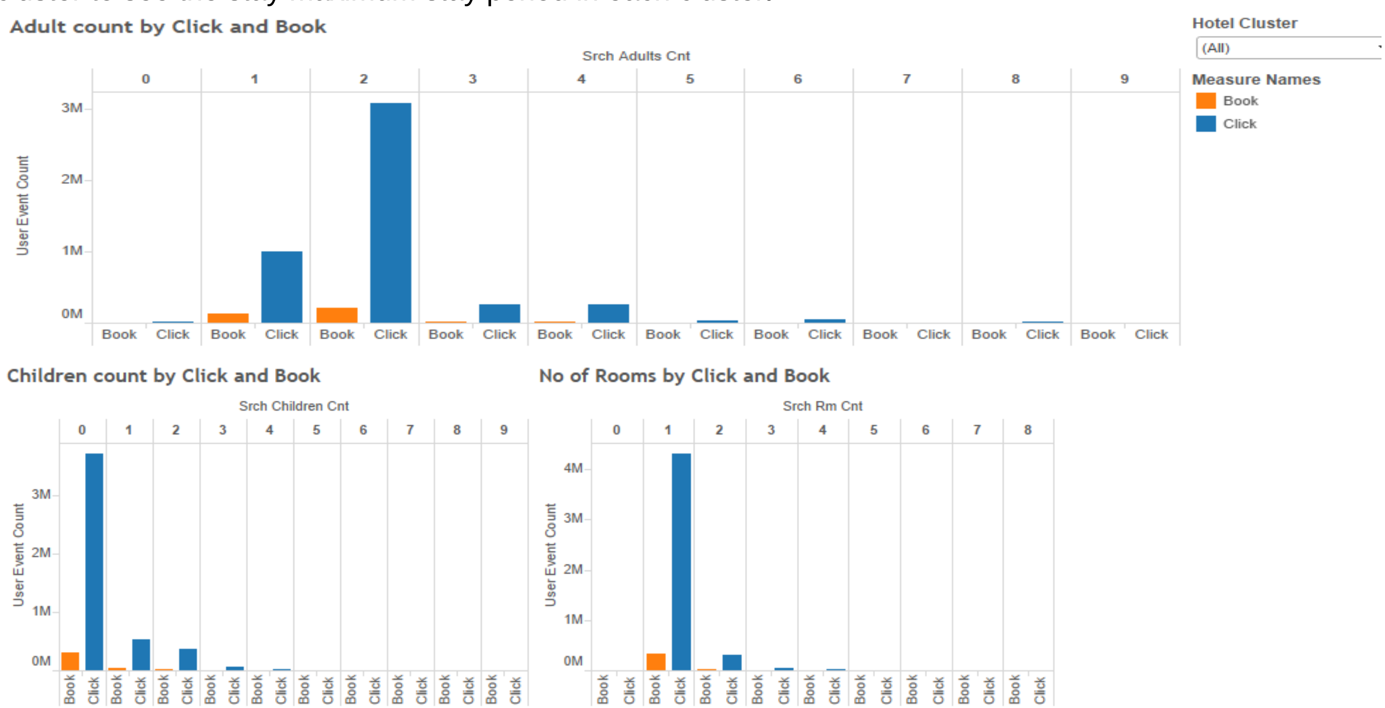
This shows the most popular Hotel Clusters by search event and can be filtered by the Destination Type Id.



The above dashboard shows the Time Series of the Check-in and Checkout dates. We can see that the trend is similar for both. We can filter to see what kind of event it was with respect to parameter like child count, adult count, room count and hotel cluster.

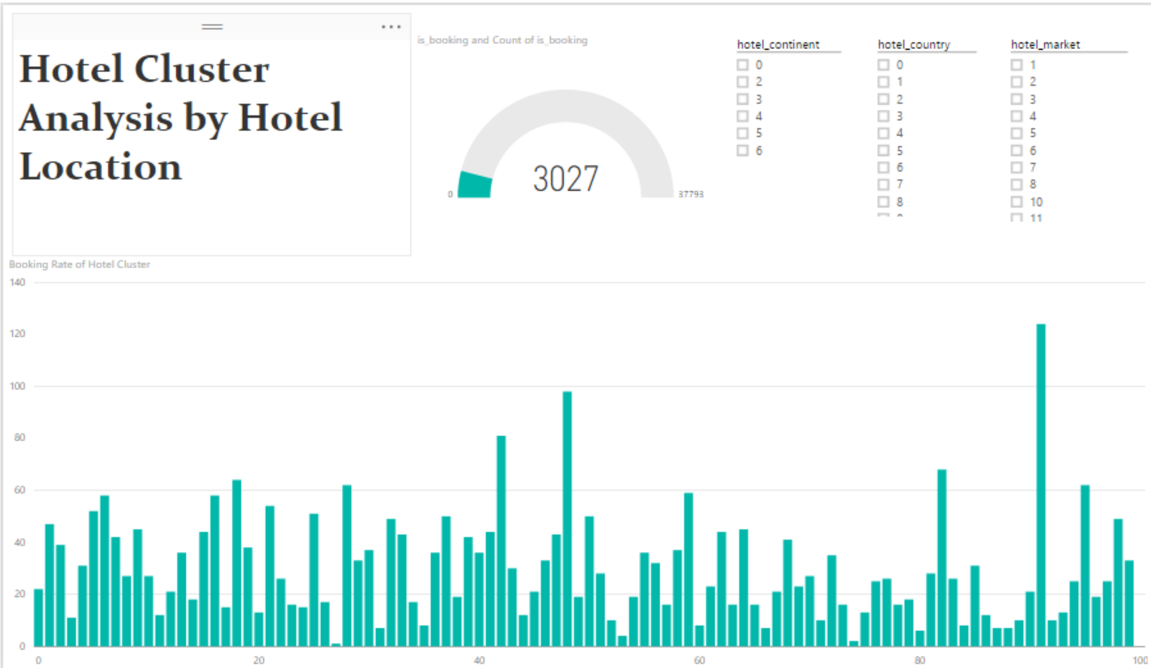


The dashboard above shows the no. of days stayed by event type which can be filtered by the cluster to see the stay maximum stay period in each cluster.



The dashboard above shows adult count, child count and room count by hotel cluster.

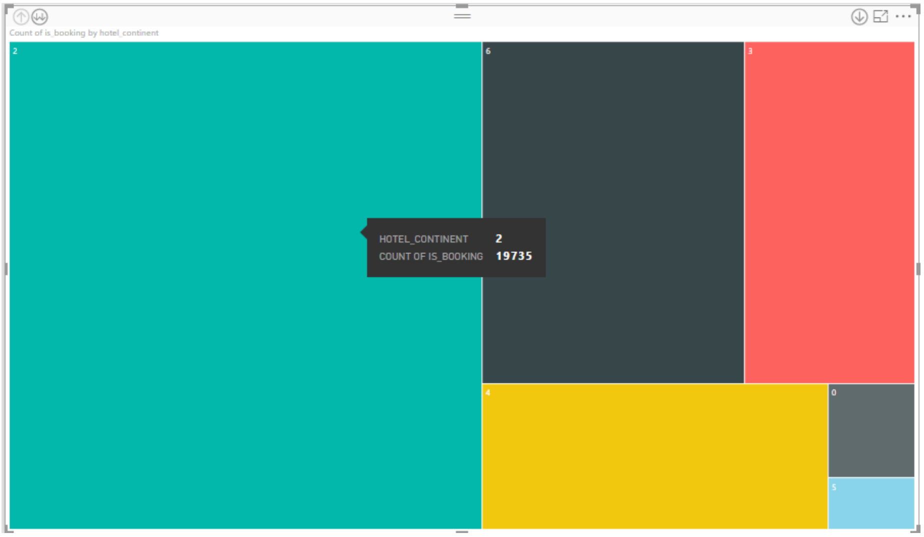
Power BI Dashboards



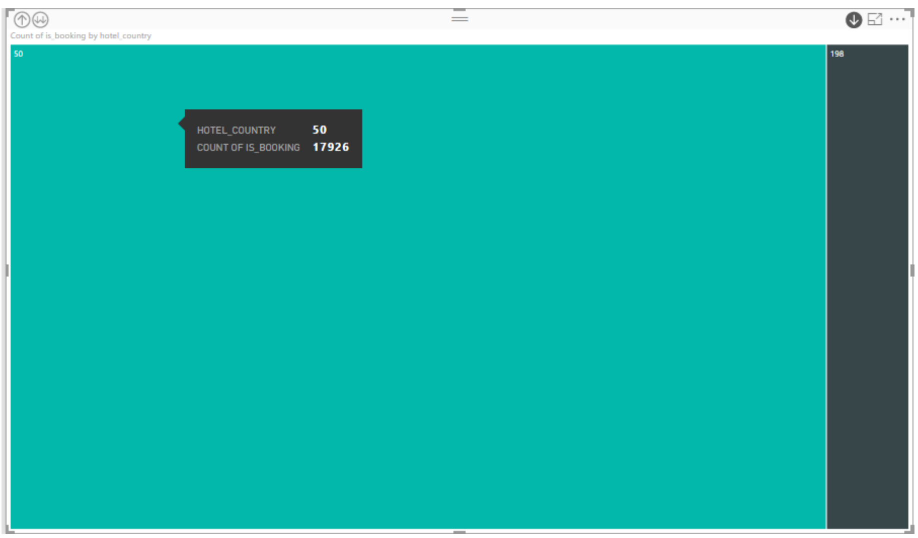
The above dashboard shows which continent/country/market have how many bookings for the hotl cluster in them.

The dashboard above helps to visualize the time series for each cluster by filtering it by cluster no.

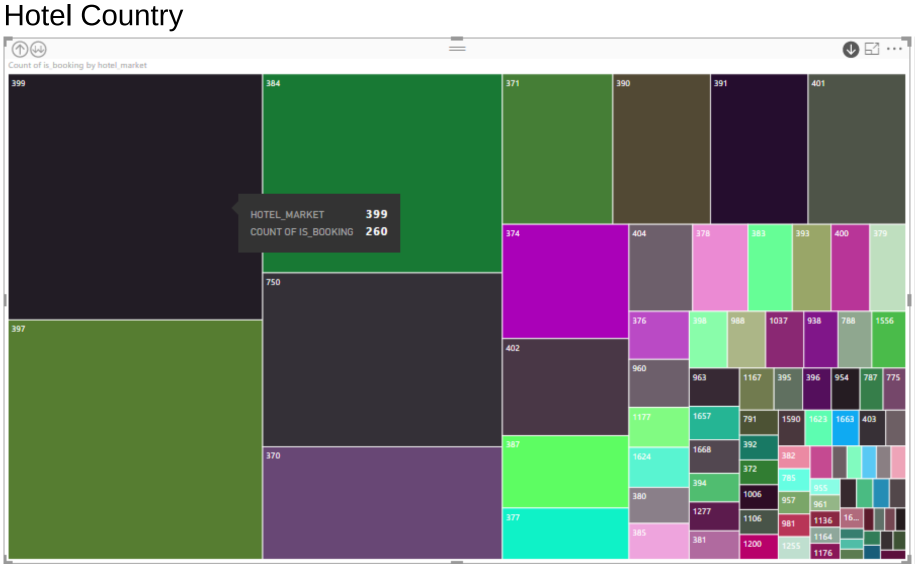
Hotel Hierarchy drilled by



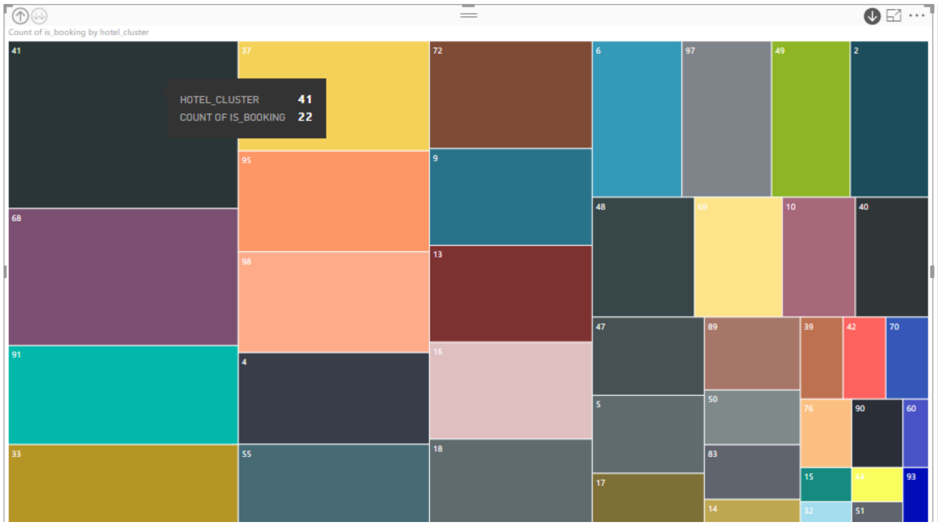
Hotel Continent



Hotel Country



Hotel Market



Hotel Cluster

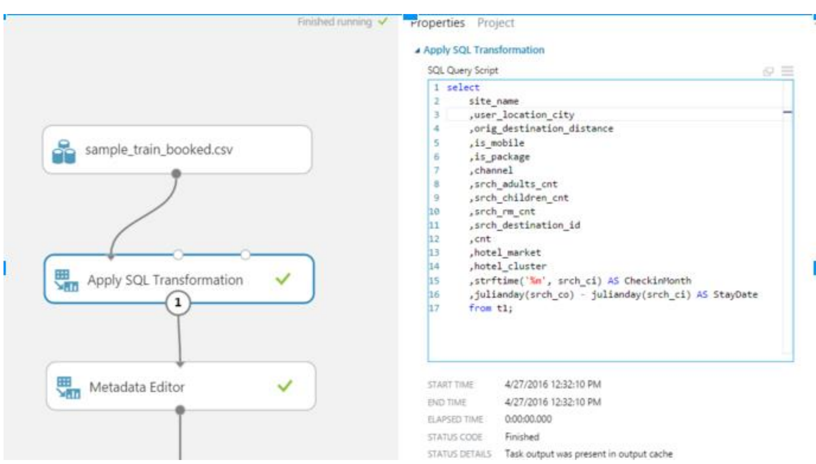
The above tree map is representation if the Hotel Hierarchy, it shows which clusters belong to what market which market belongs to which country and finally which country belongs to which continent.

Modeling

Tools: Microsoft Azure ML Studio, Python, Alteryx

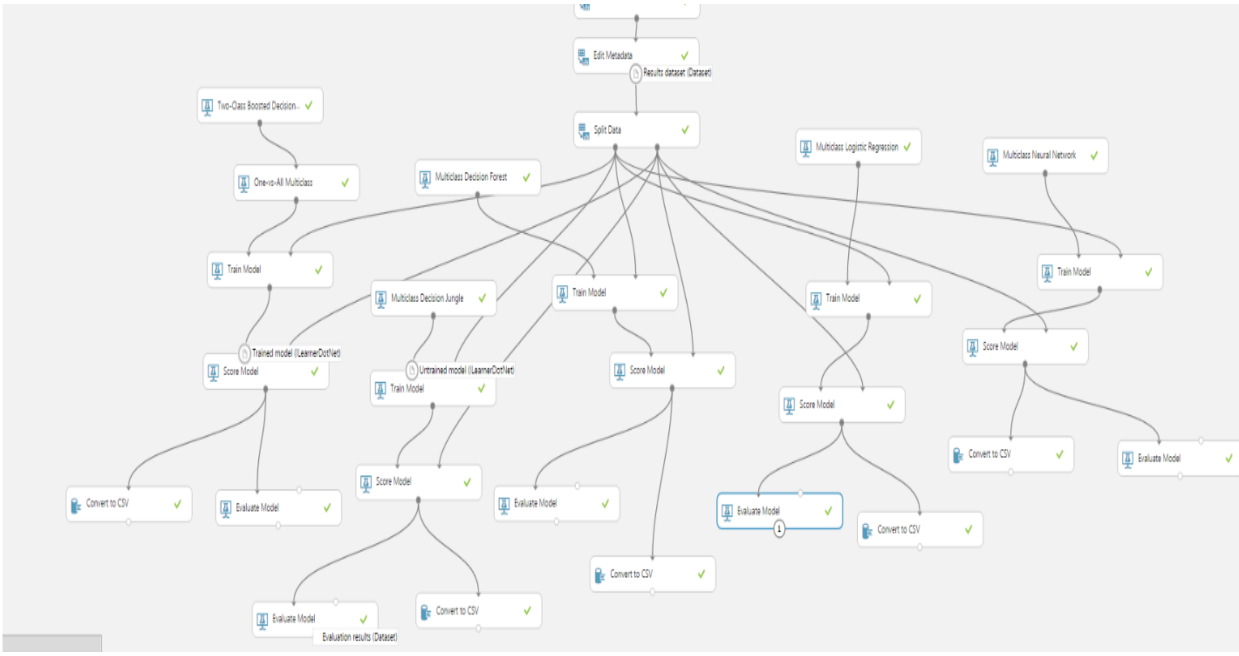
Multiple tables has been uploaded into Microsoft Azure ML Studio, for modeling and data blending purposes.

By using the widget "Apply SQL Transformation", variables has been selected using a sample dataset randomly selected from the original dataset. Derived column CheckinMonth and StayDay (number of days between check in and check out dates).



By using Metadata Editor, data has been casted into the right format. Categorical values also have been generated.

With the options available in Azure ML Studio, we applied 5 different models and compared their performance.



Model evaluation matrix are listed below from left to right the same order as they are in the graph. Multiclass Logistic regression has the highest accuracy rate.

Experiment created on 4/25/2016 › Evaluate Model ›

Metrics

Overall accuracy	0.072203
Average accuracy	0.981444
Micro-averaged precision	0.072203
Macro-averaged precision	0.071599
Micro-averaged recall	0.072203
Macro-averaged recall	0.066227

Metrics

Overall accuracy	0.08098
Average accuracy	0.98162
Micro-averaged precision	0.08098
Macro-averaged precision	NaN
Micro-averaged recall	0.08098
Macro-averaged recall	0.046765

Metrics

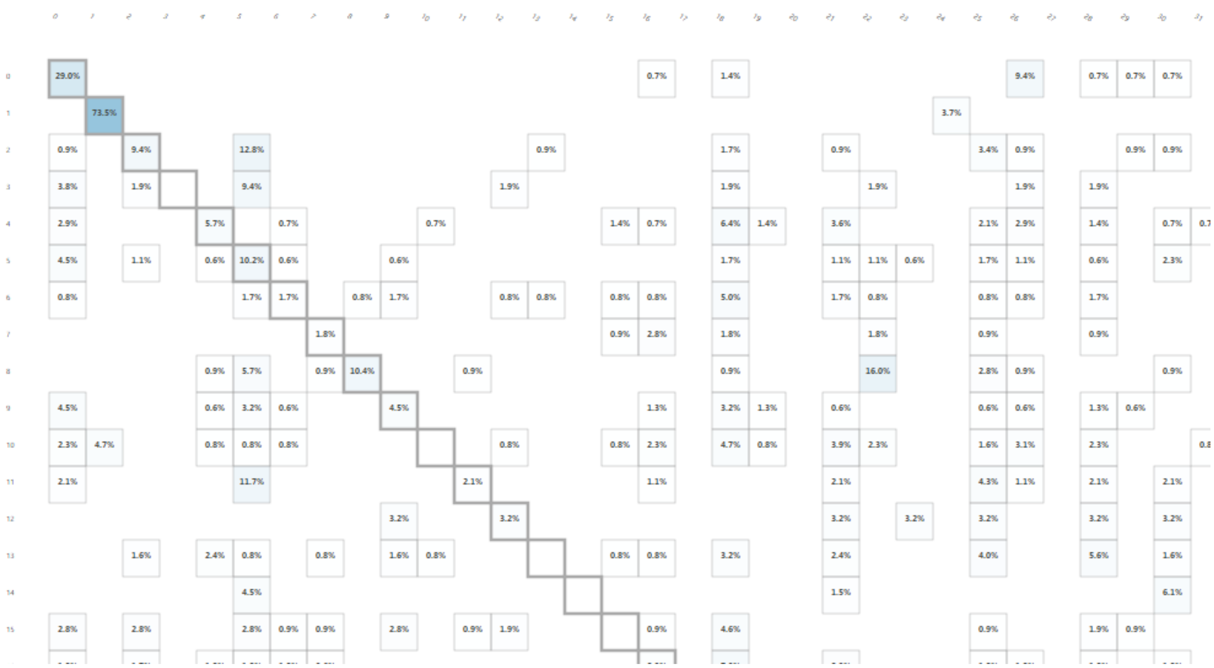
Overall accuracy	0.096547
Average accuracy	0.981931
Micro-averaged precision	0.096547
Macro-averaged precision	NaN
Micro-averaged recall	0.096547
Macro-averaged recall	0.067938

Metrics

Overall accuracy	0.10963
Average accuracy	0.982193
Micro-averaged precision	0.10963
Macro-averaged precision	NaN
Micro-averaged recall	0.10963
Macro-averaged recall	0.09204

Metrics

Overall accuracy	0.090006
Average accuracy	0.9818
Micro-averaged precision	0.090006
Macro-averaged precision	0.090509
Micro-averaged recall	0.090006
Macro-averaged recall	0.094228



According to the data definition, hotel_clusters act as labels for each search_destination, and one search_destination may falls into multiple hotel_clusters. And each hotel_cluster represents certain features of the search_destinations in its group. Due to limited information from the training dataset and special business scenario, we performed a second model evaluation by taking the top five hotel_clusters with highest probabilities in the classification models.

First, we downloaded the scored dataset and generated the top five cluster labels for each record using Python. Second, we loaded all five dataset into Alteryx and created the new evaluation matrix. According to our result, Multiclass Logistic Regression had the highest accuracy rate - 31.86%.

Based on our modeling results, we deployed a web service with Logistic Regression model.

```
In [19]: import numpy as np
         import pandas as pd
         def getClassNumber(i):
             if len(i) == 35:
                 return i[-3:-1]
             else:
                 return i[-2:-1]
             return d
```

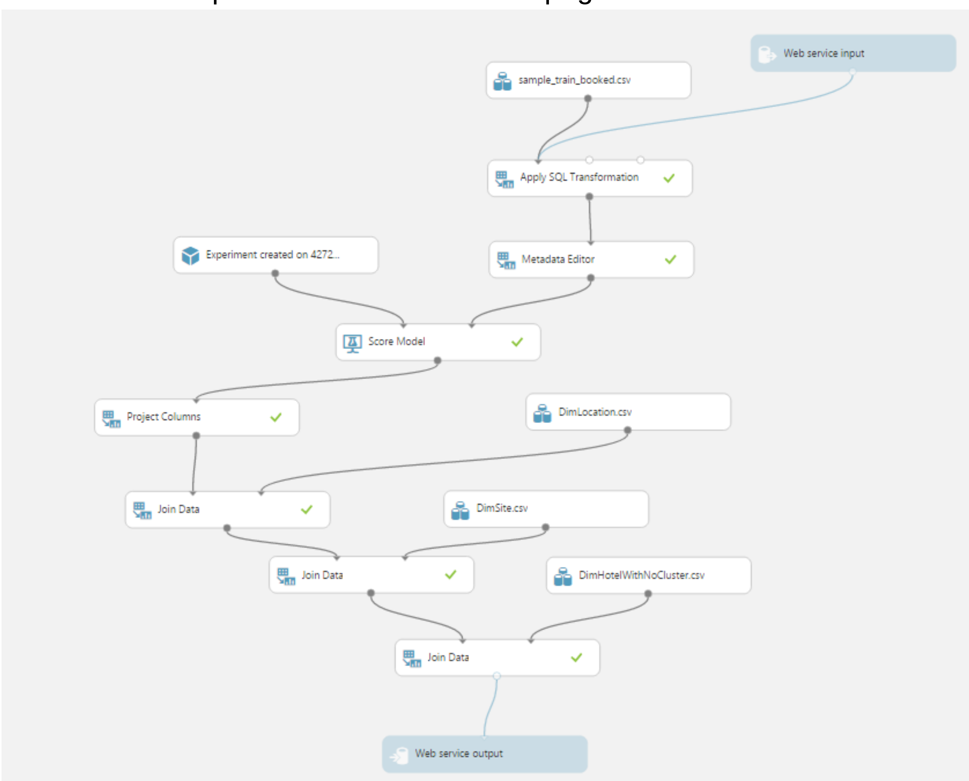
```
In [3]: i = 0
        dataframe1 = pd.read_csv("C:/Users/vanwu/Desktop/INFO 7390 ADS/Expedia data set/Score/DJ.csv", header=0, sep=',')

        for index, row in dataframe1.iterrows():
            i1 = row[18:117].argmax()
            row[i1] = 0
            i2 = row[18:117].argmax()
            row[i2] = 0
            i3 = row[18:117].argmax()
            row[i3] = 0
            i4 = row[18:117].argmax()
            row[i4] = 0
            i5 = row[18:117].argmax()
            dataframe1.set_value(i, 'Scored Labels2', getClassNumber(i2))
            dataframe1.set_value(i, 'Scored Labels3', getClassNumber(i3))
            dataframe1.set_value(i, 'Scored Labels4', getClassNumber(i4))
            dataframe1.set_value(i, 'Scored Labels5', getClassNumber(i5))
            i = i +1

        print dataframe1.head(5)
        dataframe1.to_csv(path_or_buf='C:/Users/vanwu/Desktop/DJ Score.csv', sep=',', na_rep='', header=True)

   site_name  user_location_city  orig_destination_distance  user_id  \
0          2                5938                   456.2658   295241
1         13               26167                 11343.1001    34267
2          2               19207                  1265.1455   566945
3          2               48862                   539.6487   514953
4          2               45689                  2935.2511   346209
```

7 of 7 Fields Cell Viewer 5 records displayed, 1996 bytes

Record #	Sum_Accuracy	Count	R J Accuracy Rate	R T Accuracy Rate	L R Accuracy Rate	Boost Accuracy Rate	NN Accuracy Rate
1	2572	12077	0.212967	[Null]	[Null]	[Null]	[Null]
2	2631	12077	[Null]	0.217852	[Null]	[Null]	[Null]
3	3848	12077	[Null]	[Null]	0.318622	[Null]	[Null]
4	2877	12077	[Null]	[Null]	[Null]	0.238221	[Null]
5	3415	12077	[Null]	[Null]	[Null]	[Null]	0.282769

Deployment

After deploying our model as a web service, we modified our pipeline by joining our scored output table with three dimensions. By doing this, we will be able to retrieve complete information and present them on our web pages.



Tools: Python server, Flask framework, REST API

In this step we deployed the web service and integrate it in front-end. We captured input data from the user and passed into the web service.

We called the API in front-end for the result using Javascript Ajax function.

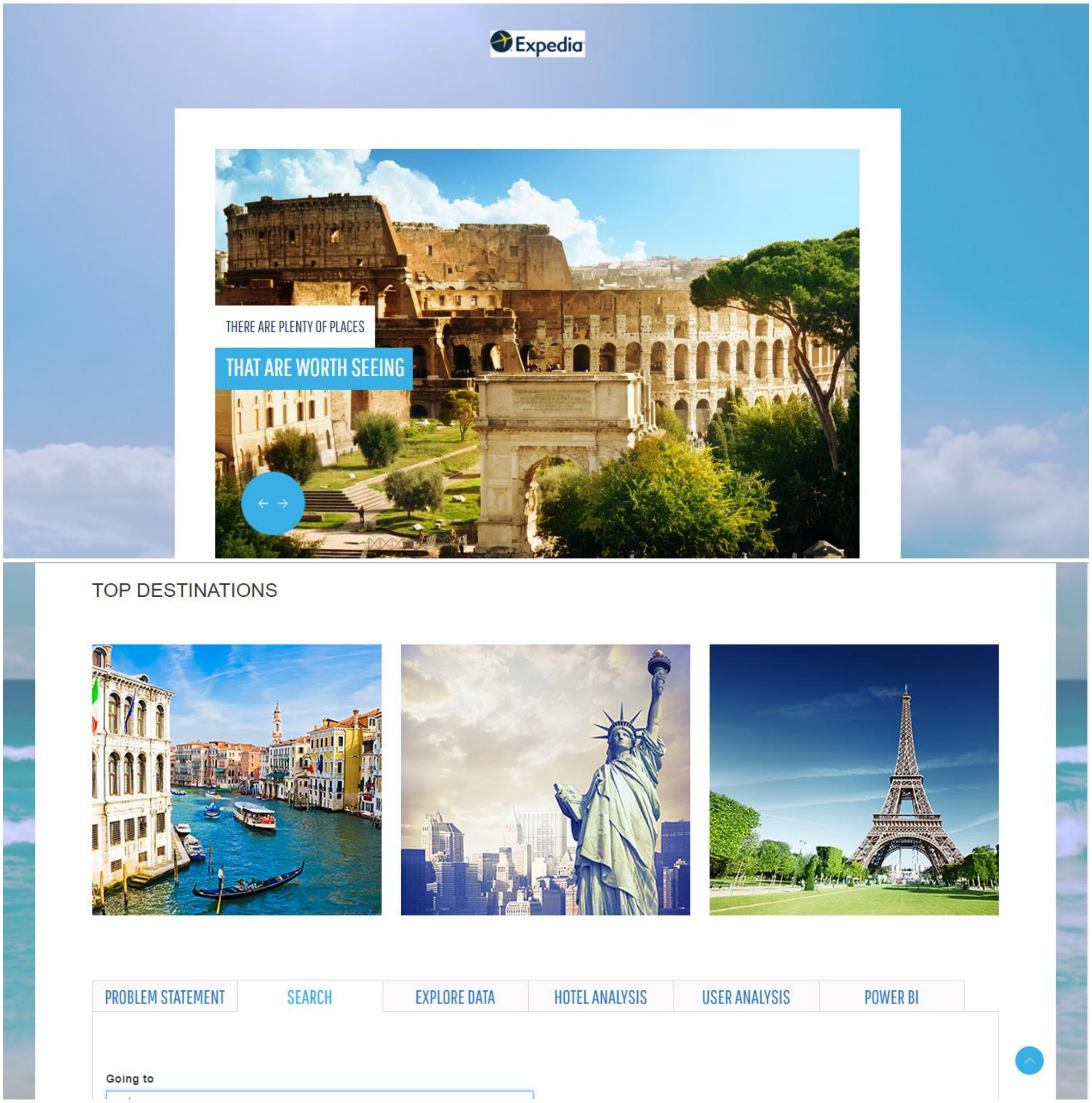
We used Flask Framework for redirecting to the pages.

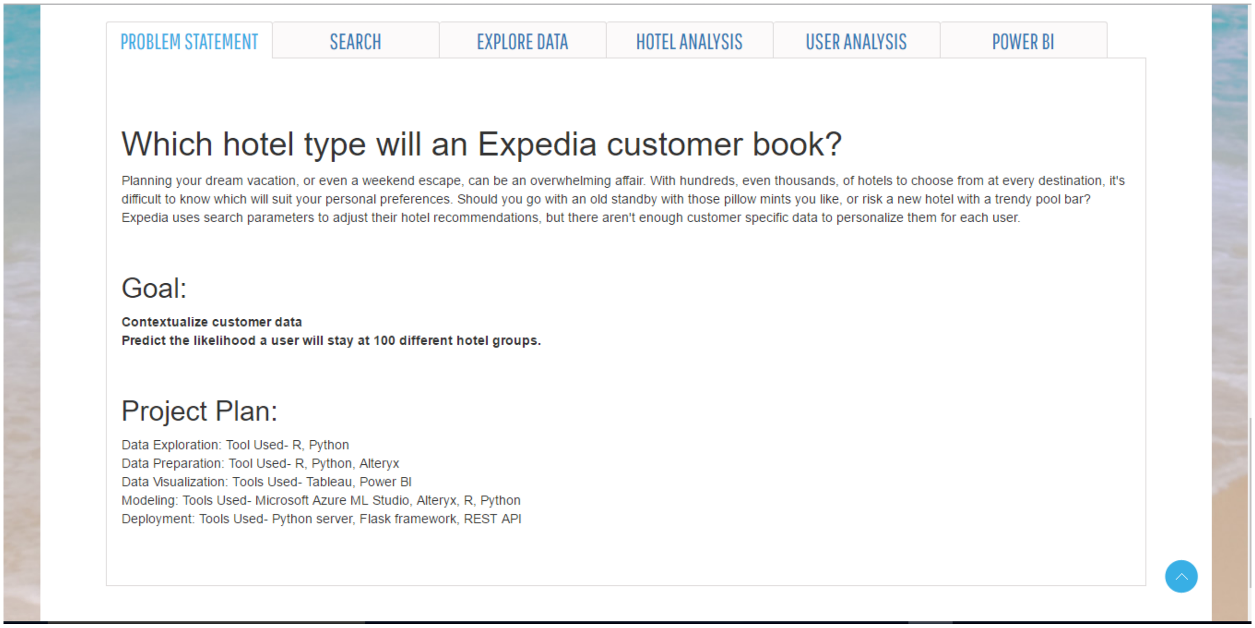
The website includes all the visualizations and brief description of the project

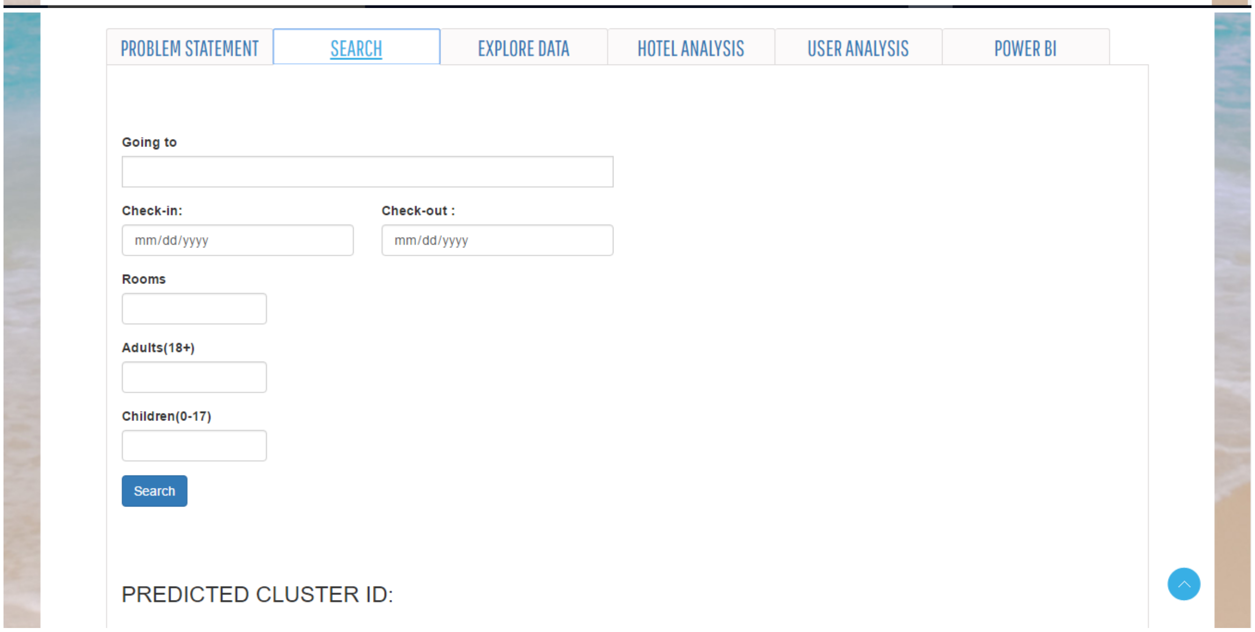
The Web Service input includes the Search Destination ID, Check-in Date, Check-out Date, Number of Adults while travelling, Number of Children and Number of Rooms searched

The output includes the Predicted Hotel Cluster ID and its related information like User Location Country, Hotel Continent, City, etc.

Front-End:
Technologies: Bootstrap, JQuery, Javascript, JQuery-UI and various javascript libraries

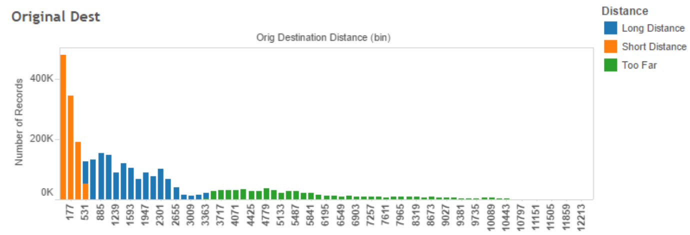



PROBLEM STATEMENT
SEARCH
EXPLORE DATA
HOTEL ANALYSIS
USER ANALYSIS
POWER BI
Which hotel type will an Expedia customer book?
Planning your dream vacation, or even a weekend escape, can be an overwhelming affair. With hundreds, even thousands, of hotels to choose from at every destination, it's difficult to know which will suit your personal preferences. Should you go with an old standby with those pillow mints you like, or risk a new hotel with a trendy pool bar? Expedia uses search parameters to adjust their hotel recommendations, but there aren't enough customer specific data to personalize them for each user.
Goal:
Contextualize customer data
Predict the likelihood a user will stay at 100 different hotel groups.
Project Plan:
Data Exploration: Tool Used- R, Python
Data Preparation: Tool Used- R, Python, Alteryx
Data Visualization: Tools Used- Tableau, Power BI
Modeling: Tools Used- Microsoft Azure ML Studio, Alteryx, R, Python
Deployment: Tools Used- Python server, Flask framework, REST API


PROBLEM STATEMENT
SEARCH
EXPLORE DATA
HOTEL ANALYSIS
USER ANALYSIS
POWER BI
Going to
Check-in:
mm/dd/yyyy
Check-out :
mm/dd/yyyy
Rooms
Adults(18+)
Children(0-17)
Search
PREDICTED CLUSTER ID:

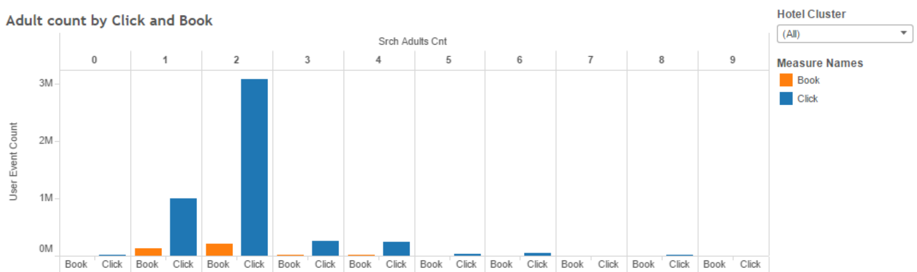
Exploratory Data Analysis

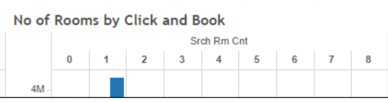
Expedia Hotel Search Events | Search Activity Time Series | Search Filtered by Adult Room and Child | Search click and books ratio | Hotel Continent and Site Continent | Users Location and Distance between hotel and user | Ho



Hotel Analysis

Hotel Market Analysis | Popular Hotel Clusters by Destination Type | Time Series | Highest No. of Bookings/ Clicks for No. of Days Stayed | Analysis by Count of Adult Child and room





```python
from flask import Flask, render_template, request, jsonify
import urllib2
# If you are using Python 3+, import urllib instead of urllib2
import json
app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")


@app.route("/ml", methods=['POST'])
def ml():
    paramVal1 = request.form["param1"]
    paramVal2 = request.form["param2"]
    paramVal3 = request.form["param3"]
    paramVal4 = request.form["param4"]
    paramVal5 = request.form["param5"]
    paramVal6 = request.form["param6"]


    data =  {

        "Inputs": {

                "input1":
                {
                    "ColumnNames": ["date_time", "site_name", "posa_continent", "user_location_country", "user_location_region", "user_location_city", "orig
                    "Values": [ [ "", "2", "0", "0", "0", "37449", "0", "0", "0", "0", "0", paramVal1, paramVal2, paramVal3, paramVal4, paramVal5, paramVal6
                },        },
            "GlobalParameters": {
```

```python
    }

    body = str.encode(json.dumps(data))

    print(body)

    url = 'https://ussouthcentral.services.azureml.net/workspaces/dcd42eb6e203459181e33854f8a7a4f7/services/3e8f52e17e524c20973e732b1b2139c5/execute?api-versi
    api_key = 'QkRUT7WOn1qyxUaycXQ1YunxQnZ6YqAmiTwFIU5JykpY8MDCPgIMF/c7/SC8lHl7R3wJJhHMm1aODHiha/XvsA==' # Replace this with the API key for the web service
    headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}

    req = urllib2.Request(url, body, headers)

    try:
        response = urllib2.urlopen(req)

        # If you are using Python 3+, replace urllib2 with urllib.request in the above code:
        # req = urllib.request.Request(url, body, headers)
        # response = urllib.request.urlopen(req)

        result = response.read()
        print(result)
        return result
    except urllib2.HTTPError, error:
        print("The request failed with status code: " + str(error.code))

    # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
        print(error.info())

        print(json.loads(error.read()))
        return render_template("index.html")
```