Expedia Hotel Recommendation

Python final project
Team 5:

- Aniket W
- Mohil J
- Taha T
- Vijay T
- Yamini R

Overview

❖ Introduction:

- Business Problem and Importance of Solving the problem
- Goals of the project
- Implications of the project

❖ Dataset:

- Describing the dataset
- Data Preprocessing
- Facts & Visualisation
- Fitting the Model

❖ Conclusion

❖ Annexure (Snippet of code for reference)

Introduction (1/2)

Business problem & What are we trying to solve:

- Planning a vacation can be overwhelming
- Hundreds of hotels/destination to choose
- Idea of vacation is leisure and thus providing personalized hotel recommendations can be ease to the user.
- Providing a better user experience, increasing customer satisfaction, and improving the overall revenue of the hotel industry.

Goals:

Primary Goal:

- Build a hotel recommendation system based on users their search, booking history and preferences
- Help selecting hotels that match their preferences, budget, and location

Secondary Goal:

- Evaluate the performance of multiple classification algorithms
- Select the best-performing model based on evaluation metrics and cross-validation results.

Introduction (2/2)

Implication of the project:

Customer:

- Reduction of time spent on deciding.
- Ease & Convenience

Hotel Industry:

- Improved customer satisfaction
- Increased revenue
- Efficient marketing
- Enhanced user experience

Dataset (1/8)

Describing the dataset:

- Data taken from Kaggle : https://www.kaggle.com/competitions/expedia-hotel-recom
- No of Rows – 37670293 & No of Columns – 24
- Variables:
  - Input variables : User location, device used, package, channel of booking etc.
  - Output variable : Hotel_cluster

Data Preprocessing

- Filtered random 1L records for preprocessing and customer who did the bookings.
- Converted Date into months and year
- Checked the data for missing values.

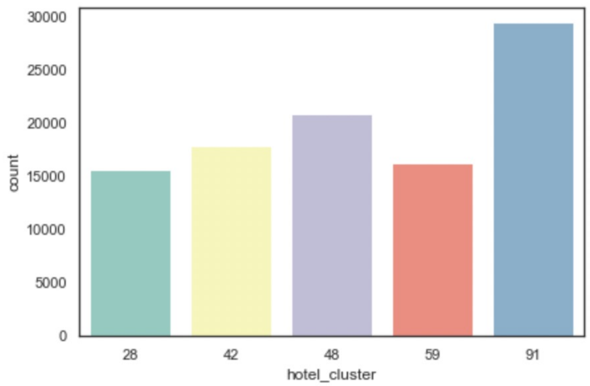
Variable treatment

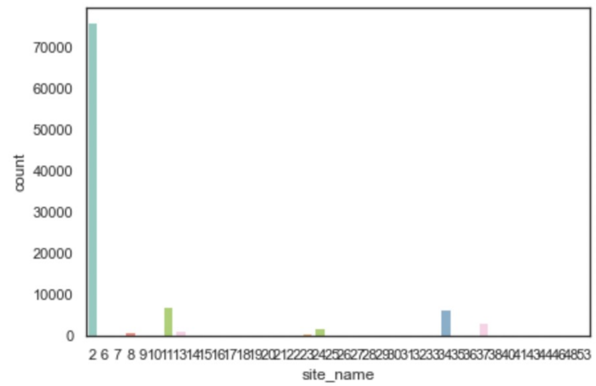
- Using existing variables created new variable **"No of days before booking" and "Stay duration"**
- Checked Zero variance and high cardinality in variables
- Imputed missing data using Median.
- Data was scaled
- Dropped variables which had high cardinality, zero variance and no correlation.
- After performing all pre-processing steps data was split into train and test (70:30)

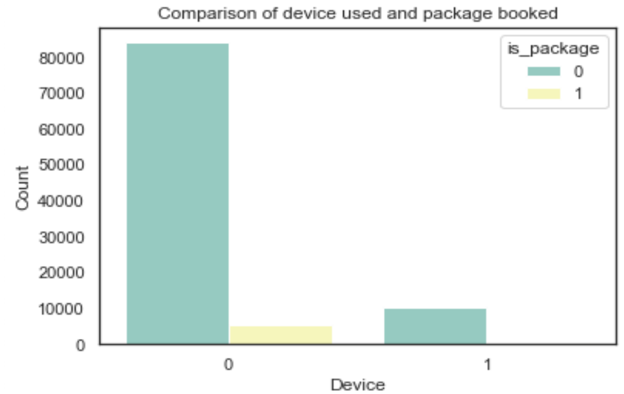
Interesting Insights (2/8)

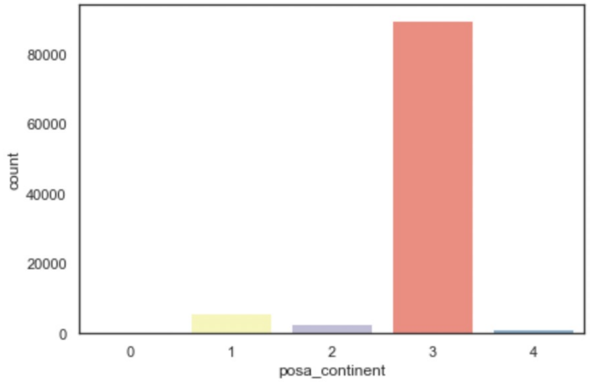
- The top 5 hotel clusters are 91,48, 42, 59, 28
- 81% booking is done using site_name ”2” and 94% from posa_continent “3”
- 90% of customers who made the booking are from country “66” and “205” and most frequent Hotel_cluster booked in country “50”
- 90% customers make bookings using “Device other than mobile” and 95% take the entire Holiday package
- 52% and 36% customers booking room for 2 adults and 1 adult respectively and 80% travel without kids.
- Channel “9” was the most effective marketing channel.
- People booking for 1 day is 56%, 2 days is 22% and 3 days is 11%.
- Graphs supporting the above are in next slides

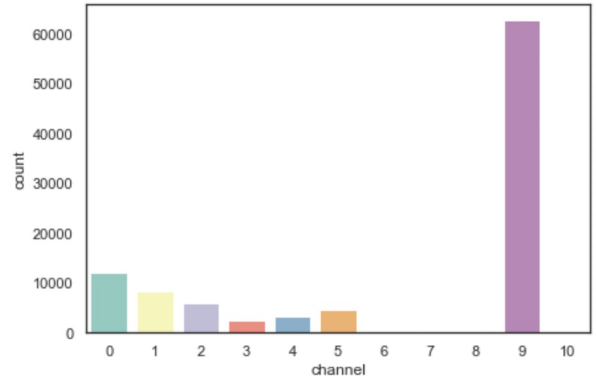
Visualisation (3/8)

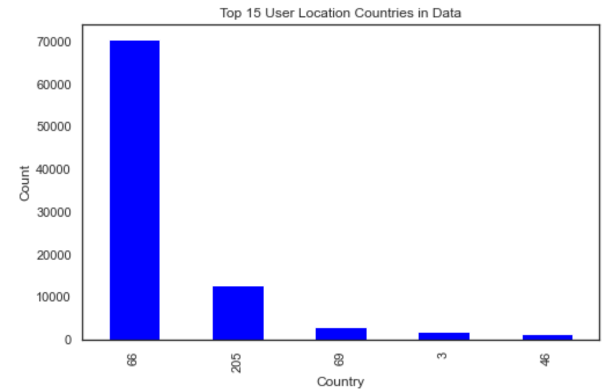




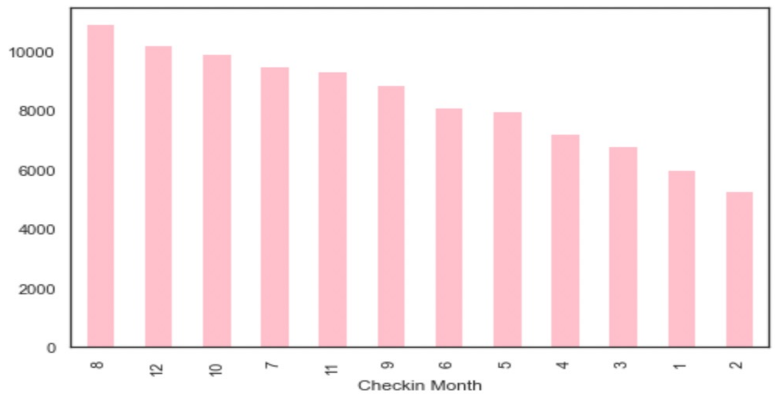


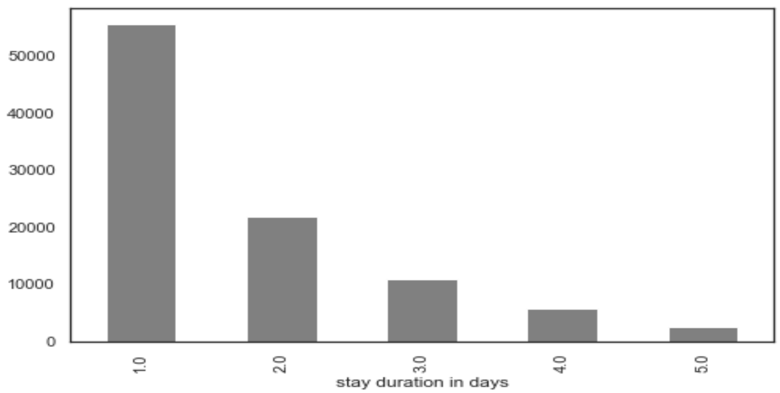


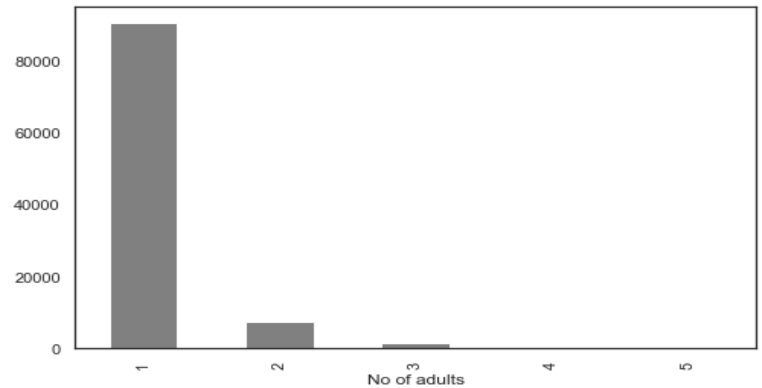




Visualisation (4/8)







Fitting the Model (5/8)

Steps for selection of Data & Variables:

- Different combination were used to identify the best fitting variables, with scaling and without scaling of the data
- Train and Test data was also split basis booking month and year i.e. 2013 and till Aug 2014 as train and beyond 2014 as test
- Performed Principal component analysis for dimension reduction.
- Total no of Hotel_cluster in the dataset was 30+ however using all of them did not give a good model for prediction.
- Thus, for better prediction we filtered Top 10 / Top 5 clusters and built a model on the same.
- The variables has no correlation and thus we did not use Linear Regression.
- Models used for prediction : K-NN, Random Forest, Decision Tree and XG Boost.
- Snippet of accuracy of various combination has been added in annexures

Fitting the Model – PCA (6/8)

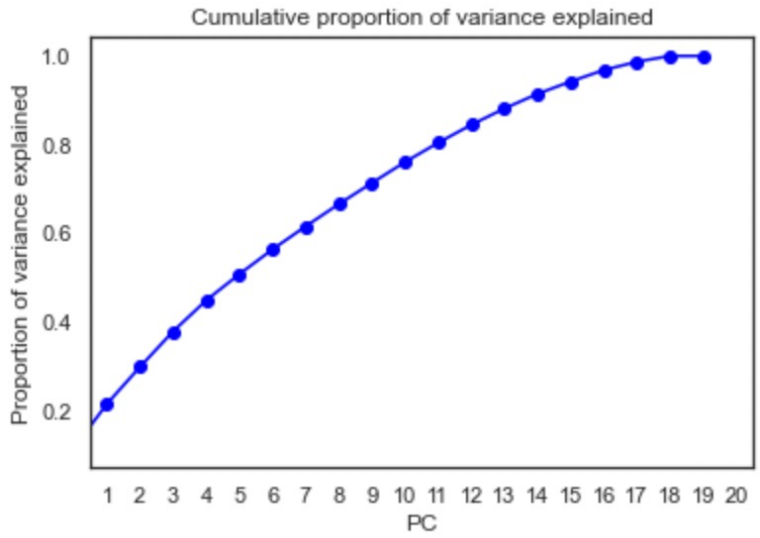
Perform PCA

In [141]:
```
from sklearn.decomposition import PCA
```

In [142]:
```
# create a PCA object
pca_train = PCA()

# fit the PCA object to the data and transform it
pc_train = pca_train.fit_transform(X_train)
```

In [144]:
```
print(pca_train.explained_variance_ratio_)
```

```
[1.14917642e-01 1.00104088e-01 8.36414010e-02 7.99920961e-02
 7.01202897e-02 5.90191278e-02 5.57967090e-02 5.20435617e-02
 5.00912789e-02 4.78026653e-02 4.72353329e-02 4.33604928e-02
 4.05180647e-02 3.67320311e-02 3.30931765e-02 2.77876733e-02
 2.59178895e-02 1.87573912e-02 1.30689208e-02 1.66932339e-07]
```



- Performed PCA for dimension reduction
- 90% variation was explained by variables till 15
- Performance of model without PCA and scaling of data for Top 5 clusters performed better than PCA

Fitting the Model – Algorithms (7/8)

```
#Training

rf = RandomForestClassifier(n_estimators=80)
rf.fit(X_train, Y_train)

#Prediction

rf_train_prediction = rf.predict(X_train)
rf_test_prediction = rf.predict(X_test)

#Accuracy

train_accuracy = accuracy_score(Y_train,rf_train_prediction)
test_accuracy = accuracy_score(Y_test,rf_test_prediction)
rf_accuracy = test_accuracy

#Print

print("Train Accuracy: %.2f%%" % (train_accuracy * 100.0))
print("Test Accuracy: %.2f%%" % (test_accuracy * 100.0))
```

```
Train Accuracy: 99.98%
Test Accuracy: 42.14%
```

```
#Training

knn = neighbors.KNeighborsClassifier()
knn.fit(X_train,Y_train)

#Prediction

knn_train_prediction = knn.predict(X_train)
knn_test_prediction = knn.predict(X_test)

#Accuracy

train_accuracy=accuracy_score(Y_train,knn_train_prediction)
test_accuracy=accuracy_score(Y_test,knn_test_prediction)
knn_accuracy = test_accuracy

#Print

print("Train Accuracy: %.2f%%" % (train_accuracy * 100.0))
print("Test Accuracy: %.2f%%" % (test_accuracy * 100.0))
```

```
Train Accuracy: 52.61%
Test Accuracy: 29.39%
```

Fitting the Model - Algorithms (8/8)

```python
# Training
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train, Y_train)

# Prediction
xgb_train_prediction = xgb_model.predict(X_train)
xgb_test_prediction = xgb_model.predict(X_test)

# Accuracy
train_accuracy = accuracy_score(Y_train, xgb_train_prediction)
test_accuracy = accuracy_score(Y_test, xgb_test_prediction)
xgb_accuracy = test_accuracy

# Printing
print("Train Accuracy: %.2f%%" % (train_accuracy * 100.0))
print("Test Accuracy: %.2f%%" % (test_accuracy * 100.0))
```

```
Train Accuracy: 57.44%
Test Accuracy: 47.36%
```

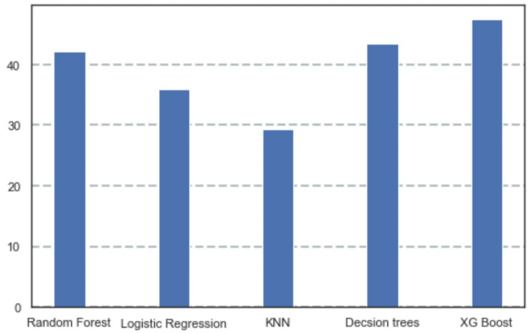
```python
#Training

dt = DecisionTreeClassifier()
dt.fit(X_train, Y_train)

#Prediction

dt_train_prediction = dt.predict(X_train)
dt_test_prediction = dt.predict(X_test)

#Accuracy

train_accuracy = accuracy_score(Y_train, dt_train_prediction)
test_accuracy = accuracy_score(Y_test, dt_test_prediction)
dt_accuracy = test_accuracy

#Printing

print("Train Accuracy: %.2f%%" % (train_accuracy * 100.0))
print("Test Accuracy: %.2f%%" % (test_accuracy * 100.0))
```

```
Train Accuracy: 99.98%
Test Accuracy: 43.40%
```

Conclusion

- After running algorithms on different combinations of data, the best combination was “Data with Top 5 Hotel_cluster”

- The best algorithm based on accuracy is “XG Boost”



	Model	Accuracy
0	Random Forest	42.1
1	logistic_regression	35.9 *
2	KNN	29.4 *
3	Decision Tree	43.3
4	XGBoost	46.9

*The accuracy for logistic regression and Knn is the lowest on train and test as it is used for binary classification problems however the data used has more than 2 categories)

- Based on the above accuracies, it can be concluded that the current models may not be performing well on the given data.

- It could also be helpful to consider alternative modeling techniques or to collect additional data to improve the accuracy of the models.

Thank You

Data Reference

https://www.kaggle.com/competitions/expedia-hotel-recommendations

Annexure – Snippet of Code

Sr.No	Data	Accuracy			
		RF	LR	Knn	DT
1	W/o Scaling	10.0	3.7	8.3	7.9
2	With Scaling	9.9	6.0	3.3	8.1
3	W/o Scaling, only top 5 clusters	42.2	34.0	42.4	43.1
4	With Scaling, only top 5 clusters	42.6	35.9	29.4	43.3
5	With Scaling, only top 10 clusters	31.8	23.1	17.6	30.6
6	With Scaling, only top 5 clusters & year and monthwise split of train and test	36.6	35.9	28.5	28.4
7	With Scaling, only top 5 clusters + PCA	35.7	35.6	30.6	27.9

Final selection

1.2 Load the dataset and only keeping 1L records, splitting data into train and test 70:30

```
In [2]: full_train = pd.read_csv("/Users/yaminirege/Desktop/MSBA Fall 2022/2.Q2 - Winter/2. MIS 636-901- Python/Project - Final

        full_test = pd.read_csv("/Users/yaminirege/Desktop/MSBA Fall 2022/2.Q2 - Winter/2. MIS 636-901- Python/Project - Final/

In [3]: full_train.shape

Out[3]: (37670293, 24)

In [4]: full_test.shape

Out[4]: (2528243, 22)

In [5]: full_train.columns

Out[5]: Index(['date_time', 'site_name', 'posa_continent', 'user_location_country',
               'user_location_region', 'user_location_city',
               'orig_destination_distance', 'user_id', 'is_mobile', 'is_package',
               'channel', 'srch_ci', 'srch_co', 'srch_adults_cnt', 'srch_children_cnt',
               'srch_rm_cnt', 'srch_destination_id', 'srch_destination_type_id',
               'is_booking', 'cnt', 'hotel_continent', 'hotel_country', 'hotel_market',
               'hotel_cluster'],
              dtype='object')
```

```
In [14]: # Select 1L rows randomly from the original dataset
         random_sample = random_sample0.sample(n=100000, random_state=42)

In [165]: random_sample.head()

Out[165]:
```

	date_time	site_name	posa_continent	user_location_country	user_location_region	user_location_city	orig_destination_distance	user_id	is_mobile	is_package
0	2014-01-26 02:40:37	2	3	63	451	29000	118.5376	656753	0	0
1	2014-10-28 02:11:27	18	2	119	0	27731	NaN	186216	0	0
2	2014-07-23 20:34:39	2	3	66	184	11740	37.4398	399072	0	0
3	2014-09-18 16:15:54	2	3	66	462	48217	1037.6654	522505	1	0
4	2014-06-10 18:01:42	2	3	66	325	32561	206.7990	124909	0	0

5 rows × 32 columns

```
In [166]: random_sample.shape

Out[166]: (100000, 32)
```

Annexure – Snippet of Code

In [164]: `random_sample.describe().T`

Out[164]:

	count	mean	std	min	25%	50%	75%	max
site_name	100000.0	7.169870	10.669735	2.0000	2.0000	2.0000	2.0000	5.300000e+01
posa_continent	100000.0	2.855500	0.540595	0.0000	3.0000	3.0000	3.0000	4.000000e+00
user_location_country	100000.0	87.895890	54.298385	0.0000	66.0000	66.0000	66.0000	2.390000e+02
user_location_region	100000.0	314.379360	166.031814	0.0000	174.0000	321.0000	385.0000	1.021000e+03
user_location_city	100000.0	27876.212930	16564.833483	1.0000	13910.0000	27655.0000	42396.0000	5.650700e+04
orig_destination_distance	76421.0	1009.320472	1473.470374	0.0056	165.2992	436.0174	1217.9191	1.166669e+04
user_id	100000.0	600256.328790	343858.208209	11.0000	305813.0000	600696.0000	895193.2500	1.198760e+06
is_mobile	100000.0	0.105040	0.306606	0.0000	0.0000	0.0000	0.0000	1.000000e+00
is_package	100000.0	0.068930	0.253336	0.0000	0.0000	0.0000	0.0000	1.000000e+00
channel	100000.0	6.332630	3.650208	0.0000	2.0000	9.0000	9.0000	1.000000e+01

2.0 Data Preprocessing

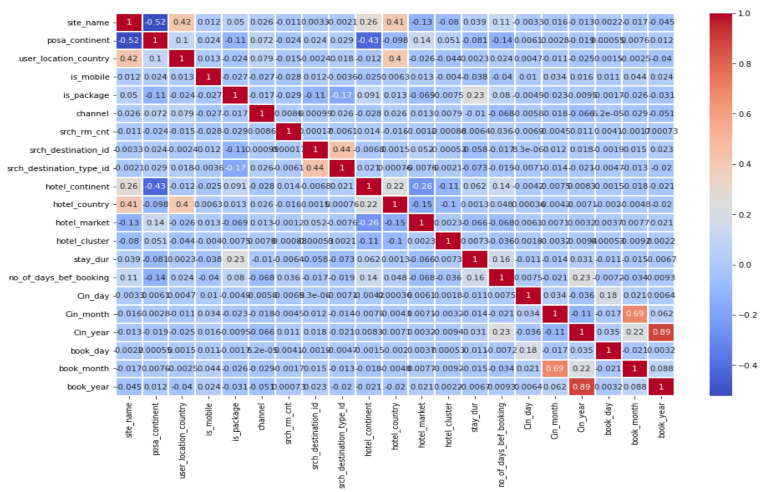
2.1 Creating new variable

stay_dur: number of duration of stay no_of_days_bet_booking: number of days between the booking and Cin_day: Check-in day Cin_month: Check-in month Cin_year: Check-out year

```
# Function to convert date object into relevant attributes
def convert_date_into_days(df):
    random_sample['srch_ci'] = pd.to_datetime(random_sample['srch_ci'])
    random_sample['srch_co'] = pd.to_datetime(random_sample['srch_co'])
    random_sample['date_time'] = pd.to_datetime(random_sample['date_time'])
```

```
## Convert srch_ci and srch_co to datetime objects
random_sample['srch_ci'] = pd.to_datetime(random_sample['srch_ci'], format='%Y-%m-%d')
random_sample['srch_co'] = pd.to_datetime(random_sample['srch_co'], format='%Y-%m-%d')
random_sample['date_time'] = pd.to_datetime(random_sample['date_time'], format='%Y-%m-%d')
```

```
#calculate stay duration
random_sample['stay_dur'] = (random_sample['srch_co'] - random_sample['srch_ci']).astype('timedelta64[D]')
```




```
#create a col names for std values

new_col = [i+'_standarized' for i in numerical_variables1]
```

```
#convert to numpy

array = random_sample2[numerical_variables1].values
```

```
#create standrisation instance

data_scaler = StandardScaler().fit(array)
```

```
#standarised the numerical variables

data_rescaled = pd.DataFrame(data_scaler.transform(array), columns = new_col)
```

```
data_rescaled.head()
```

	site_name_standarized	posa_continent_standarized	user_location_country_standarized	is_mobile_standarized	is_package_standarized	channel_standarized	srcl
0	-0.484538	0.267300	-0.458504	-0.342591	-0.27209	0.730748	
1	1.015038	-1.582524	0.572840	-0.342591	-0.27209	-1.186960	
2	-0.484538	0.267300	-0.403253	-0.342591	-0.27209	-1.186960	
3	-0.484538	0.267300	-0.403253	2.918935	-0.27209	0.730748	
4	-0.484538	0.267300	-0.403253	-0.342591	-0.27209	0.730748	