Hotel Bookings Cancellation

BY

Heba Mohamed Abd-elmonam

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In tourism and travel-related industries, most of the research on Revenue Management demand forecasting and prediction problems employ data from the aviation industry, in the format known as the Passenger Name Record (PNR). This is a format developed by the aviation industry.

The main goal is to generate meaningful estimators from the data set we have and then choose the model that best predicts cancellation by comparing it to the accuracy ratings of several ML models. Following the data science life cycle, I will work towards this aim through several processes. I present a study approach that utilizes a variety of methods and algorithms including (Logistic Regression, Random Forest, K-Folds cross-validation, Decision Tree, and XgBoost) to make a prediction model classify a hotel booking's likelihood to be canceled.

The results show that there is one feature (called reservation status) that has a very high correlation with the target column (is canceled), and a logistic regression classifier gives an accuracy of 99% while the same model gives an accuracy of 81% when this feature is removed. As a result, I choose to delete the feature and conduct a fair comparison of the different classifiers. The XgBoost classifier achieved the highest accuracy of 88%.

Keywords: Hotel cancellation prediction, Classification, logistic regression, Random Forest, K-Fold CV, Grid search, XgBoost.

1 PROBLEM STATEMENT

In tourism and travel-related industries, most of the research on Revenue Management demand forecasting and prediction problems employ data from the aviation industry, in the format known as the Passenger Name Record (PNR). This is a format developed by the aviation industry. However, the remaining tourism and travel industries like hospitality, cruising, theme parks, etc., have different requirements and particularities that cannot be fully explored without industry's specific data. Hence, two hotel datasets with demand data are shared to help in overcoming this limitation [1].

2 PROPOSED SOLUTION

The main goal is to generate meaningful estimators from the data set we have and then choose the model that best predicts cancellation by comparing it to the accuracy ratings of several ML models. Following the data science life cycle, I will work towards this aim through several processes.

By cleaning and analyzing the hotel booking dataset, we will be able to understand how cancellation actions are affected by various factors, and then present a study approach that utilizes a variety of methods and algorithms including (Logistic Regression, Random Forest, K-Folds cross-validation, Decision Tree, and XgBoost) to make a prediction model classify a hotel booking's likelihood to be canceled. Nevertheless, due to the characteristics of the variables included in these datasets, their use goes beyond this cancellation prediction problem.

3 TOOLS

In this work, a several APIs and libraries used listed as follows:

- *Pandas Library*
- *NumPy Library*
- *Plolty Library.*
- *Sklearn Library.*
- *Keras APIs.*

4 DATA SCIENCE LIFE CYCLE

In simple terms, a data science life cycle is nothing but a repetitive set of steps that you need to take to complete and deliver a project/product to your client. before starting any data science project that we have got from either our clients or stakeholder first we need to understand the underlying problem statement presented by them. Once we understand the business problem, we have to gather the relevant data that will help us in solving the use case [3].

The process is fairly simple wherein the company has to first gather data, perform data cleaning, perform EDA to extract relevant features, preparing the data by performing feature engineering and feature scaling. In the second phase, the model is built and deployed after a proper evaluation [3].

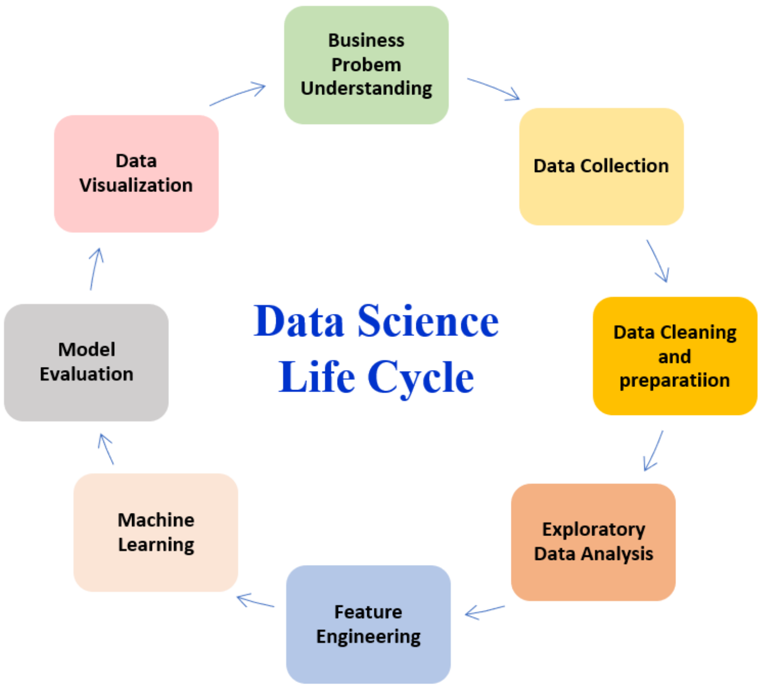


Figure 1: Data science life cycle.

5 DATASET

This dataset contains booking information for a city hotel and a resort hotel and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces, among other things. From the publication sciencedirect [1] we know that:

- *Both hotels are located in Portugal (southern Europe) ("H1 at the resort region of Algarve and H2 at the city of Lisbon"). The distance between these two locations is ca. 280 km by car and both locations border on the north Atlantic.*
- *The 'adr' column stands for (Average Daily Rate) and is calculated by dividing the sum of all lodging transactions by the total number of staying nights*
- *The data contains "bookings due to arrive between the 1st of July of 2015 and the 31st of August 2017".*

Table 1: Dataset variables description.

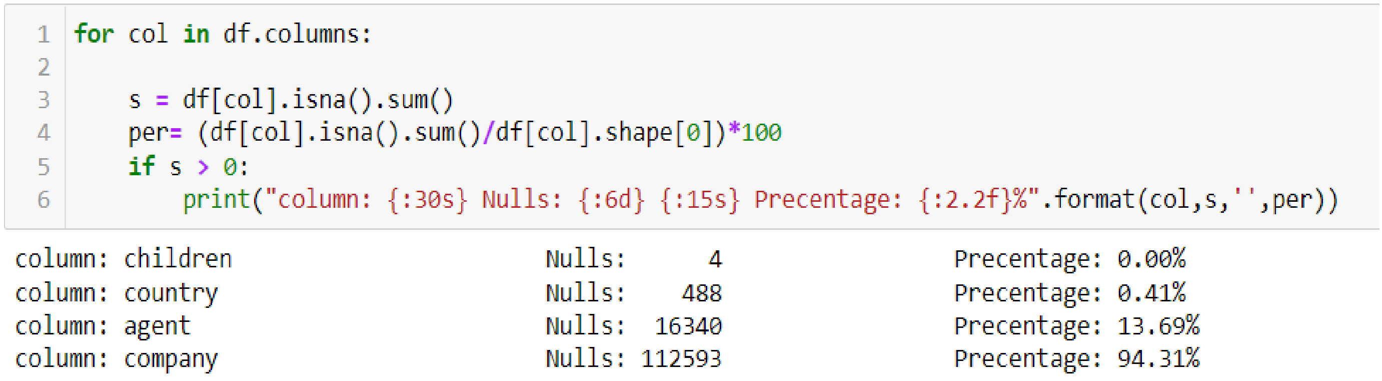
VARIABLE	TYPE	DESCRIPTION
ADR	Numeric	Average Daily Rate as defined by
Adults	Integer	Number of adults
Agent	Categorical	ID of the travel agency that made the booking[a]
ArrivalDateDayOfMonth	Integer	Day of the month of the arrival date
ArrivalDateMonth	Categorical	Month of arrival date with 12 categories: "January" to "December"
ArrivalDateWeekNumber	Integer	Week number of the arrival date
ArrivalDateYear	Integer	Year of arrival date
AssignedRoomType	Categorical	Code for the type of room assigned to the booking. Sometimes the assigned room type differs from the reserved room type due to hotel operation reasons (e.g. overbooking) or by customer request. Code is presented instead of designation for anonymity reasons
Babies	Integer	Number of babies
BookingChanges	Integer	Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
Children	Integer	Number of children
Company	Categorical	ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons
Country	Categorical	Country of origin. Categories are represented in the ISO 3155–3:2013 format.
CustomerType	Categorical	Type of booking, assuming one of four categories:
		Contract - when the booking has an allotment or other type of contract associated to it;
		Group – when the booking is associated to a group;
		Transient – when the booking is not part of a group or contract, and is not associated to other transient booking;
		Transient-party – when the booking is transient, but is associated to at least other transient booking
DaysInWaitingList	Integer	Number of days the booking was in the waiting list before it was confirmed to the customer

DepositType	Categorical	Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories:
		No Deposit – no deposit was made;
		Non Refund – a deposit was made in the value of the total stay cost;
		Refundable – a deposit was made with a value under the total cost of stay.
DistributionChannel	Categorical	Booking distribution channel. The term “TA” means “Travel Agents” and “TO” means “Tour Operators”
IsCanceled	Categorical	Value indicating if the booking was canceled (1) or not (0)
IsRepeatedGuest	Categorical	Value indicating if the booking name was from a repeated guest (1) or not (0)
LeadTime	Integer	Number of days that elapsed between the entering date of the booking into the PMS and the arrival date
MarketSegment	Categorical	Market segment designation. In categories, the term “TA” means “Travel Agents” and “TO” means “Tour Operators”
Meal	Categorical	Type of meal booked. Categories are presented in standard hospitality meal packages:
		Undefined/SC – no meal package;
		BB – Bed & Breakfast;
		HB – Half board (breakfast and one other meal – usually dinner);
		FB – Full board (breakfast, lunch and dinner)
PreviousBookingsNotCanceled	Integer	Number of previous bookings not cancelled by the customer prior to the current booking
PreviousCancellations	Integer	Number of previous bookings that were cancelled by the customer prior to the current booking
RequiredCardParkingSpaces	Integer	Number of car parking spaces required by the customer
ReservationStatus	Categorical	Reservation last status, assuming one of three categories:
		Canceled – booking was canceled by the customer;
		Check-Out – customer has checked in but already departed;
		No-Show – customer did not check-in and did inform the hotel of the reason why
ReservationStatusDate	Date	Date at which the last status was set. This variable can be used in conjunction with the *ReservationStatus* to understand when was the booking canceled or when did the customer checked-out of the hotel
ReservedRoomType	Categorical	Code of room type reserved. Code is presented instead of designation for anonymity reasons
StaysInWeekendNights	Integer	Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
StaysInWeekNights	Integer	Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel

TotalOfSpecialRequests	Integer	Number of special requests made by the customer (e.g. twin bed or high floor)

6 DATA CLEANING PHASE

6.1 Checking null values



```python
for col in df.columns:

    s = df[col].isna().sum()
    per= (df[col].isna().sum()/df[col].shape[0])*100
    if s > 0:
        print("column: {:30s} Nulls: {:6d} {:15s} Precentage: {:2.2f}%".format(col,s,'',per))
```

```
column: children                       Nulls:      4                 Precentage: 0.00%
column: country                        Nulls:    488                 Precentage: 0.41%
column: agent                          Nulls:  16340                 Precentage: 13.69%
column: company                        Nulls: 112593                 Precentage: 94.31%
```

Figure 2: Null checking implementation and results.

6.2 Dealing with nulls in company column

6.2.1 Dealing with nulls in the company column

A 94.31% of company column are missing values. Therefore, we do not have enough values to fill the rows of the company column by predicting, filling by mean, etc. It seems that the best option is dropping the company column.

6.2.2 Dealing with nulls in the agent column

A 13.69% of agent columns are missing values, there is no need to drop the agent column. But also, we should not drop the rows because 13.69% of data is a huge amount and those rows have the chance to have crucial information. There are 334 unique agents, since there are too many agents, they may not be predictable. I will decide what to do about the agent after the correlation section.

6.2.3 Dealing with nulls in the children column

We have also 4 missing values in the children column. If there is no information about children those customers do not have any children.

6.2.4 Dealing with nulls in the country column

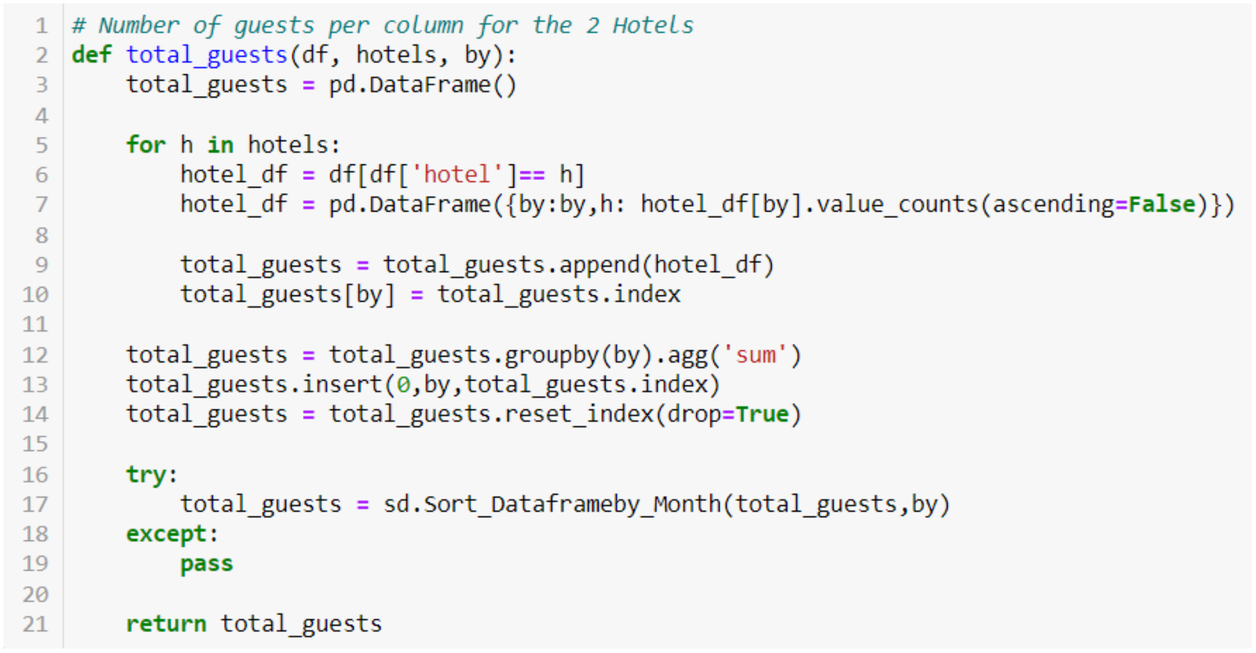
We have also only 0.41% missing values in the country column. we can simply drop them.

7 Exploratory Data Analysis (EDA)

With the help of Plolty library, we have to reply to some questions as follows:

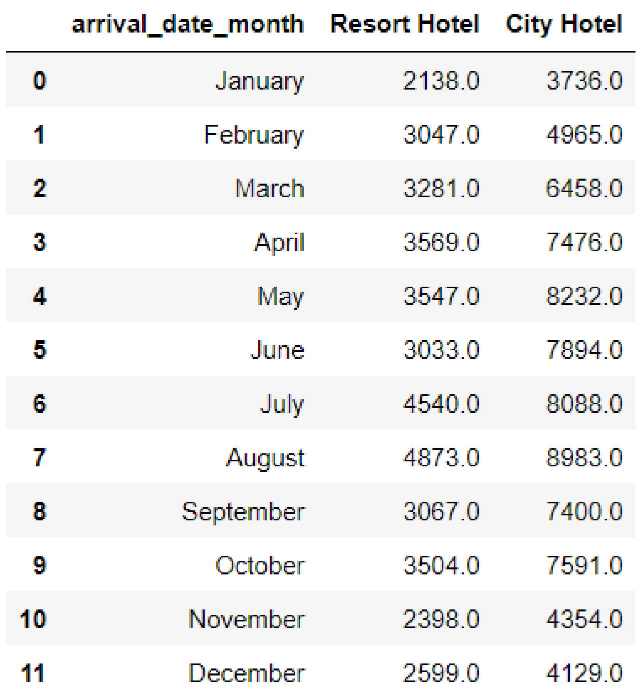
- *What is the busiest month?*
- *What is the busiest hotel?*

By implementing the (total guests) function we will have a DataFrame containing the month and number of guests per hotel



```python
# Number of guests per column for the 2 Hotels
def total_guests(df, hotels, by):
    total_guests = pd.DataFrame()

    for h in hotels:
        hotel_df = df[df['hotel']== h]
        hotel_df = pd.DataFrame({by:by,h: hotel_df[by].value_counts(ascending=False)})

        total_guests = total_guests.append(hotel_df)
        total_guests[by] = total_guests.index

    total_guests = total_guests.groupby(by).agg('sum')
    total_guests.insert(0,by,total_guests.index)
    total_guests = total_guests.reset_index(drop=True)

    try:
        total_guests = sd.Sort_Dataframeby_Month(total_guests,by)
    except:
        pass

    return total_guests
```

Figure 3: total guests function implementation.



	arrival_date_month	Resort Hotel	City Hotel
0	January	2138.0	3736.0
1	February	3047.0	4965.0
2	March	3281.0	6458.0
3	April	3569.0	7476.0
4	May	3547.0	8232.0
5	June	3033.0	7894.0
6	July	4540.0	8088.0
7	August	4873.0	8983.0
8	September	3067.0	7400.0
9	October	3504.0	7591.0
10	November	2398.0	4354.0
11	December	2599.0	4129.0

Figure 4: Total guests per hotel for each month.

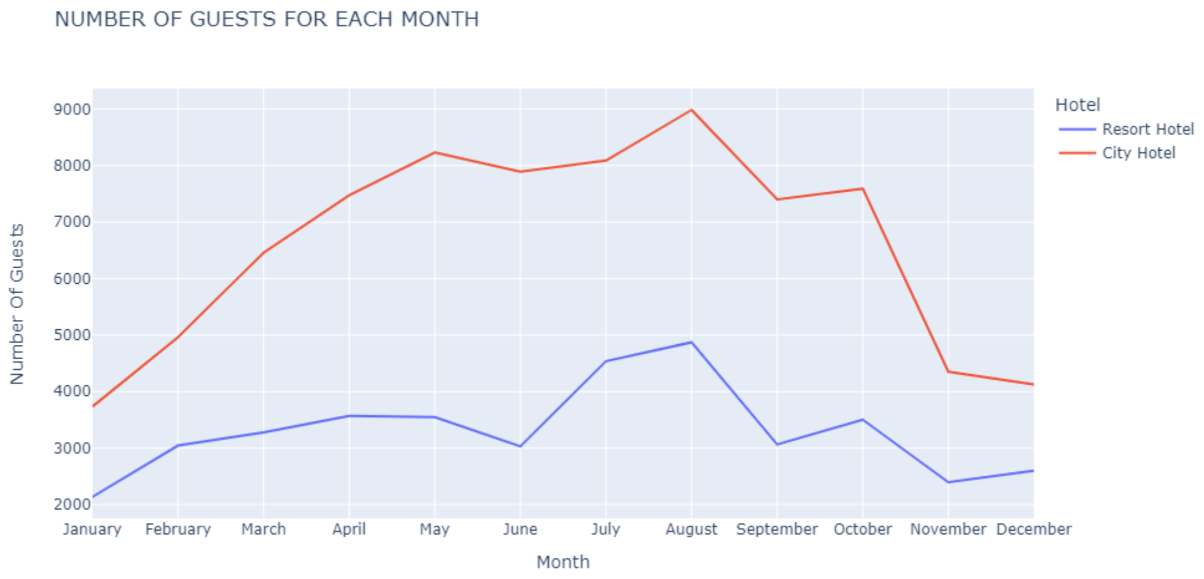


Figure 5: Number of guests for each month.

- *What is the number of guests for each time duration (per night)?*
- *What is the hotel type with more time spent?*

Most people do not seem to prefer to stay at the hotel for more than 1 week. But it seems normal to stay in Resort hotels for up to 15 days.

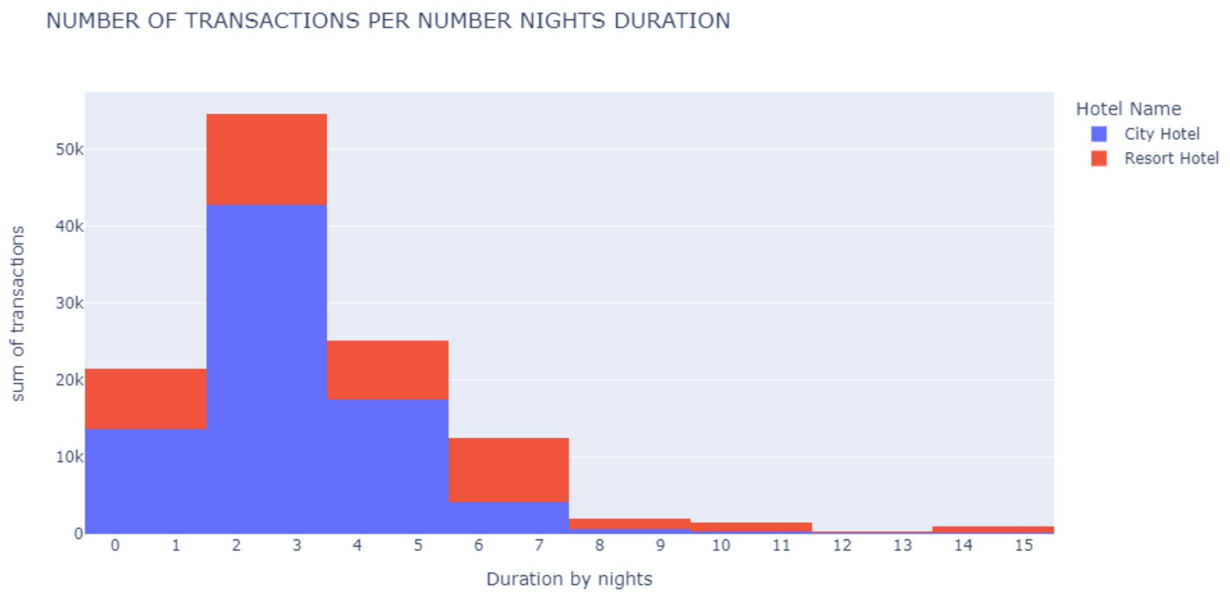
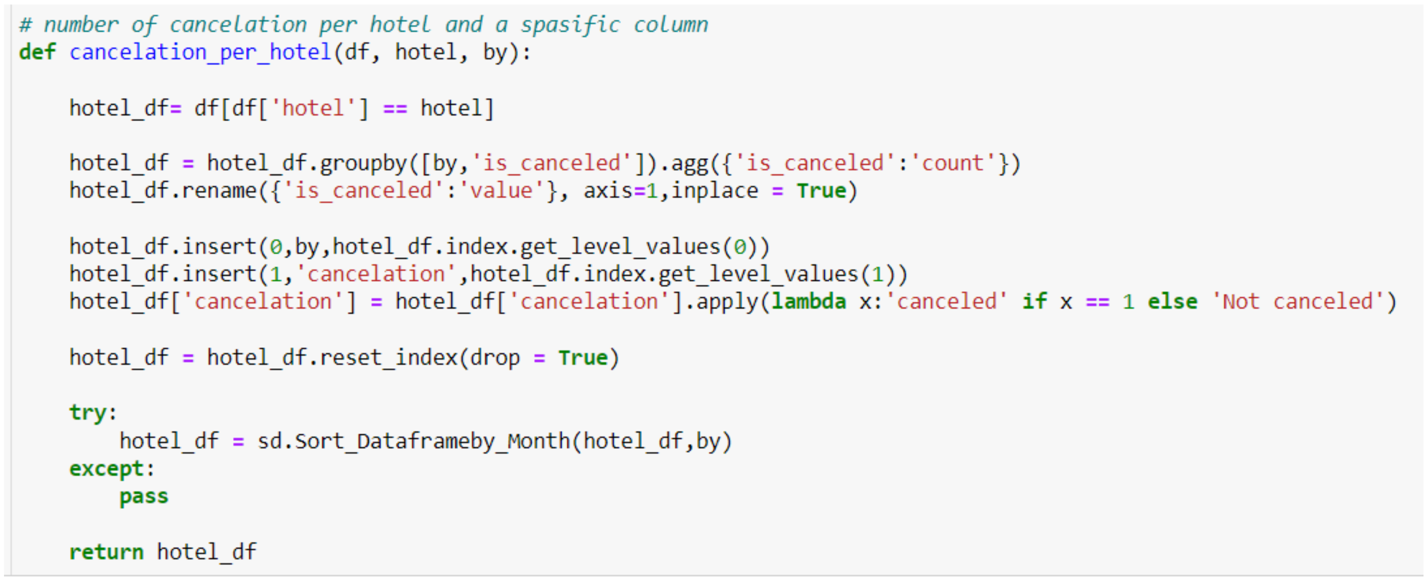


Figure 6: Number of transactions per number of nights duration.

7.1.1 How many bookings were canceled?

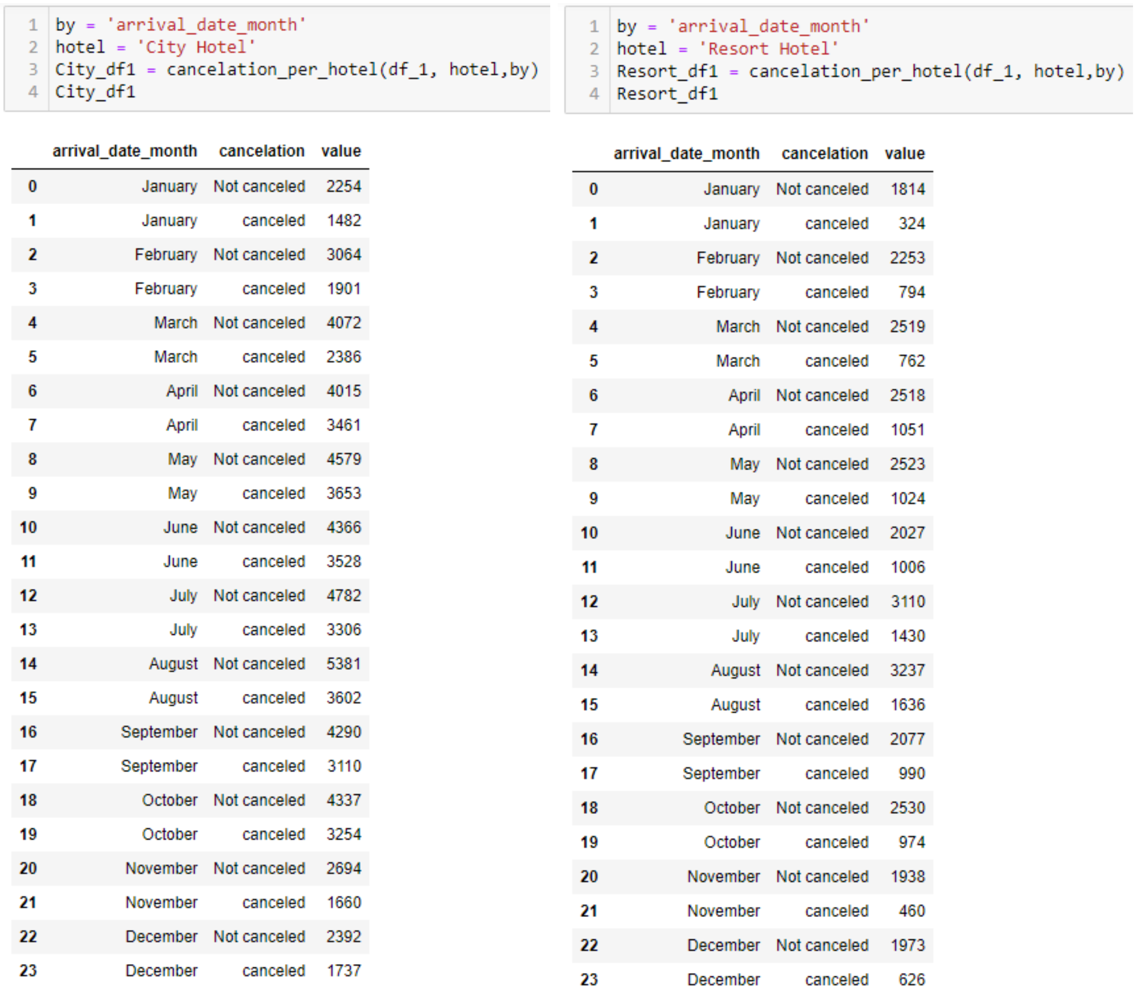
I created a function (called cancellation per hotel) to retrieve the needed information from the dataset.

- *What is the number of cancellations according to the month in both hotels?*
- *What is the number of cancellations according to customer type in both hotels?*
- *What is the number of cancellations according to waiting days type in both hotels?*
- *What is the number of cancellations of 0 waiting days and n waiting days in both hotels?*



```python
# number of cancelation per hotel and a spasific column
def cancelation_per_hotel(df, hotel, by):

    hotel_df= df[df['hotel'] == hotel]

    hotel_df = hotel_df.groupby([by,'is_canceled']).agg({'is_canceled':'count'})
    hotel_df.rename({'is_canceled':'value'}, axis=1,inplace = True)

    hotel_df.insert(0,by,hotel_df.index.get_level_values(0))
    hotel_df.insert(1,'cancelation',hotel_df.index.get_level_values(1))
    hotel_df['cancelation'] = hotel_df['cancelation'].apply(lambda x:'canceled' if x == 1 else 'Not canceled')

    hotel_df = hotel_df.reset_index(drop = True)

    try:
        hotel_df = sd.Sort_Dataframeby_Month(hotel_df,by)
    except:
        pass

    return hotel_df
```

Figure 7: The cancellation per hotel function implementation.



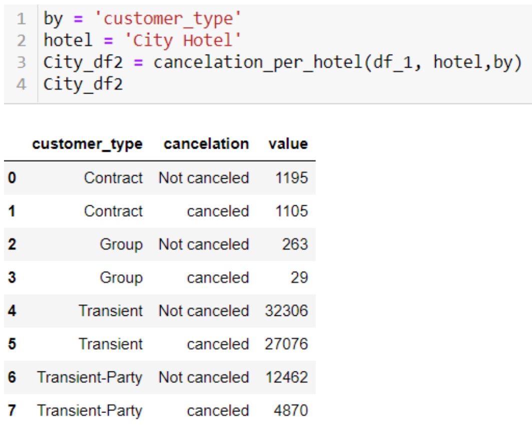
```python
by = 'arrival_date_month'
hotel = 'City Hotel'
City_df1 = cancelation_per_hotel(df_1, hotel,by)
City_df1
```

	arrival_date_month	cancelation	value
0	January	Not canceled	2254
1	January	canceled	1482
2	February	Not canceled	3064
3	February	canceled	1901
4	March	Not canceled	4072
5	March	canceled	2386
6	April	Not canceled	4015
7	April	canceled	3461
8	May	Not canceled	4579
9	May	canceled	3653
10	June	Not canceled	4366
11	June	canceled	3528
12	July	Not canceled	4782
13	July	canceled	3306
14	August	Not canceled	5381
15	August	canceled	3602
16	September	Not canceled	4290
17	September	canceled	3110
18	October	Not canceled	4337
19	October	canceled	3254
20	November	Not canceled	2694
21	November	canceled	1660
22	December	Not canceled	2392
23	December	canceled	1737

```python
by = 'arrival_date_month'
hotel = 'Resort Hotel'
Resort_df1 = cancelation_per_hotel(df_1, hotel,by)
Resort_df1
```

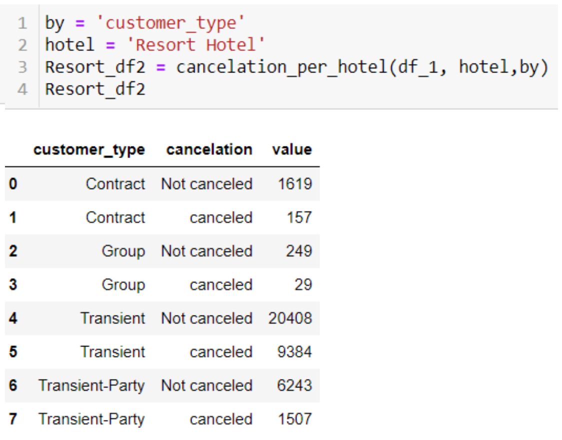
	arrival_date_month	cancelation	value
0	January	Not canceled	1814
1	January	canceled	324
2	February	Not canceled	2253
3	February	canceled	794
4	March	Not canceled	2519
5	March	canceled	762
6	April	Not canceled	2518
7	April	canceled	1051
8	May	Not canceled	2523
9	May	canceled	1024
10	June	Not canceled	2027
11	June	canceled	1006
12	July	Not canceled	3110
13	July	canceled	1430
14	August	Not canceled	3237
15	August	canceled	1636
16	September	Not canceled	2077
17	September	canceled	990
18	October	Not canceled	2530
19	October	canceled	974
20	November	Not canceled	1938
21	November	canceled	460
22	December	Not canceled	1973
23	December	canceled	626

Figure 8: DataFrames retrieved by month from the cancellation per hotel function.



```python
by = 'customer_type'
hotel = 'City Hotel'
City_df2 = cancelation_per_hotel(df_1, hotel,by)
City_df2
```

	customer_type	cancelation	value
0	Contract	Not canceled	1195
1	Contract	canceled	1105
2	Group	Not canceled	263
3	Group	canceled	29
4	Transient	Not canceled	32306
5	Transient	canceled	27076
6	Transient-Party	Not canceled	12462
7	Transient-Party	canceled	4870



```python
by = 'customer_type'
hotel = 'Resort Hotel'
Resort_df2 = cancelation_per_hotel(df_1, hotel,by)
Resort_df2
```

	customer_type	cancelation	value
0	Contract	Not canceled	1619
1	Contract	canceled	157
2	Group	Not canceled	249
3	Group	canceled	29
4	Transient	Not canceled	20408
5	Transient	canceled	9384
6	Transient-Party	Not canceled	6243
7	Transient-Party	canceled	1507

Figure 9: DataFrames retrieved by customer type from the cancellation per hotel function.

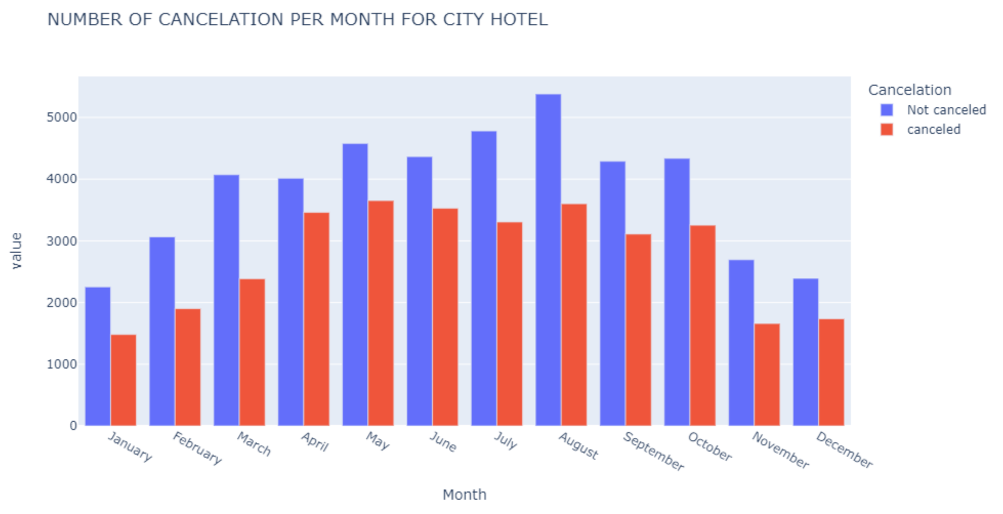


Figure 10: Number of cancellations according to the month in City hotel.

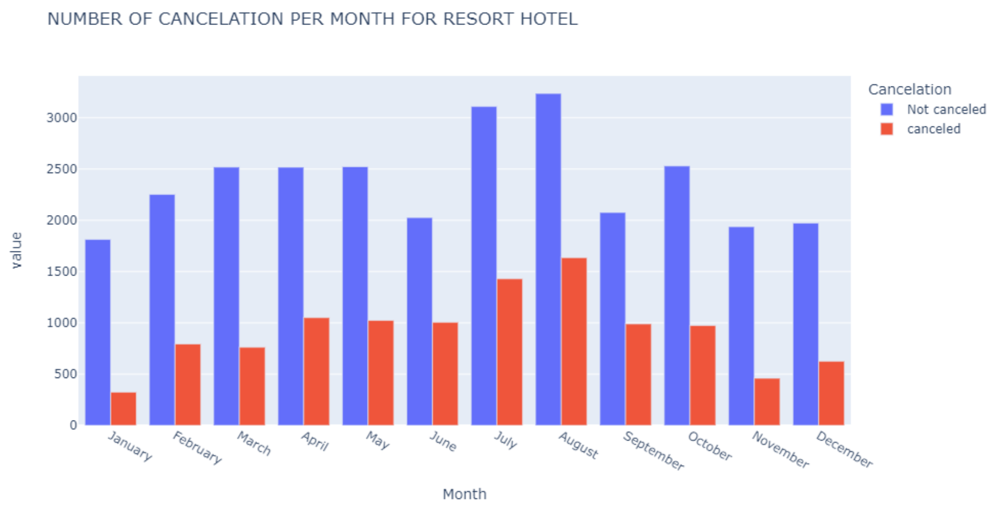


Figure 11: Number of cancellations according to the month in Resort hotel.

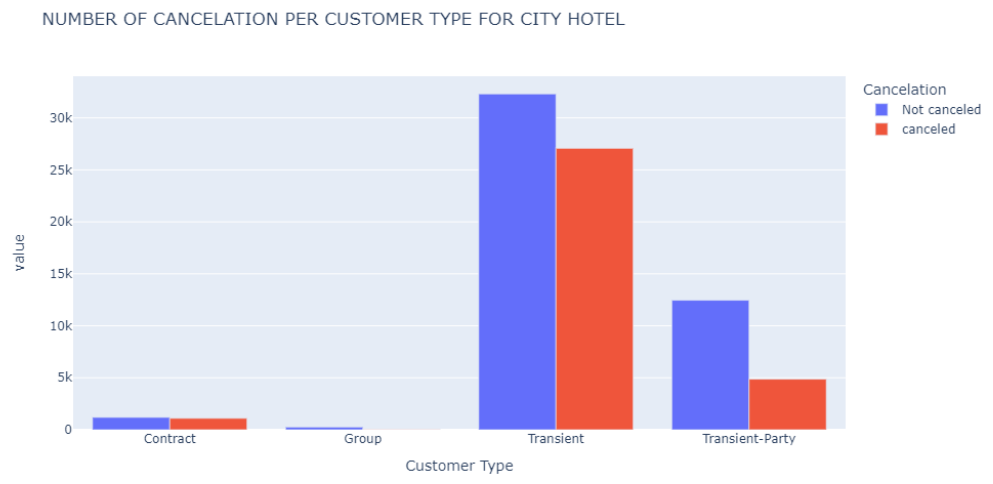


Figure 12: Number of cancellations according to the customer type in City hotel.

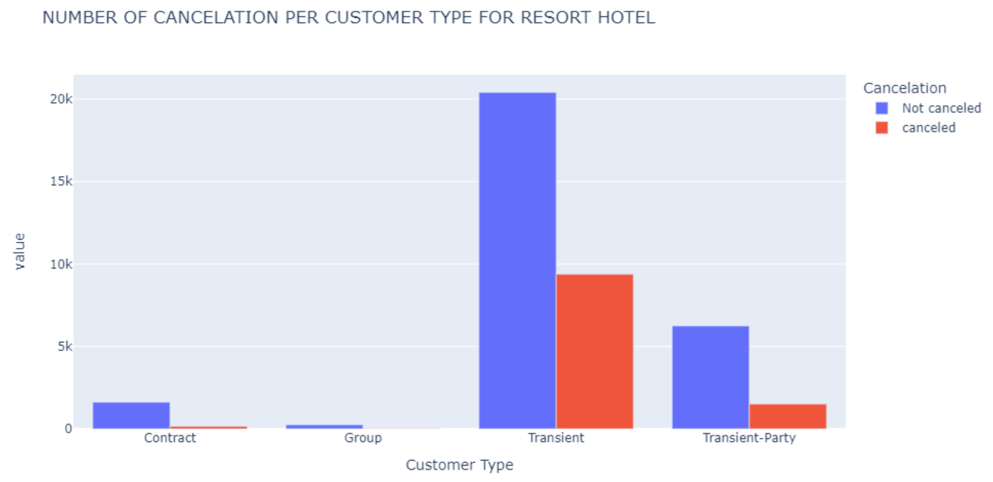
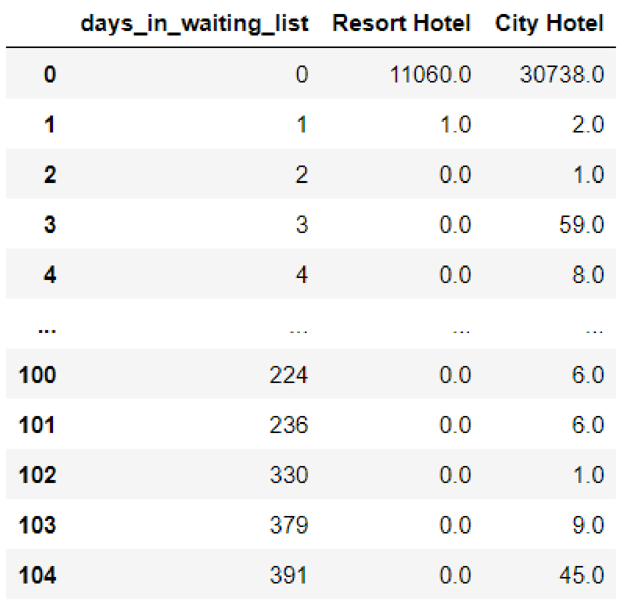


Figure 13: Number of cancellations according to the customer type in Restore hotel.



	days_in_waiting_list	Resort Hotel	City Hotel
0	0	11060.0	30738.0
1	1	1.0	2.0
2	2	0.0	1.0
3	3	0.0	59.0
4	4	0.0	8.0
...	...	...	...
100	224	0.0	6.0
101	236	0.0	6.0
102	330	0.0	1.0
103	379	0.0	9.0
104	391	0.0	45.0

Figure 14: DataFrames retrieved by days in the waiting list for both hotels.

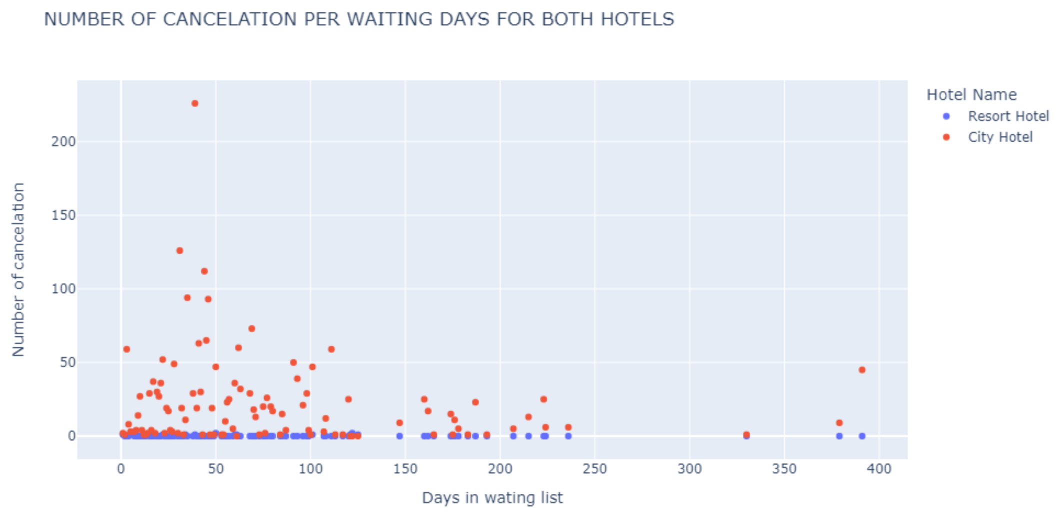
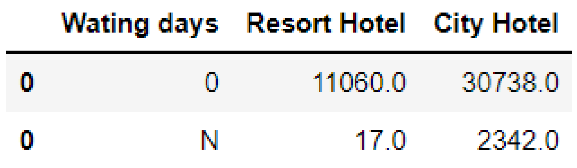


Figure 15: Number of cancelations per days on the waiting list for both hotels.

In Figures 14, 15 above, the City hotel cancelation Rate decreased when the number of days on the waiting list increased, and the Resort hotel has a very low cancelation rate compared with the first one.



	Wating days	Resort Hotel	City Hotel
0	0	11060.0	30738.0
0	N	17.0	2342.0

Figure 16: DataFrames retrieved by 0 and n days in the waiting list for both hotels.

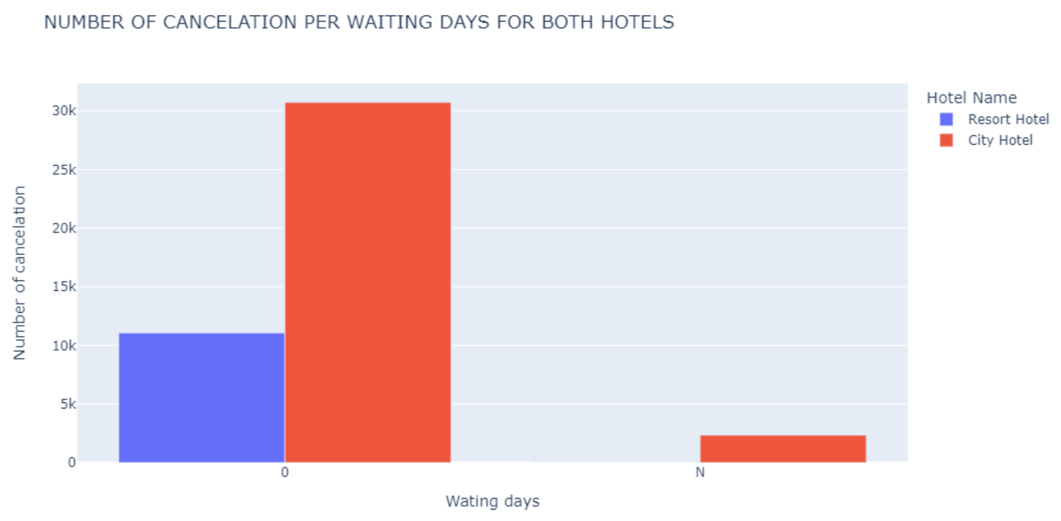


Figure 17: Number of cancelations per waiting days for both hotels.

In Figures 16, 17 above, the number of cancelations for both hotels when the waiting days = 0 is huge compared to the reminder waiting duration.

8 FEATURE ENGINEERING

8.1 Adding new features

Adding the following features to the dataset

- ***is_family***

$$x = (adults > 0 \,\&\, children > 0) \,|\, (adults > 0 \,\&\, babies > 0)$$

$$isfamily(x) = \begin{cases} 1, & x = 1 \\ 0, & x = 0 \end{cases}$$

- ***total_customer***

$$totalcustomers = adults + children + babies$$

- ***deposit_given***

$$depositgiven(x) = \begin{cases} 1, & x = '\text{Refundable}' \,||\, '\text{No Deposit}' \\ 0, & x = '\text{Non Refund}' \end{cases}$$

- ***total_nights***

$$totalnights = stays\_in\_weekend\_nights + stays\_in\_week\_nights$$

8.2 Drop useless features

I created new features more expressive than this one so I'll drop the following columns:

- *adults*
- *babies*
- *children*
- *deposit_type*
- *reservation_status_date*

8.3 Handling the categorical features

8.3.1 Replace the (hotel, arrival_date_month) features with numerical values manually.

8.3.2 Using LabelEncoder with the following columns:

- *meal*
- *distribution_channel*
- *reserved_room_type*
- *assigned_room_type*
- *agent*
- *customer_type*
- *reservation_status*
- *market_segment*

8.4 Correlation checking

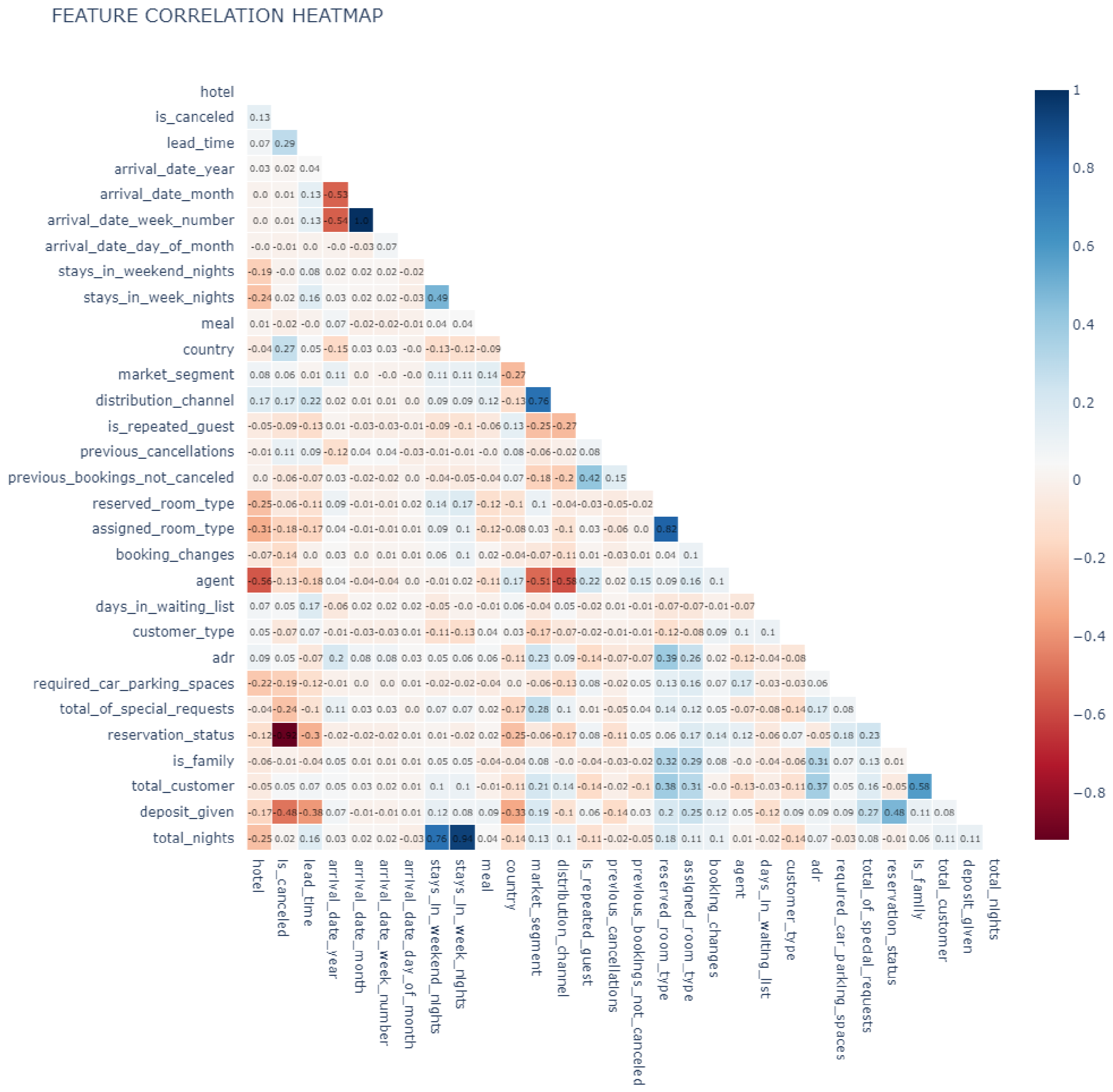


Figure 18: feature correlation heatmap.

8.4.1 High correlation features

Table 2: High correlation features.

FEATURE	CORRELATION VALUE WITH THE CANCELLATION
reservation_status	-0.917196
deposit_given	-0.481457
total_of_special_requests	-0.234658

The reservation_status seems to be the most impactful feature. With that, the information accuracy rate should be high.

8.4.2 Low correlation features

Table 3: Low correlation features.

FEATURE	CORRELATION VALUE WITH THE CANCELLATION
arrival_date_day_of_month	-0.006130
stays_in_weekend_nights	-0.001791
arrival_date_week_number	0.008148
arrival_date_year	0.016339
agent	-0.130010

Backing to the agent column which still has some missing values. It has nice importance on predicting cancellation by correlation (-0.130010) but since the missing values are equal to 13% of the total data it is better to drop that column.

8.4.3 Correlation of created features

Table 4: Correlation of created features.

FEATURE	CORRELATION VALUE WITH THE CANCELLATION
deposit_given	-0.481457
is_family	-0.013010
total_nights	0.017779
total_customer	0.046522

8.5 Drop features with low correlation

Dropping the following low features:

- *total_nights*
- *is_family*
- *arrival_date_week_number*
- *stays_in_weekend_nights*
- *arrival_date_month*
- *agent*

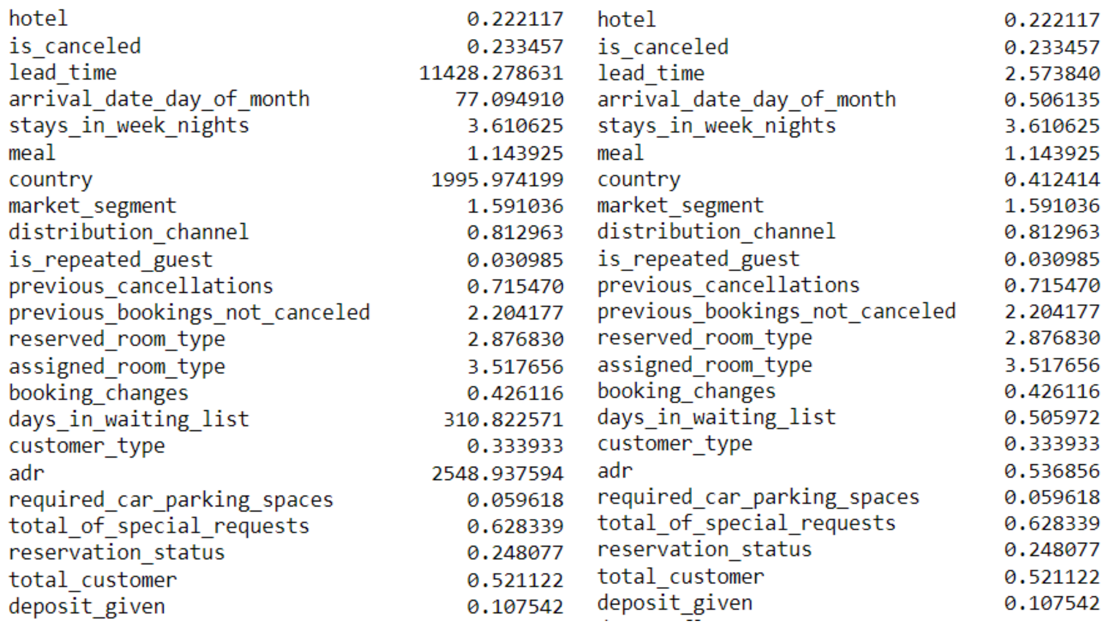
8.6 Save the last DataFrame as a CSV file.

9 MODELING

9.1 Normalize numerical features.

Normalize the following columns by logarithmic function:

- *lead_time*
- *arrival_date_day_of_month*
- *days_in_waiting_list*
- *country*
- *adr*



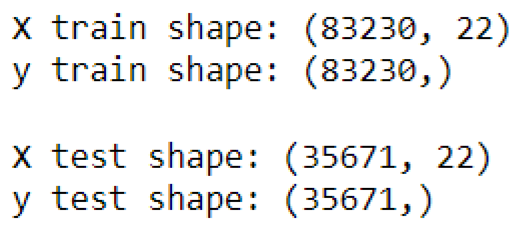
```
hotel                              0.222117
is_canceled                        0.233457
lead_time                      11428.278631
arrival_date_day_of_month         77.094910
stays_in_week_nights               3.610625
meal                               1.143925
country                         1995.974199
market_segment                     1.591036
distribution_channel               0.812963
is_repeated_guest                  0.030985
previous_cancellations             0.715470
previous_bookings_not_canceled     2.204177
reserved_room_type                 2.876830
assigned_room_type                 3.517656
booking_changes                    0.426116
days_in_waiting_list             310.822571
customer_type                      0.333933
adr                             2548.937594
required_car_parking_spaces        0.059618
total_of_special_requests          0.628339
reservation_status                 0.248077
total_customer                     0.521122
deposit_given                      0.107542
```

```
hotel                             0.222117
is_canceled                       0.233457
lead_time                         2.573840
arrival_date_day_of_month         0.506135
stays_in_week_nights              3.610625
meal                              1.143925
country                           0.412414
market_segment                    1.591036
distribution_channel              0.812963
is_repeated_guest                 0.030985
previous_cancellations            0.715470
previous_bookings_not_canceled    2.204177
reserved_room_type                2.876830
assigned_room_type                3.517656
booking_changes                   0.426116
days_in_waiting_list              0.505972
customer_type                     0.333933
adr                               0.536856
required_car_parking_spaces       0.059618
total_of_special_requests         0.628339
reservation_status                0.248077
total_customer                    0.521122
deposit_given                     0.107542
```

Figure 19: Variance before and after data normalization.

9.2 Splitting dataset to train and test sets

Using Scikit learn library, I split the whole dataset 2 training and testing sets by 70/30.



```
X train shape: (83230, 22)
y train shape: (83230,)

X test shape: (35671, 22)
y test shape: (35671,)
```

Figure 20: Training and testing set shape.

9.3 Logistic Regression with reservation status feature

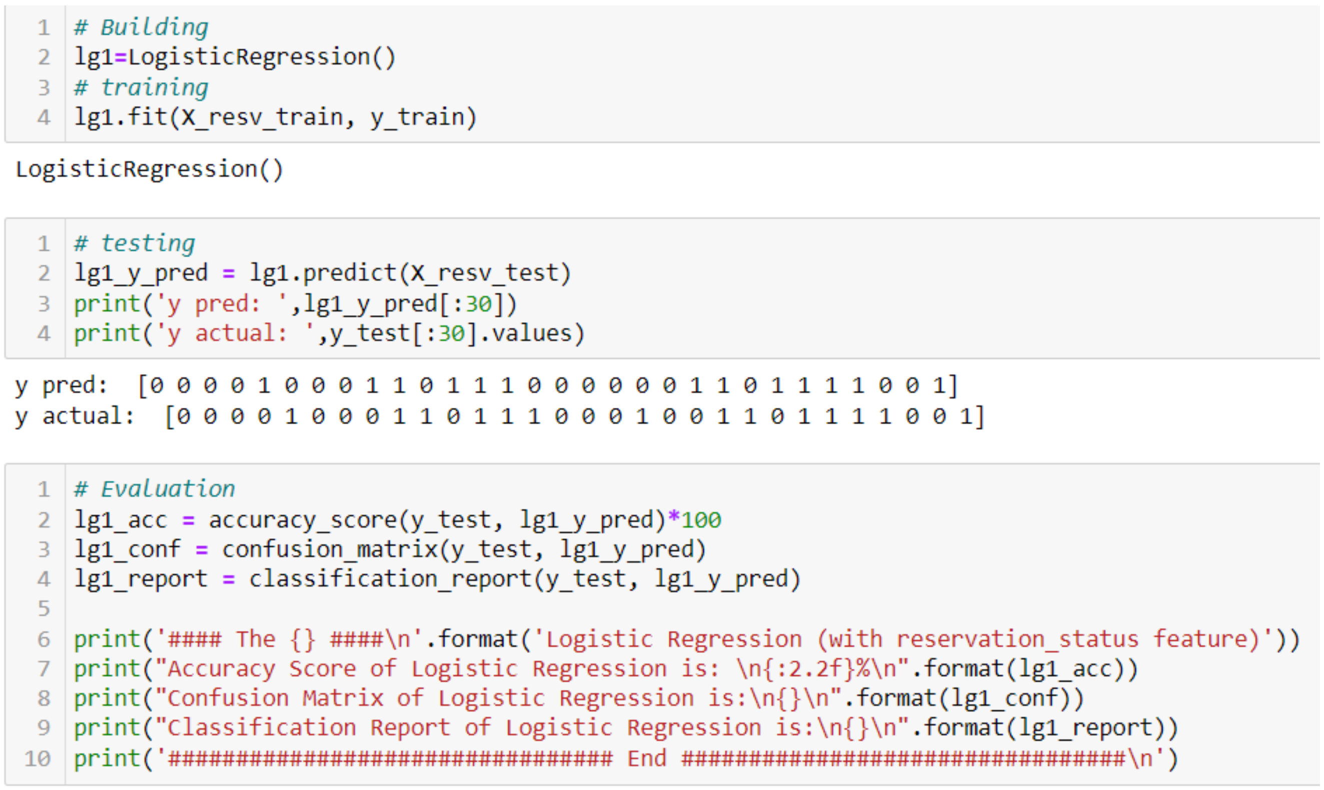
Logistic Regression is a Machine Learning algorithm that is used for classification problems, it is a predictive analysis algorithm and based on the concept of probability. Logistic Regression uses a more complex cost function, this cost function can be defined as the 'Sigmoid function' or also known as the 'logistic function'. The hypothesis of logistic regression tends to limit the cost function between 0 and 1 [5].

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Figure 21: Logistic regression hypothesis equation.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

Figure 22: Logistic Regression cost function.



```python
# Building
lg1=LogisticRegression()
# training
lg1.fit(X_resv_train, y_train)
```

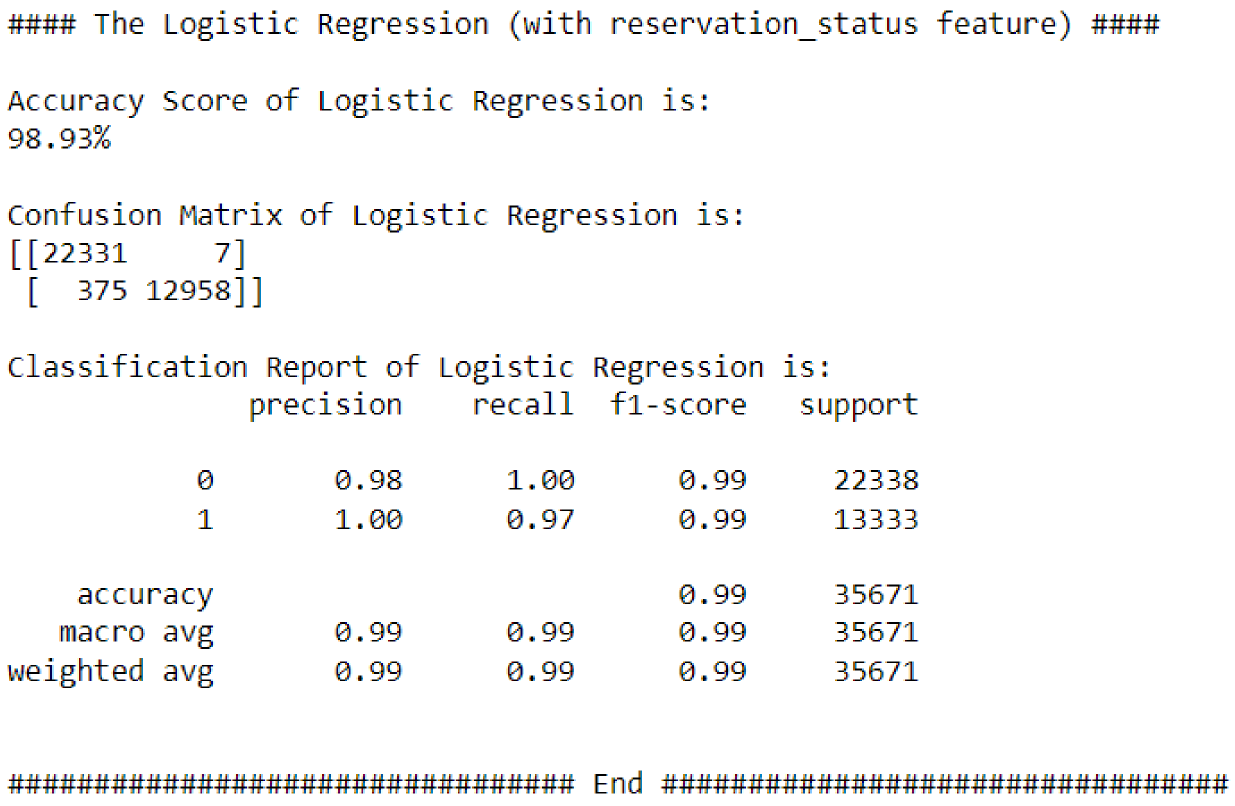
```
LogisticRegression()
```

```python
# testing
lg1_y_pred = lg1.predict(X_resv_test)
print('y pred: ',lg1_y_pred[:30])
print('y actual: ',y_test[:30].values)
```

```
y pred:  [0 0 0 0 1 0 0 0 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1]
y actual:  [0 0 0 0 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1]
```

```python
# Evaluation
lg1_acc = accuracy_score(y_test, lg1_y_pred)*100
lg1_conf = confusion_matrix(y_test, lg1_y_pred)
lg1_report = classification_report(y_test, lg1_y_pred)

print('#### The {} ####\n'.format('Logistic Regression (with reservation_status feature)'))
print("Accuracy Score of Logistic Regression is: \n{:2.2f}%\n".format(lg1_acc))
print("Confusion Matrix of Logistic Regression is:\n{}\n".format(lg1_conf))
print("Classification Report of Logistic Regression is:\n{}\n".format(lg1_report))
print('################################# End #################################\n')
```

Figure 23: Logistic regression model code stages.

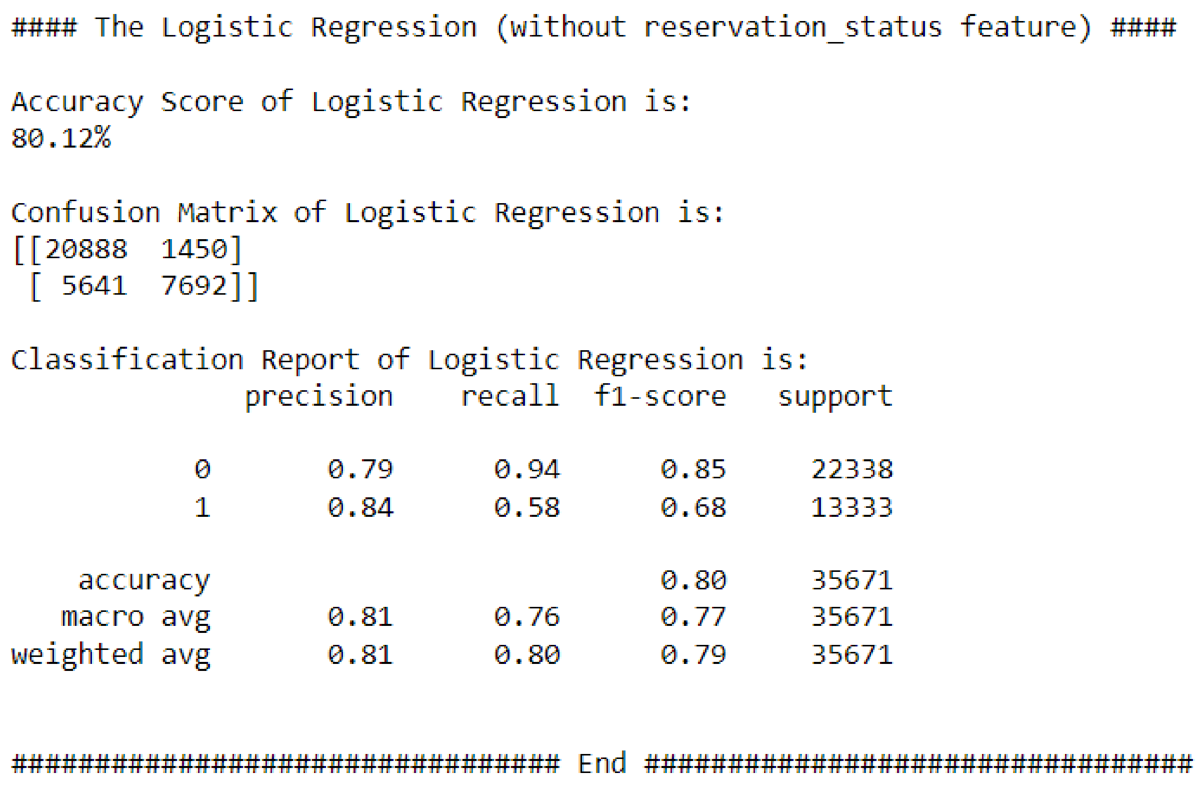


```
#### The Logistic Regression (with reservation_status feature) ####

Accuracy Score of Logistic Regression is:
98.93%

Confusion Matrix of Logistic Regression is:
[[22331     7]
 [  375 12958]]

Classification Report of Logistic Regression is:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     22338
           1       1.00      0.97      0.99     13333

    accuracy                           0.99     35671
   macro avg       0.99      0.99      0.99     35671
weighted avg       0.99      0.99      0.99     35671


################################# End #################################
```

Figure 24: Logistic regression model 1 results.

9.4 Logistic Regression without reservation status feature

The reservation_status feature has a very high correlation with the cancelation action it makes the model score almost 99% accuracy. SO, I tried to build my models without this column and compare between the different models

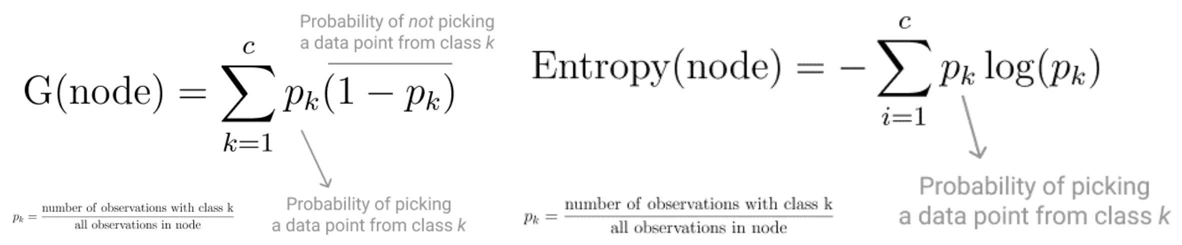


```
#### The Logistic Regression (without reservation_status feature) ####

Accuracy Score of Logistic Regression is:
80.12%

Confusion Matrix of Logistic Regression is:
[[20888  1450]
 [ 5641  7692]]

Classification Report of Logistic Regression is:
              precision    recall  f1-score   support

           0       0.79      0.94      0.85     22338
           1       0.84      0.58      0.68     13333

    accuracy                           0.80     35671
   macro avg       0.81      0.76      0.77     35671
weighted avg       0.81      0.80      0.79     35671


################################### End ###################################
```

Figure 25: Logistic regression model 2 results.

9.5 Decision Tree Classifier

A Decision Tree is a Supervised Machine Learning Algorithm that uses a set of rules to make decisions, similar to how humans make decisions. A loss function that compares the class distribution before and after the split, like Gini Impurity and Entropy [6].



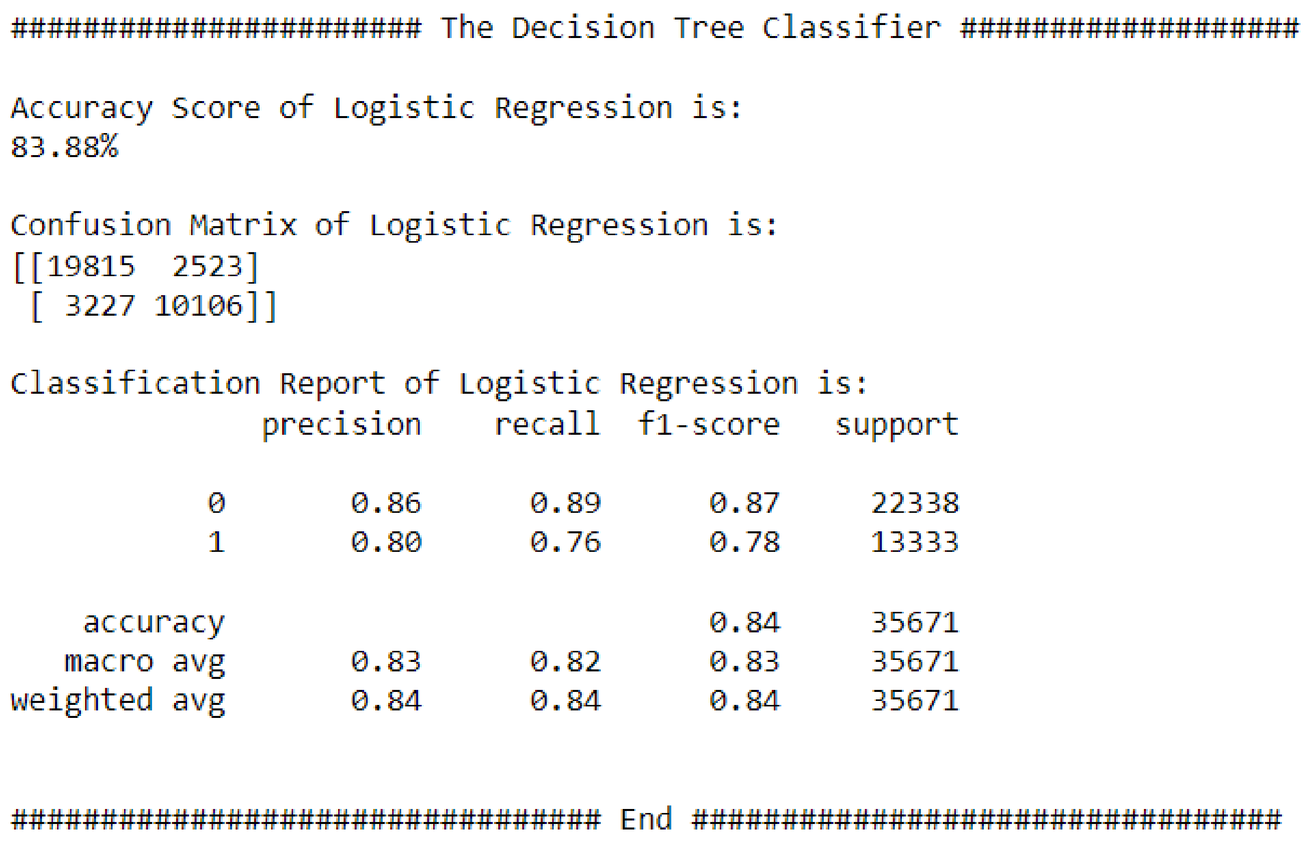
$$G(\text{node}) = \sum_{k=1}^{c} p_k(1-p_k) \qquad \text{Entropy}(\text{node}) = -\sum_{i=1}^{c} p_k \log(p_k)$$

$$p_k = \frac{\text{number of observations with class k}}{\text{all observations in node}}$$

Probability of *not* picking a data point from class *k*
Probability of picking a data point from class *k*
Probability of picking a data point from class *k*

Figure 26: Decision Tree model equations.

With max depth 15, I train the DT classifier and got these results.



```
######################## The Decision Tree Classifier ####################

Accuracy Score of Logistic Regression is:
83.88%

Confusion Matrix of Logistic Regression is:
[[19815  2523]
 [ 3227 10106]]

Classification Report of Logistic Regression is:
              precision    recall  f1-score   support

           0       0.86      0.89      0.87     22338
           1       0.80      0.76      0.78     13333

    accuracy                           0.84     35671
   macro avg       0.83      0.82      0.83     35671
weighted avg       0.84      0.84      0.84     35671


################################### End ###################################
```

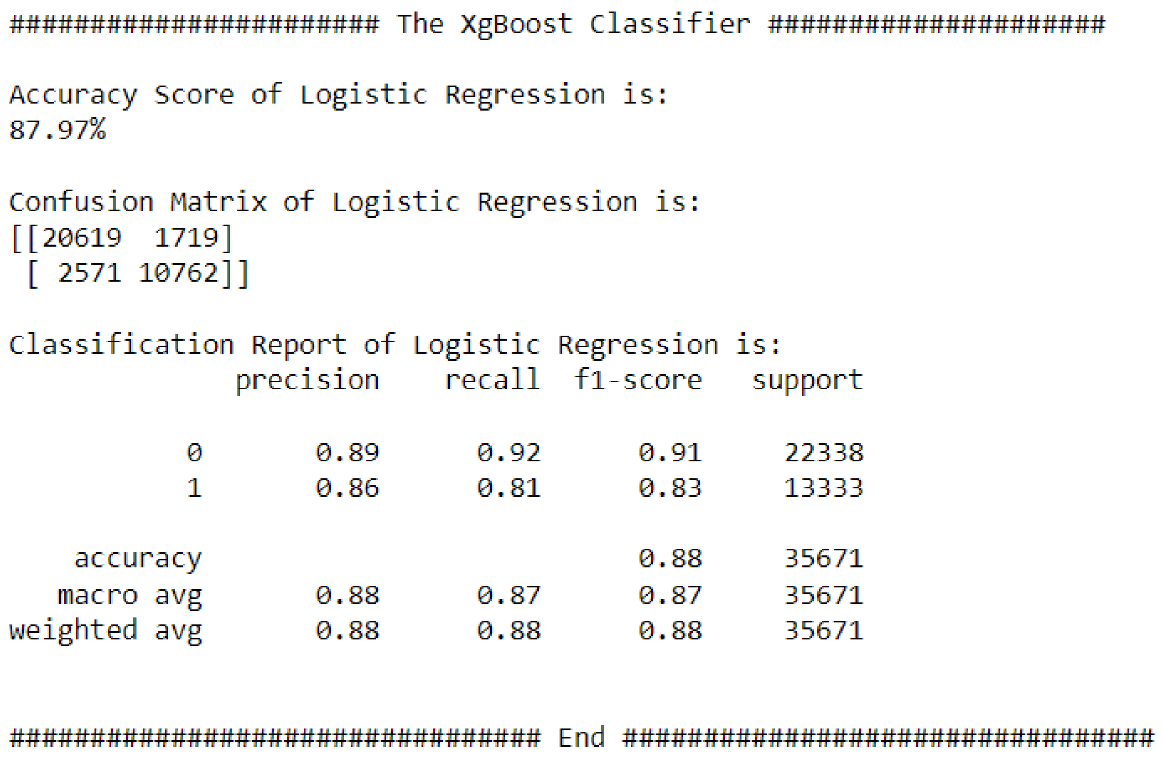
Figure 27: Decision Tree classifier results.

9.6 XgBoost Classifier

XGBoost algorithm basically reduces the error by using gradient descent optimization. It uses a regularized function which is a combination of the loss function and a penalty term for the complexity of the model. The training in this algorithm works in an iterative manner which appends new trees that predict the errors of previous trees and then are put together with the previous trees to predict the final output [7].

Table 5: XgBoost Classifier Parameters.

PARAMETER	VALUE
Booster	‘gbtree’ uses tree-based model.
learning_rate	0.1
max_depth	15
n_estimators	500



```
####################### The XgBoost Classifier #####################

Accuracy Score of Logistic Regression is:
87.97%

Confusion Matrix of Logistic Regression is:
[[20619  1719]
 [ 2571 10762]]

Classification Report of Logistic Regression is:
              precision    recall  f1-score   support

           0       0.89      0.92      0.91     22338
           1       0.86      0.81      0.83     13333

    accuracy                           0.88     35671
   macro avg       0.88      0.87      0.87     35671
weighted avg       0.88      0.88      0.88     35671


################################# End ################################
```

Figure 28: XgBoost classifier results.

9.7 Random Forest Classifier using Grid Search CV

9.7.1 Cross-Validation

In K-Fold CV, we further split our training set into K number of subsets, called folds. We then iteratively fit the model K times, each time training the data on K-1 of the folds and evaluating on the Kth fold (called the validation data) [8].

For hyperparameter tuning, we perform many iterations of the entire K-Fold CV process, each time using different model settings. We then compare all of the models, select the best one, train it on the full training set, and then evaluate the testing set [8].

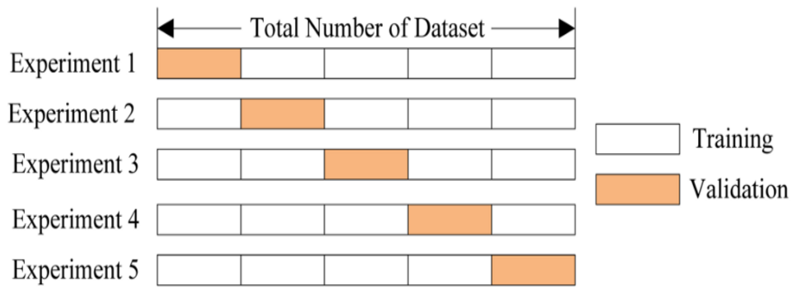
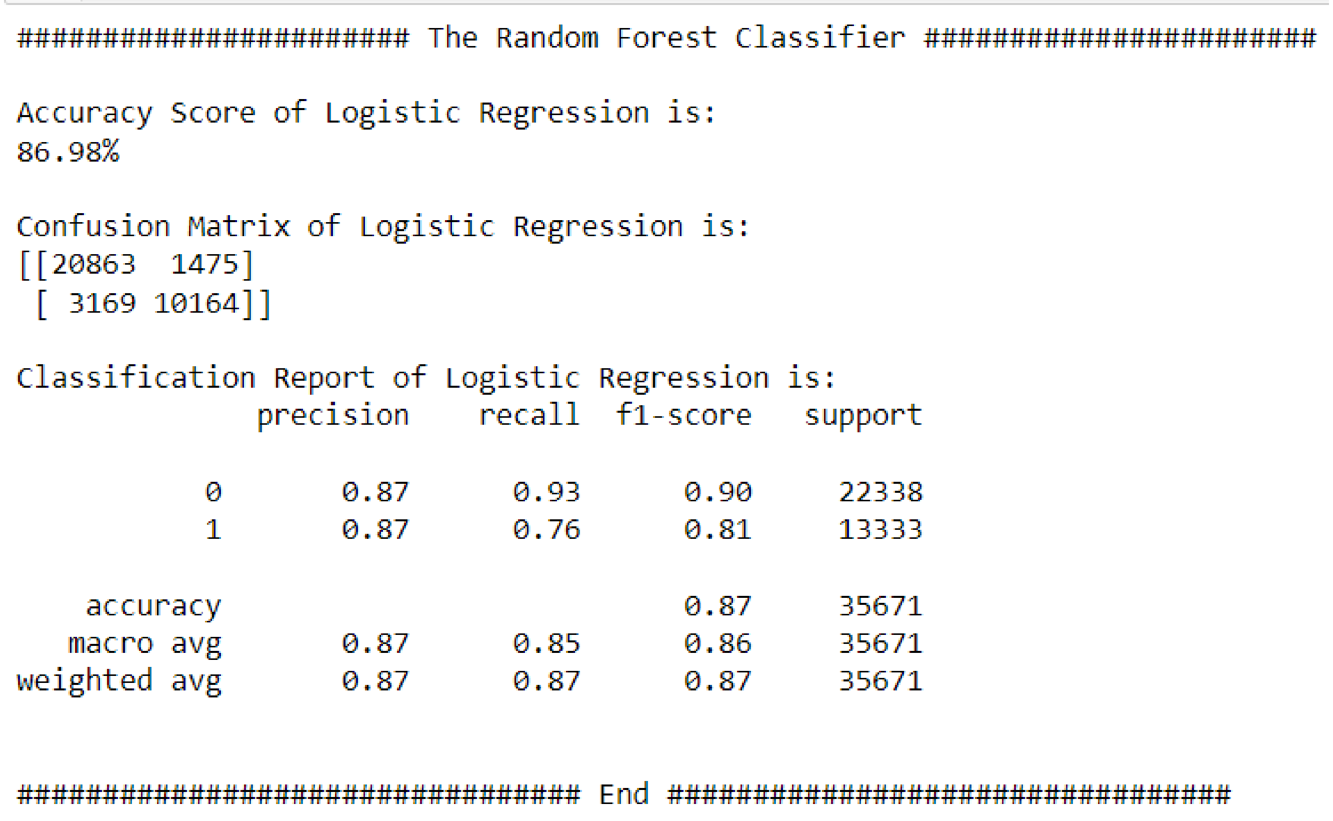


Figure 29: Cross-validation splitting criteria.

Using Scikit-Learn's RandomizedSearchCV method, we can define a grid of hyperparameter ranges and randomly sampled it from the grid, performing K-Fold CV with each combination of values.

Table 6: Random Forest classifier parameters.

PARAMETER	VALUE
max_depth	[16,18,20]
n_estimators	[100,500]
min_samples_split	[2,5]
CV	5
n_jobs	-1



```
####################### The Random Forest Classifier #######################

Accuracy Score of Logistic Regression is:
86.98%

Confusion Matrix of Logistic Regression is:
[[20863  1475]
 [ 3169 10164]]

Classification Report of Logistic Regression is:
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     22338
           1       0.87      0.76      0.81     13333

    accuracy                           0.87     35671
   macro avg       0.87      0.85      0.86     35671
weighted avg       0.87      0.87      0.87     35671


################################# End #################################
```

Figure 30: Best Random Forest classifier results.

10 CONCLUSION

After studying, analyzing, cleansing the data, as well as understanding the correlation between the features and the impact of each on the cancellation, I created five classification models, the results of each were as follows in Table 7.

The results show that there is one feature (called reservation status) that has a very high correlation with the target column (is canceled), and a logistic regression classifier gives an accuracy of 99% while the same model gives an accuracy of 81% when this feature is removed. As a result, I choose to delete the feature and conduct a fair comparison of the different classifiers. The XgBoost classifier achieved the highest accuracy of 88 percent.

Table 7: Models Comparison.

MODEL NAME	**ACCURACY**
LR (with reservation)	98.93%
LR (without reservation)	80.12%
DT Classifier	83.88%
XgBoost Classifier	87.97%
RF Classifier using CV	86.98%

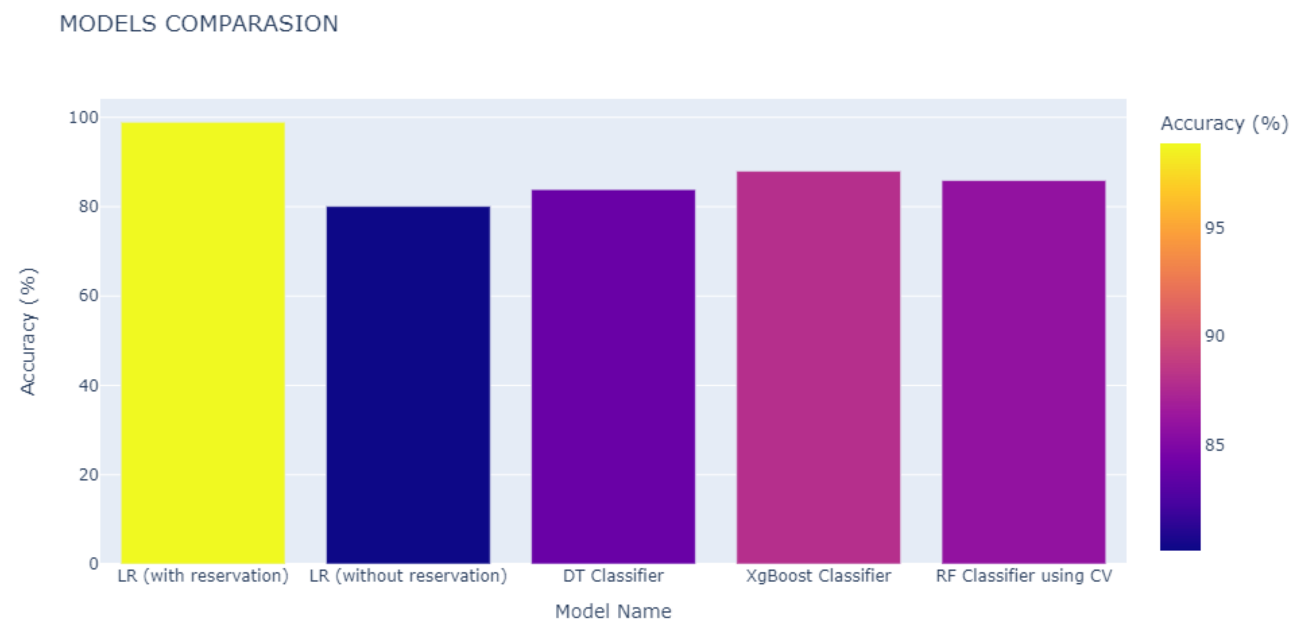


Figure 31: Models comparison.

11 REFERENCES

[1] Hotel booking demand datasets. (2019, February 1). ScienceDirect. Retrieved December 17, 2021, from https://www.sciencedirect.com/science/article/pii/S2352340918315191#bib2

[2] Data Science Project Lifecycle | Lifecycle of Data Science Project. (2021, July 6). Analytics Vidhya. Retrieved December 17, 2021, from https://www.analyticsvidhya.com/blog/2021/05/introduction-to-data-science-project-lifecycle/

[3] Pant, A. (2021, December 7). Introduction to Logistic Regression - Towards Data Science. Medium. Retrieved December 17, 2021, from https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[4] A.V.I.H.A. (n.d.). report.docx - XGBoost Algorithm In Machine learning. Course Hero. Retrieved December 17, 2021, from https://www.coursehero.com/file/79258686/reportdocx/

[5] Koehrsen, W. (2019, December 10). Hyperparameter Tuning the Random Forest in Python - Towards Data Science. Medium. Retrieved December 17, 2021, from https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74