# PROPOSAL REPORT

## TEAM 13


## ABSTRACT

In this pilot study, we have used a publicly available real-world dataset[1] of two hotels in Portugal: City Hotel and Resort Hotel. We propose a decision support system that involves 1) modeling hotel data based on machine learning techniques to produce probability estimations 2) leverage these estimations through the expected value (revenue) framework for an overbooking limit that optimizes for revenue across underbooked and overbooked scenarios.


# Business and Data Understanding

## INTRODUCTION

In this project, we tackle a **revenue management issue** prevalent across many businesses, particularly within the hospitality industry, that arises from the preponderance of no-shows and last-minute cancellations. Hotels, in particular, have used various measures in the past to mitigate risk through several incentive and disincentive-based methods. Particularly, incentive methods can involve providing several attractive offers to customers in order to increase the likelihood of their honoring a booking; disincentive methods can be restrictive in nature and have other negative effects for the business. However, we found that the former involves causal analysis of different offers to customers and its effect on no-shows, while the latter must include a study of non-refund policies on a business' overall demand and competitiveness in the market to account for negative consequences. There is a third measure that businesses can, and do, leverage to mitigate loss of revenue and that is through: ***overbooking*****.** Currently, most businesses already collect the data that is needed for such a study i.e., information on bookings made with their hotel. This makes it perfect for a pilot study by a data science team because it does not require engineering infrastructure for new sources of data to begin.

# BUSINESS PROBLEM FORMULATION

The business problem to be solved is mitigating against the risk of revenue loss due to no-shows for hotels. The data science solution formulation solves for this through an overbooking limit that maximizes revenue. However, several assumptions have been made throughout this study and we highlight them based on relevance.

The study was conducted on a real-world dataset of two hotels (City; Resort) in Portugal. Both datasets have the same structure with 36 variables:

```
df.columns
```

```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'name', 'email',
       'phone-number', 'credit_card'],
      dtype='object')
```

The dataset contains a total of 119, 390 instances:

```
['Resort Hotel' 'City Hotel']
City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
```

# UNDERSTANDING THE DATA:

## Target variable

| | |
|---|---|
| is_canceled | binary variable and 1 represents no-show bookings, 0 represents check-ins/shows |

## Predictor variables

| | |
|---|---|
| lead_time | numeric variable to represent the number of days that elapsed between the entering date of the booking into the PMS and the arrival date |
| arrival_date_year | categorical variable that represents the year of arrival date |
| arrival_date_month | categorical variable that represents the month of arrival date |
| arrival_date_week | categorical variable that represents the week of arrival date (W1 to W53) |
| arrival_date_day_month | categorical variable that represents the day of the month of the arrival date (1 to 31) |
| stays_in_weekend_nights | numeric variable that represents the number of weekend nights that the guest stayed or booked to stay at the hotel |
| stays_in_week_nights | numeric variable that represents the number of the week nights that the guest stayed or booked to stay at the hotel |
| adults | numeric variable that represents the number of adult guests |
| children | numeric variable that represents the number of child guests |
| babies | numeric variable that represents the number of baby guests |
| meal | categorical variable that represents the type of meal booked |
| country | categorical variable that represents the country of origin of the guest |

| | |
|---|---|
| market_segment | categorical variable that represents the market segment designation |
| distribution channel | categorical variable that represents the booking distribution channel |
| is_repeated_guest | binary variable that represents whether the booking was made by a repeated guest or not |
| previous_cancellations | numeric variable that represents the number of previous bookings that were canceled by the customer prior to the current booking |
| previous_booking_cancellations | numeric variable that represents the number of booking not canceled by the customer prior to the current booking |
| reserved_room_type | categorical variable that represents the code of the room type reserved |
| assigned_room_type | categorical variable that represents the room type assigned to the booking. Sometimes the assigned room type differs from the reserved room type due |
| booking_changes | numeric variable that represents the number of changes made to the booking |
| deposit_type | categorical variable that represents the type of deposit made ("No Deposit", "No refund", "Refundable") |
| agent | categorical variable that represents the ID of the travel agency that made the booking |
| company | numeric variable that represents the ID of the company of entity that made the booking or responsible for paying the booking |

| | |
|---|---|
| days_in_waiting_line | numeric variable that represents the booking was in the waiting list before it was confirmed to the customer |
| customer_type | categorical variable that represents whether the booking is associated to a group or transient (not part of a group or contract) |
| adr | numeric variable that represents the average daily rate (calculated by dividing the sum of all lodging transactions by the total number of staying nights) |
| required_car_parking | numeric variable that represents the number of car parking spaces required by the customer |
| total_of_special_requests | numeric variable that represents the number of special requests made by the customer |
| reservation_status | categorical variable that represents the status of the reservation |
| reservation_status_date | date variable that represents the date of latest update to reservation status |

## Personally Identifiable Information

Note: These features were replaced with dummy information.

| | |
|---|---|
| name | text variable that represents the name of the customer |
| email | text variable that represents the name of the customer |
| phone-number | text variable that represents the name of the customer |

| | |
|---|---|
| credit_card | text variable that represents the name of the customer |

1. This is a supervised learning problem.
2. Each instance corresponds to a booking.
3. The target variable “is_canceled” is defined to be a binary variable. That is, the booking was either honored (0) or not honored (1).
4. The dataset was prepared with dummy values in columns pertaining to PII (personally identifiable information) “name”, “email”, “phone-number” and “credit_card” due to privacy concerns. We drop these columns entirely.
5. On analyzing the data, we treat for null values:
    a. COMPANY: > 90% of null values; therefore this column is dropped
    b. COUNTRY: < 5% of null values; therefore imputation by mode value
    c. AGENT: <15% of null values; therefore imputation by mode value
    d. CHILDREN: 4 records of null values; therefore instances are dropped
6. For our analysis, we consider only the City Hotel case as it has the higher number of instances within the dataset. However, similar analysis can be performed for Resort Hotel as well with a few modifications.
7. City Hotel has 79326 instances in total.
8. High-level Summary of Numeric/Categorical Features:

```
        Summary of numeric variables

booking_changes has 21 unique values and 0 has highest frequency
lead_time has 453 unique values and 0 has highest frequency
stays_in_weekend_nights has 14 unique values and 0 has highest frequency
stays_in_week_nights has 29 unique values and 2 has highest frequency
adults has 5 unique values and 2 has highest frequency
children has 4 unique values and 0.0 has highest frequency
babies has 5 unique values and 0 has highest frequency
previous_cancellations has 10 unique values and 0 has highest frequency
previous_bookings_not_canceled has 73 unique values and 0 has highest frequency
days_in_waiting_list has 115 unique values and 0 has highest frequency
adr has 5405 unique values and 62.0 has highest frequency
required_car_parking_spaces has 4 unique values and 0 has highest frequency
total_of_special_requests has 6 unique values and 0 has highest frequency
```

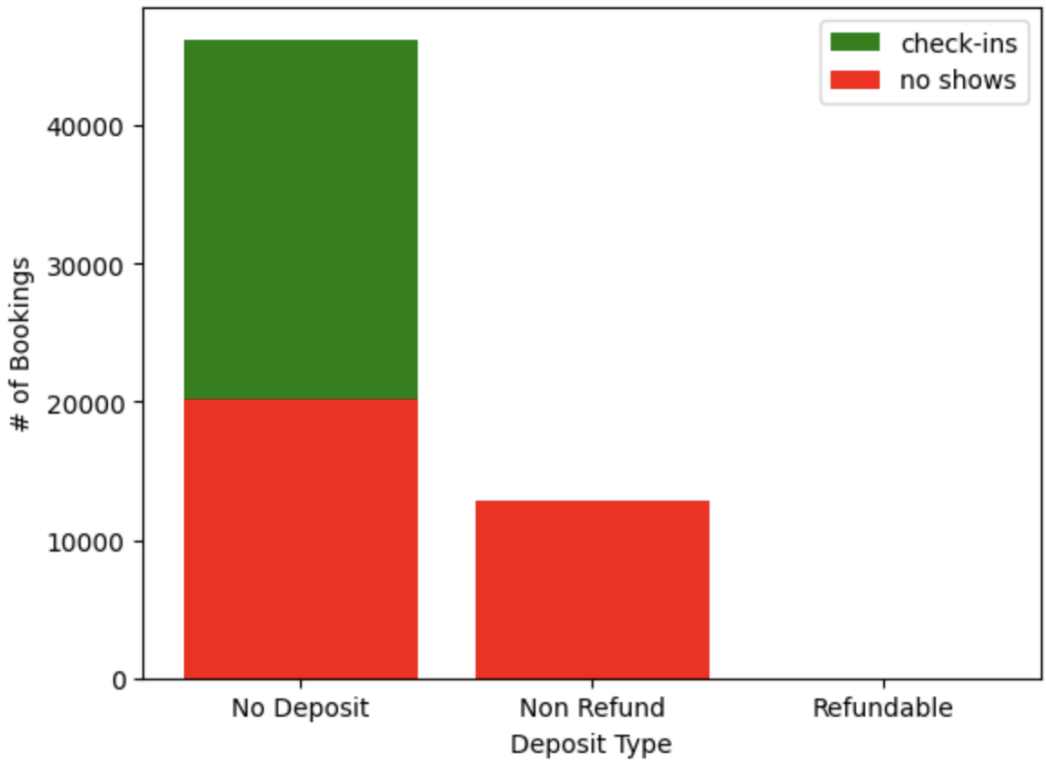
```
        Summary of categorical variables

arrival_date_year has 3 unique values and 2016 has highest frequency
arrival_date_month has 12 unique values and August has highest frequency
arrival_date_week_number has 53 unique values and 33 has highest frequency
arrival_date_day_of_month has 31 unique values and 17 has highest frequency
meal has 4 unique values and BB has highest frequency
is_repeated_guest has 2 unique values and 0 has highest frequency
country has 166 unique values and PRT has highest frequency
market_segment has 7 unique values and Online TA has highest frequency
distribution_channel has 4 unique values and TA/TO has highest frequency
reserved_room_type has 8 unique values and A has highest frequency
assigned_room_type has 9 unique values and A has highest frequency
deposit_type has 3 unique values and No Deposit has highest frequency
agent has 223 unique values and 9.0 has highest frequency
customer_type has 4 unique values and Transient has highest frequency
reservation_status has 3 unique values and Check-Out has highest frequency
```

9. SUMMARY STATISTICS:

| | is_canceled | lead_time | arrival_date_year | arrival_date_week_number | arrival_date_day_of_month | stays_in_weekend_nights | stays_in_week_nights |
|---|---|---|---|---|---|---|---|
| count | 79326.000000 | 79326.000000 | 79326.000000 | 79326.000000 | 79326.000000 | 79326.000000 | 79326.000000 |
| mean | 0.417240 | 109.741106 | 2016.174344 | 27.177193 | 15.787094 | 0.795187 | 2.182954 |
| std | 0.493106 | 110.948732 | 0.699149 | 13.398812 | 8.728379 | 0.885029 | 1.456398 |
| min | 0.000000 | 0.000000 | 2015.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 23.000000 | 2016.000000 | 17.000000 | 8.000000 | 0.000000 | 1.000000 |
| 50% | 0.000000 | 74.000000 | 2016.000000 | 27.000000 | 16.000000 | 1.000000 | 2.000000 |
| 75% | 1.000000 | 163.000000 | 2017.000000 | 38.000000 | 23.000000 | 2.000000 | 3.000000 |
| max | 1.000000 | 629.000000 | 2017.000000 | 53.000000 | 31.000000 | 16.000000 | 41.000000 |

10. DEPOSIT_TYPE : This feature has three unique values - ‘No Deposit’, ‘Non-refund’, ‘Refundable’. Since the non-refundable case does not fit within the scenarios to be considered in this study, we have dropped these instances. Total instances drop to 66458 (check-ins ~40k; no shows ~20k).




11. RESERVATION_STATUS: This feature has three unique values - 'Check-Outs', 'Canceled', 'No-Shows'.

    CHECK-OUTS: Customer checked in and checked out; is_canceled = 0
    CANCELED: Customer canceled the booking; is_canceled = 1
    NO SHOWS: Customer did not show up on arrival date; is_canceled = 1

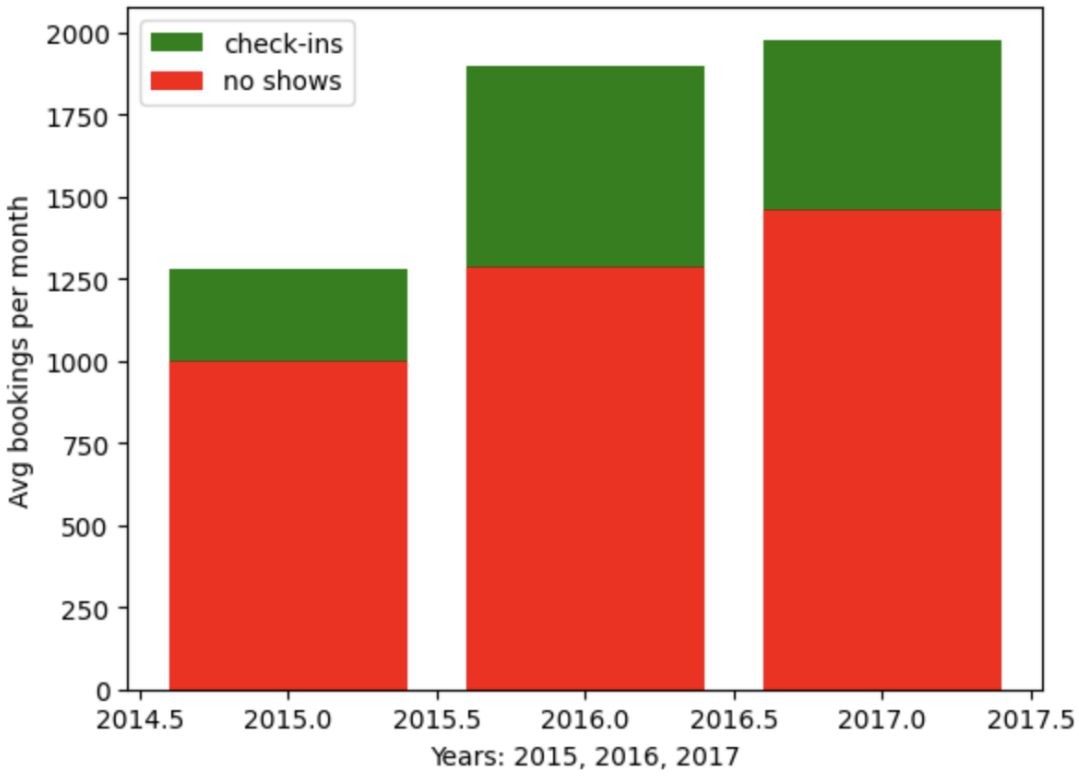
We leveraged "reservation_status_date", "arrival_date" and "reservation_status" in conjunction to compute a new field "time_diff_reservation_to_arrival" and it allowed us to analyze how many cancellations were made well in advance and how many cancellations were made ***last minute.*** The idea of ***last minute*** depends on when the business decides to take actionable steps based on the modeling predictions and, for the purpose of our study, this is assumed to be 30 days.

Since no-shows represent a small pool of the "is_canceled = 1" case, we used a threshold of <=30 days such that if a cancellation was made <=30 days out, it would still be treated as a no-show.

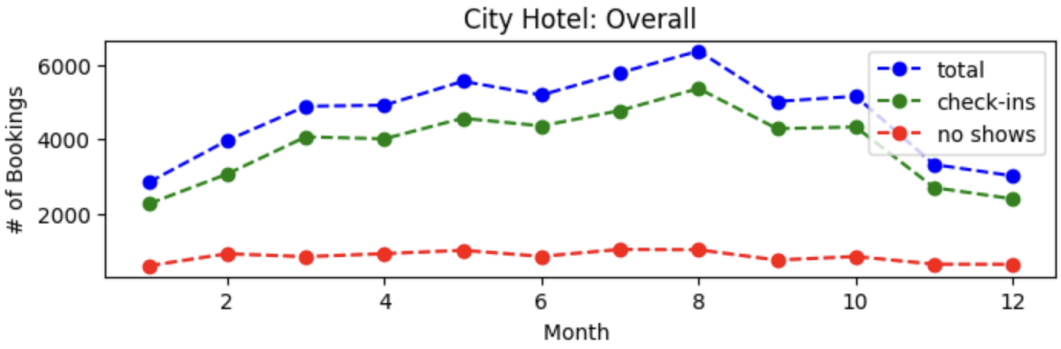
This thresholding reduces our dataset to 56114 records (check-ins ~46k; no shows ~10k)

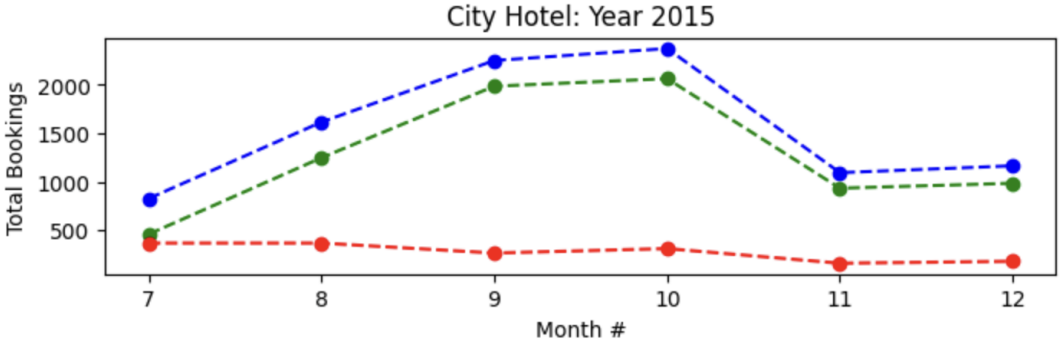
# EXPLORATORY DATA ANALYSIS

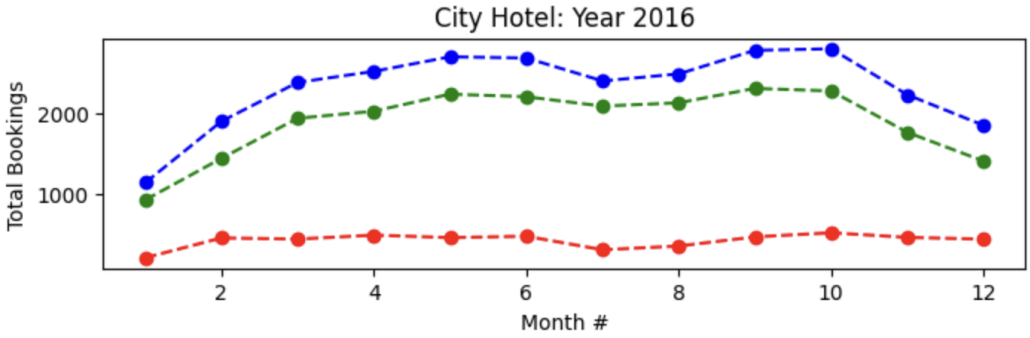
1. arrival_date_year: The data collected across years is for different months for different years, so we plotted the average bookings per month instead of total bookings.

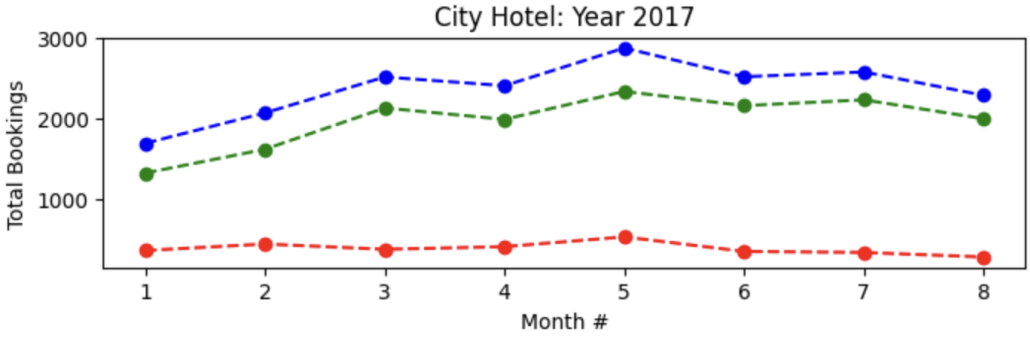


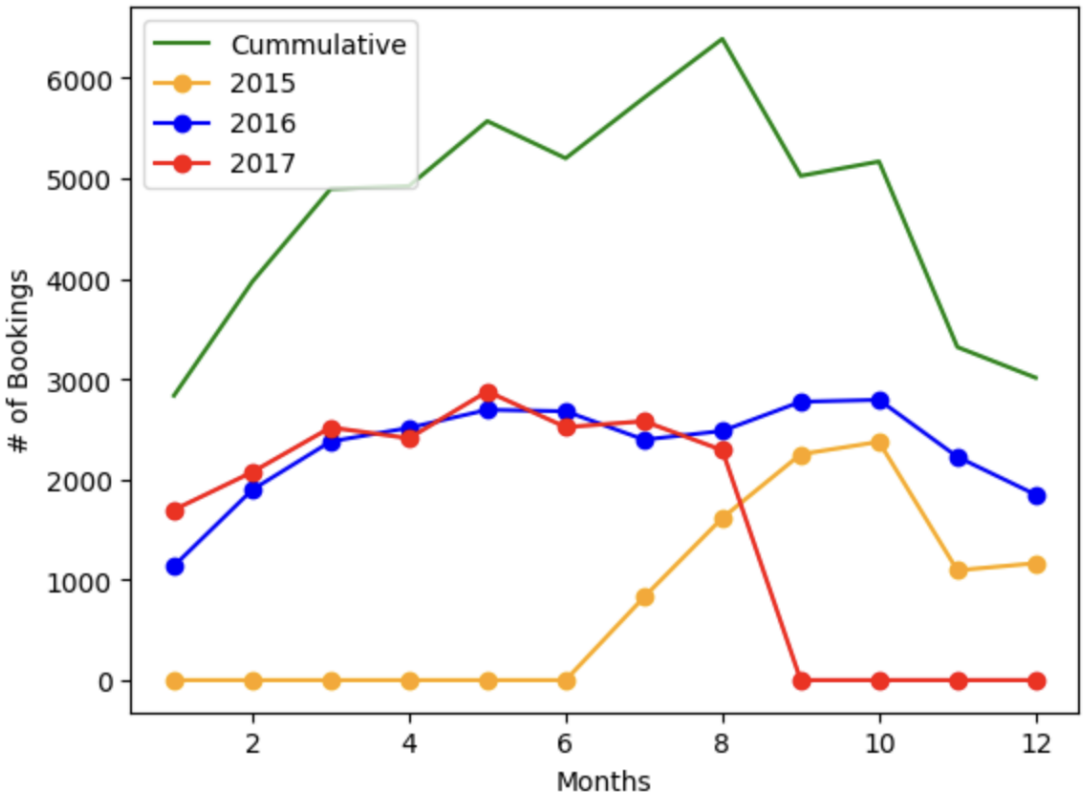

2. arrival_date_month: The first four figures suggest that demand is **relatively high** for months April through Oct. The fifth figure suggests that the demand also stays **relatively stable** across years for all three years. Therefore, we limit our analysis to these months in the case of City Hotel.


City Hotel: Overall
total
check-ins
no shows
# of Bookings
6000
4000
2000
2
4
6
8
10
12
Month

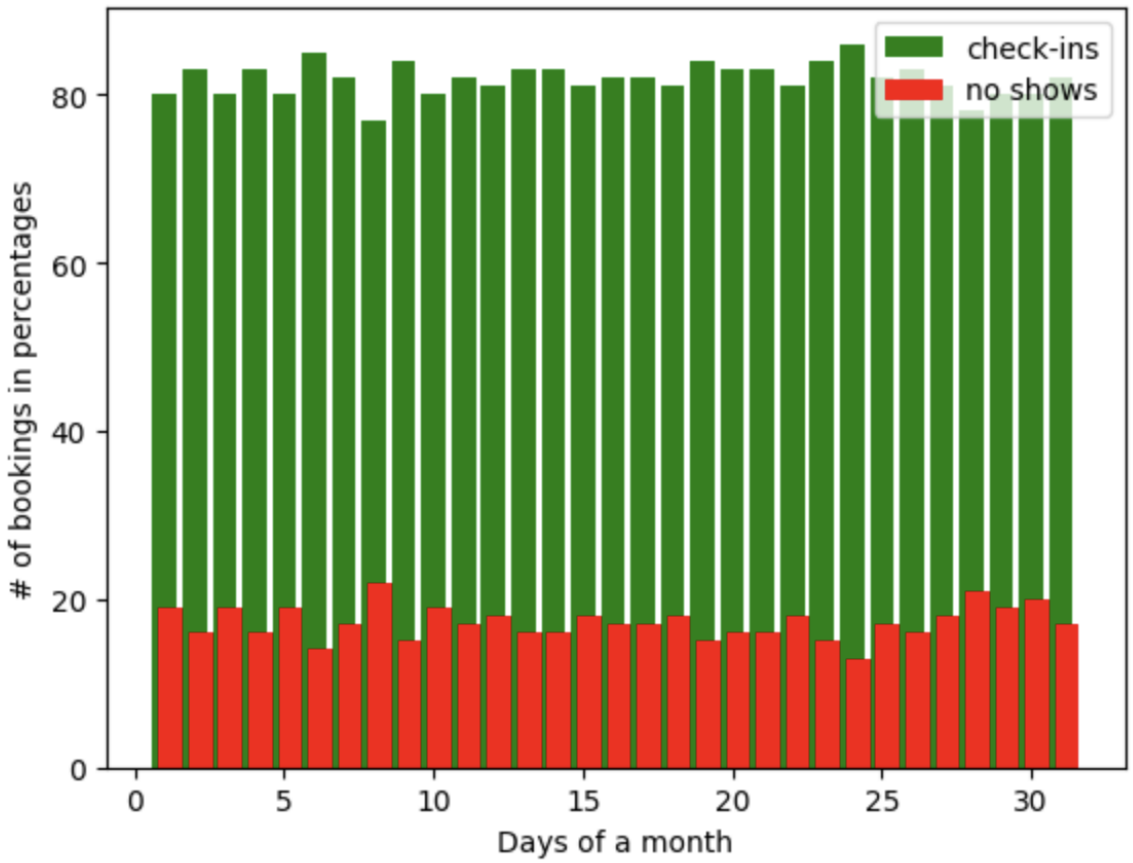

City Hotel: Year 2015
Total Bookings
2000
1500
1000
500
7
8
9
10
11
12
Month #


City Hotel: Year 2016
Total Bookings
2000
1000
2
4
6
8
10
12
Month #

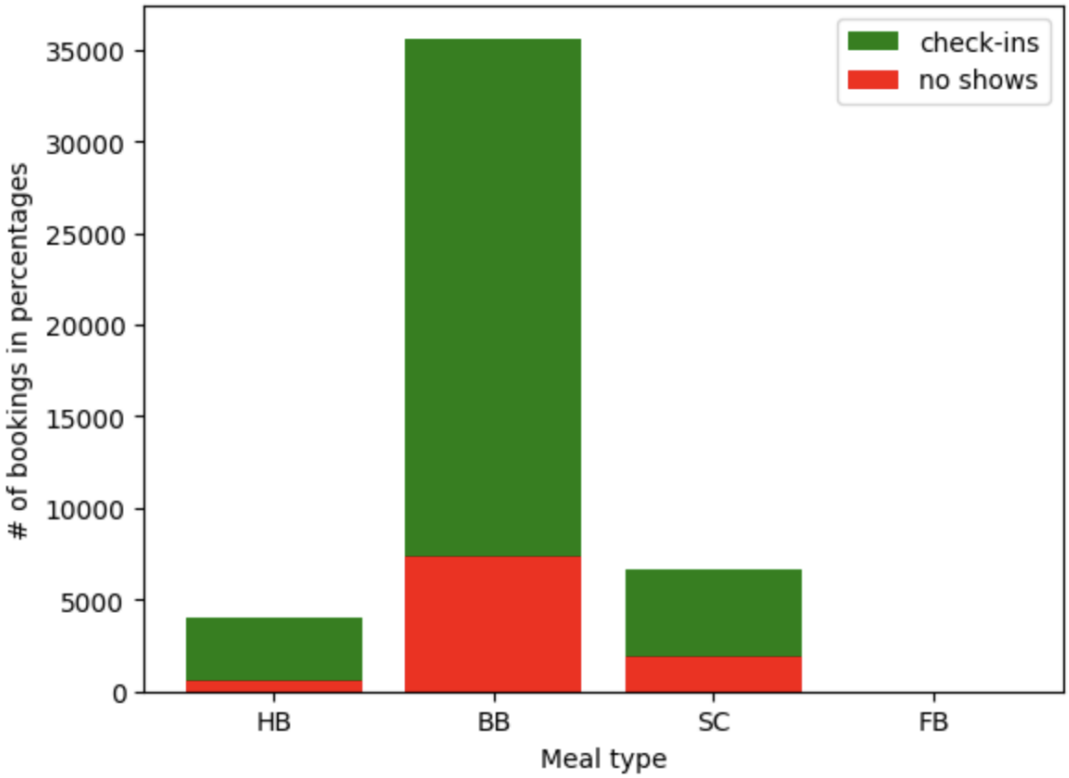

City Hotel: Year 2017
Total Bookings
3000
2000
1000
1
2
3
4
5
6
7
8
Month #

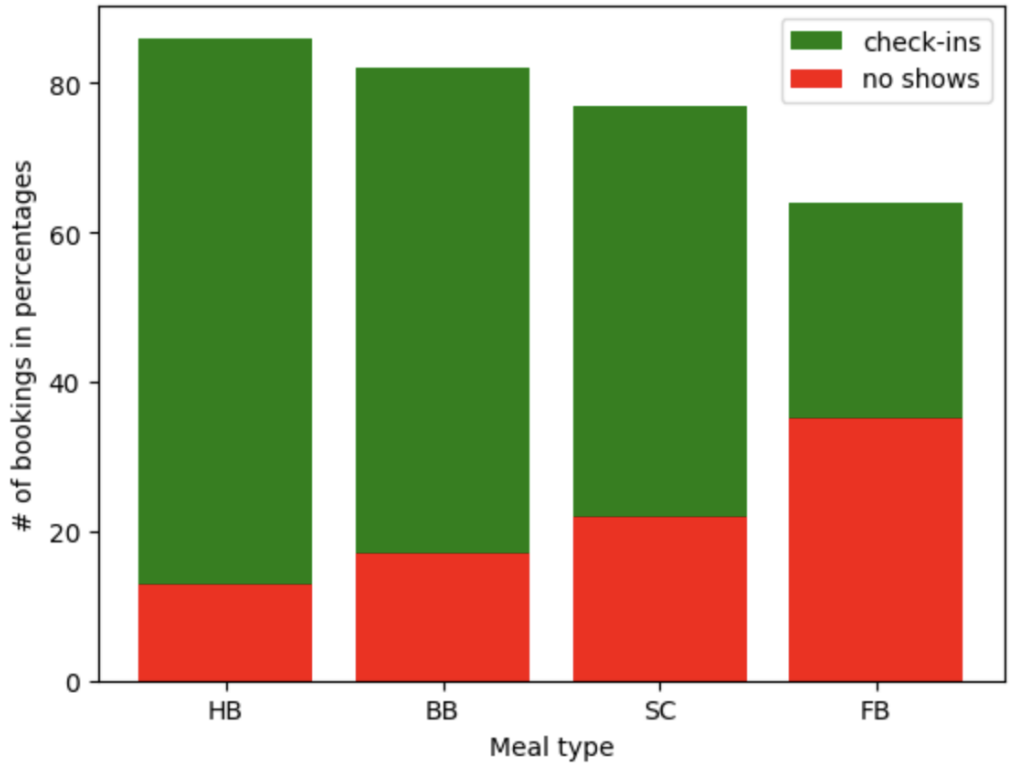



3. arrival_day_of_month: The data collected across days of the month are more or less similar for both check-ins and no-shows (i.e., check-ins for first day of the month is around 80%, check-ins for second day of the month is around 83% and so on).

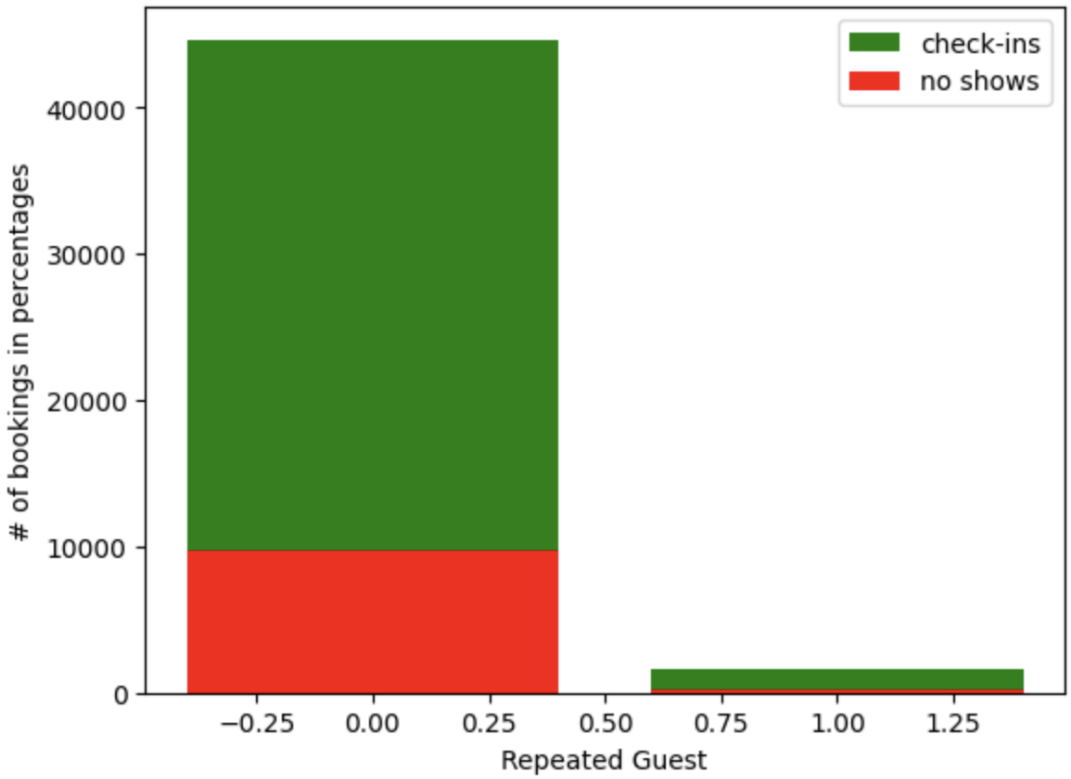


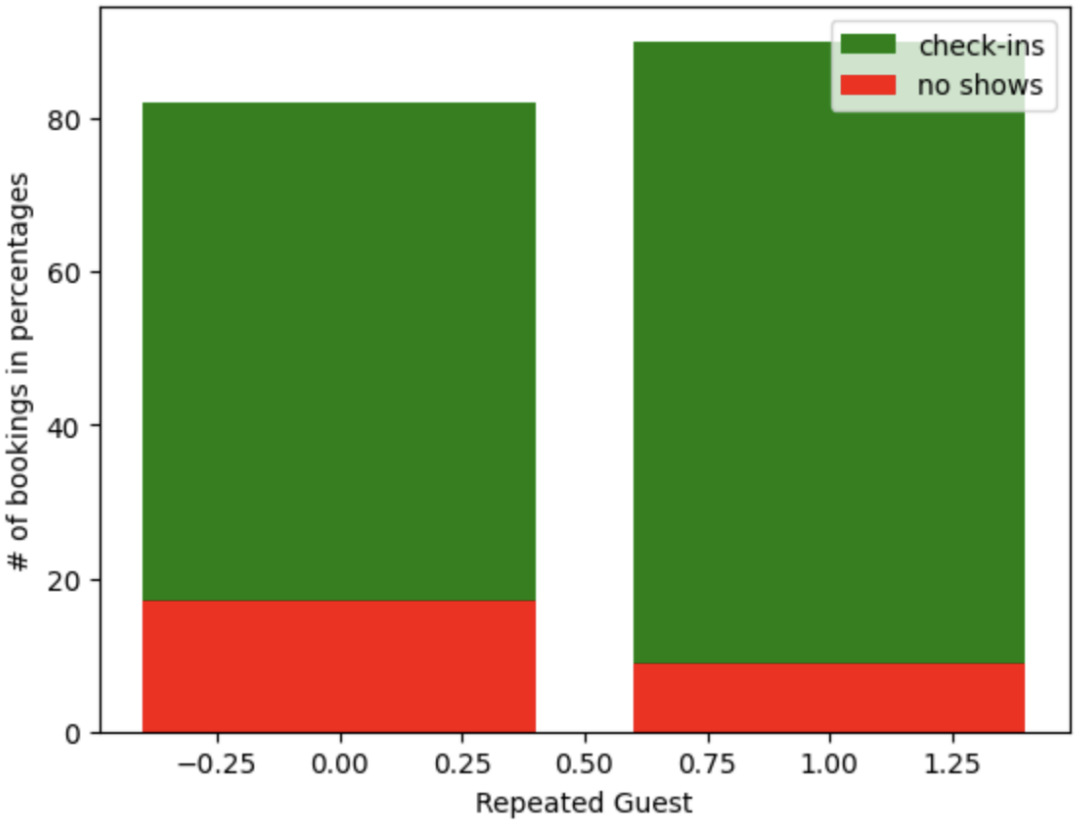

4. meal_type: The second figure suggests that there are slightly higher cancellations for meal_type = “FB” but the first figure shows that this meal_type corresponds to a very small portion of the total bookings compared to others and therefore no useful insight can truly be drawn about it. We defer to modeling in this case and do not make any data processing decisions at this point.


check-ins
no shows
35000
30000
25000
20000
15000
10000
5000
0
# of bookings in percentages
HB
BB
SC
FB
Meal type


check-ins
no shows
80
60
40
20
0
# of bookings in percentages
HB
BB
SC
FB
Meal type

5. is_repeated_guest: 1 represents yes, 0 represents no.

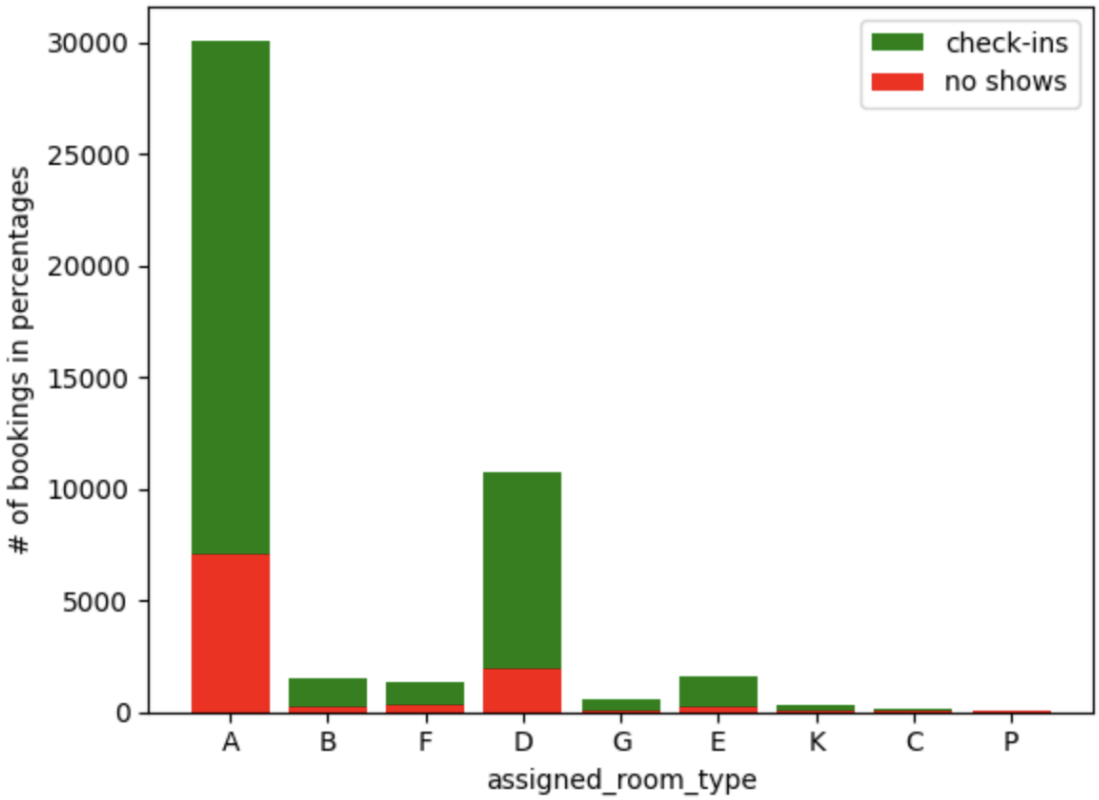


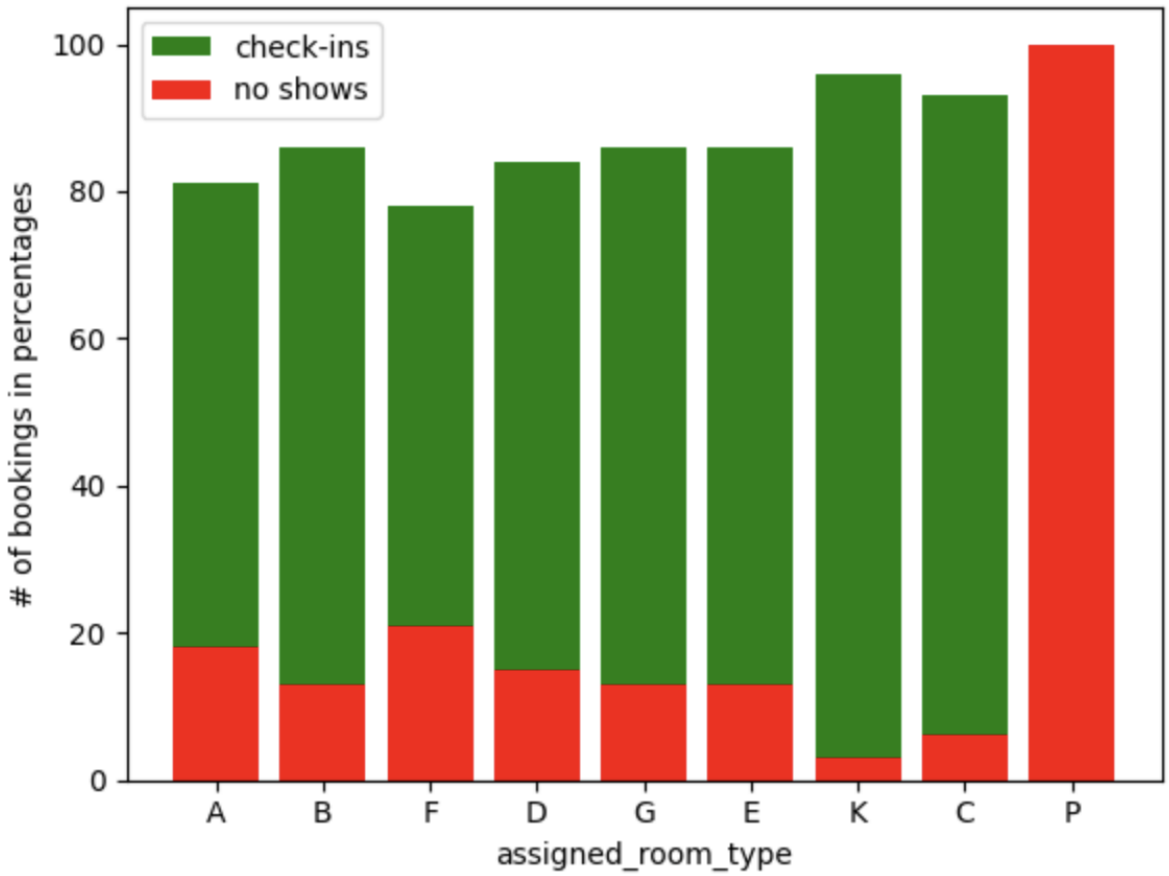



6. country: This feature has 166 unique values.

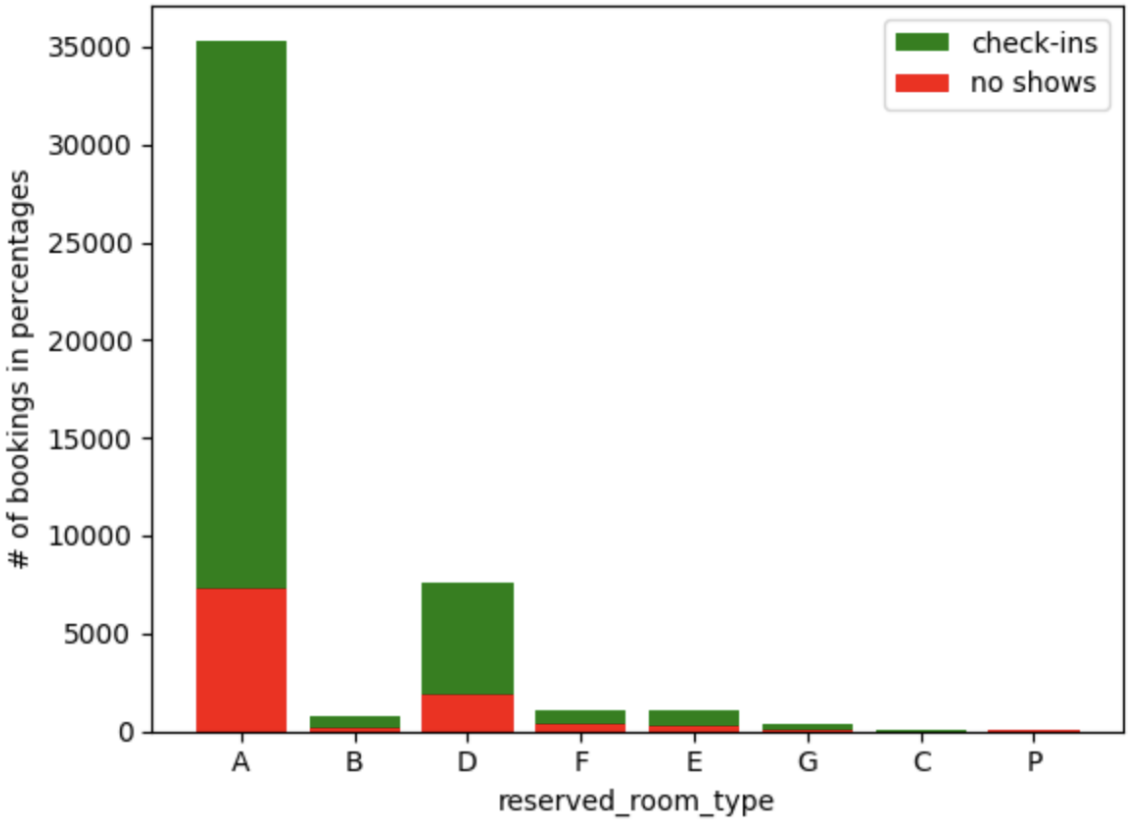
```
Bookings from this country are honored 100%:
['STP', 'CMR', 'KNA', 'JOR', 'LKA', 'IRQ', 'RWA', 'CRI', 'MMR', 'DOM', 'PAN', 'BFA', 'MCO', 'LBY', 'MLI', 'BHR', 'N
AM', 'BOL', 'JAM', 'SYC', 'PRY', 'BRB', 'ABW', 'AIA', 'SLV', 'DMA', 'CUB', 'PYF', 'UGA', 'GUY', 'LCA', 'ATA', 'MKD'
, 'MNE', 'GTM', 'GHA', 'ASM', 'SYR', 'TGO', 'SUR', 'MRT', 'CAF', 'NCL', 'KIR', 'SDN', 'ATF', 'SLE', 'LAO', 'COM']
Bookings from this country are cancelled 100%:
['AND', 'HND', 'ZMB', 'KHM', 'NIC', 'GIB', 'MAC', 'UMI', 'JEY']
```

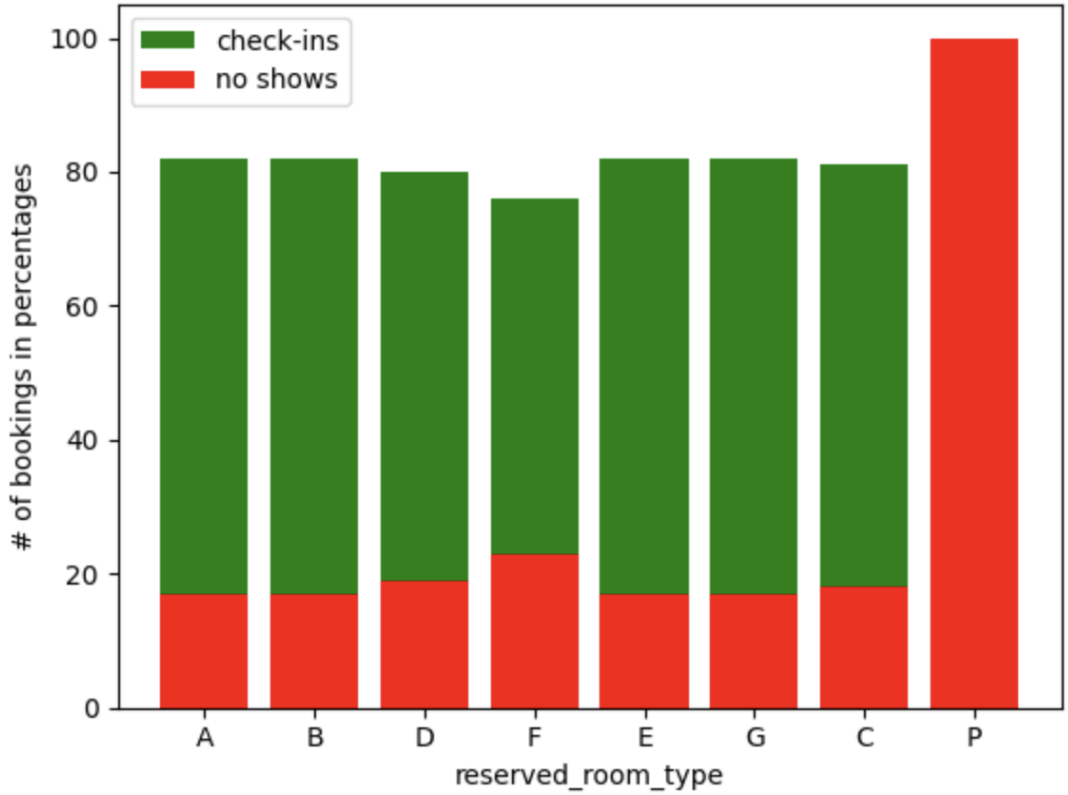
7. assigned_room_type: “P” room type seems to have 100% no-shows. However, it comprises a very small portion of the data.

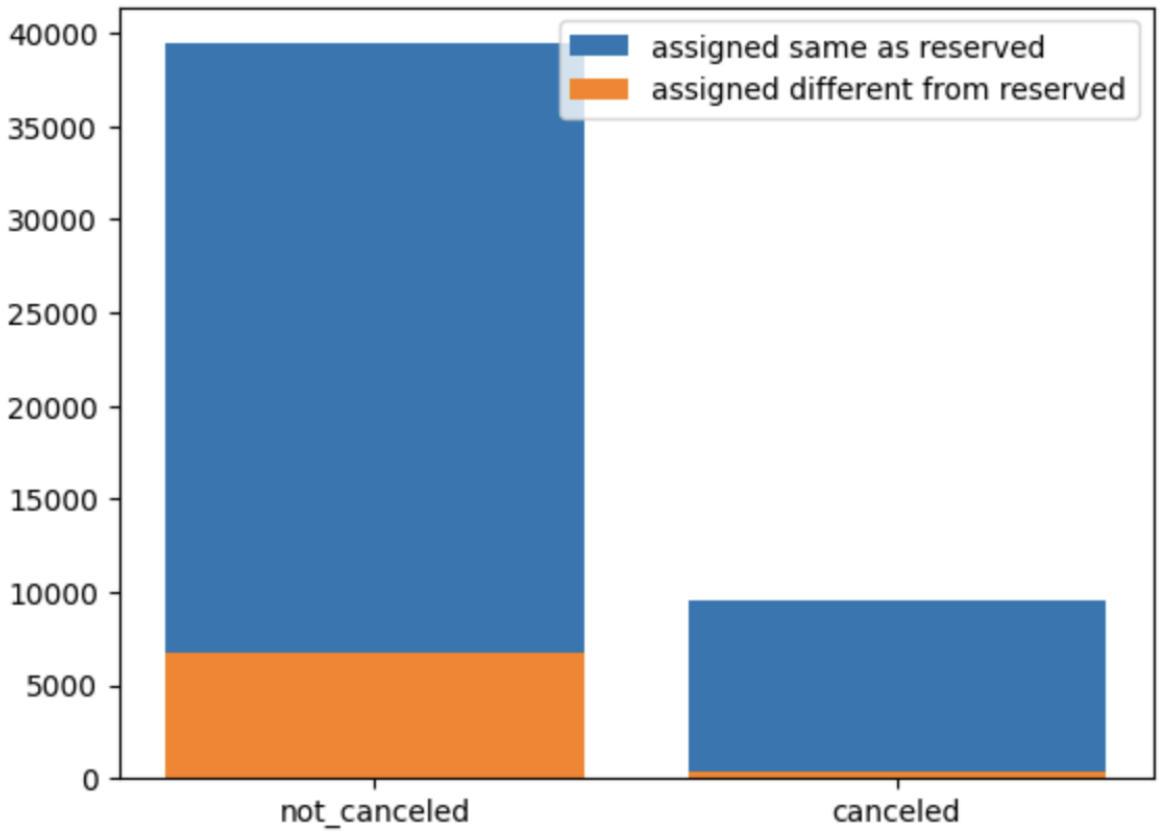

check-ins
no shows
30000
25000
20000
15000
10000
5000
0
# of bookings in percentages
A
B
F
D
G
E
K
C
P
assigned_room_type




8. reserved_room_type: “P” room type seems to have 100% no-shows. However, it comprises a very small portion of the data.

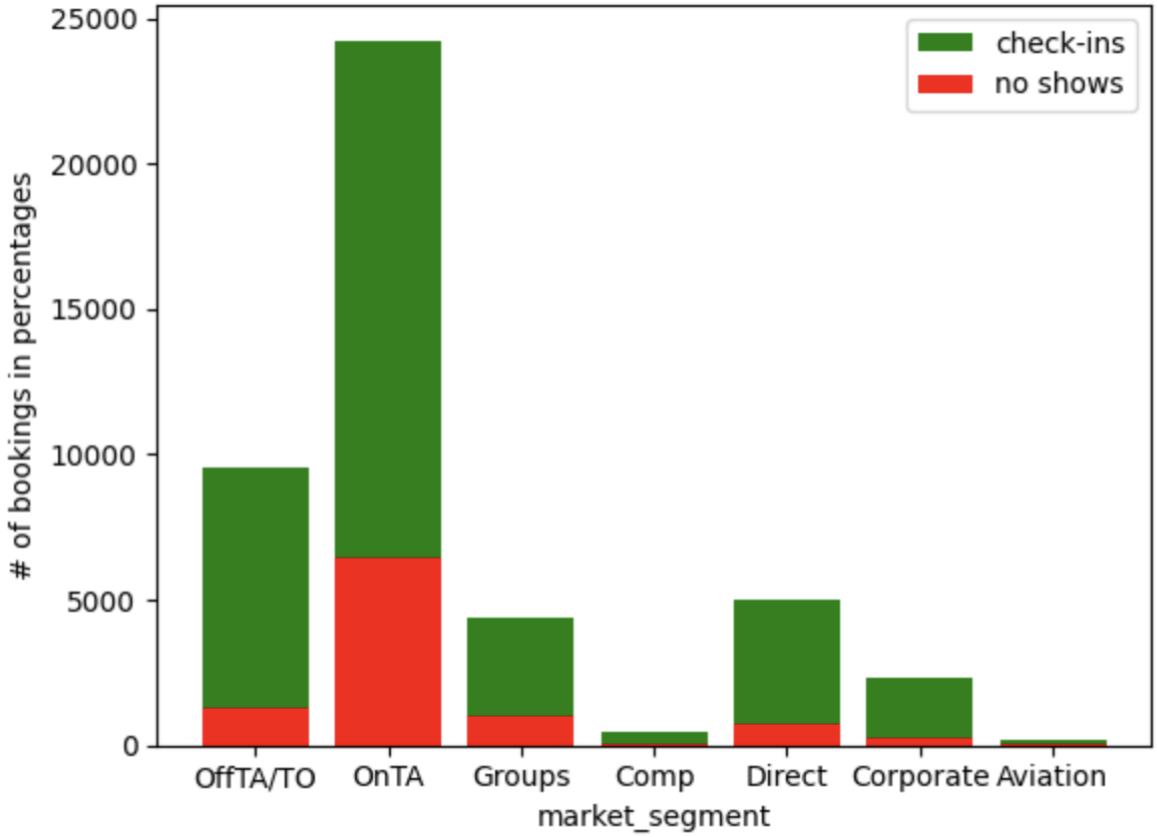

check-ins
no shows
35000
30000
25000
20000
15000
10000
5000
0
# of bookings in percentages
A
B
D
F
E
G
C
P
reserved_room_type

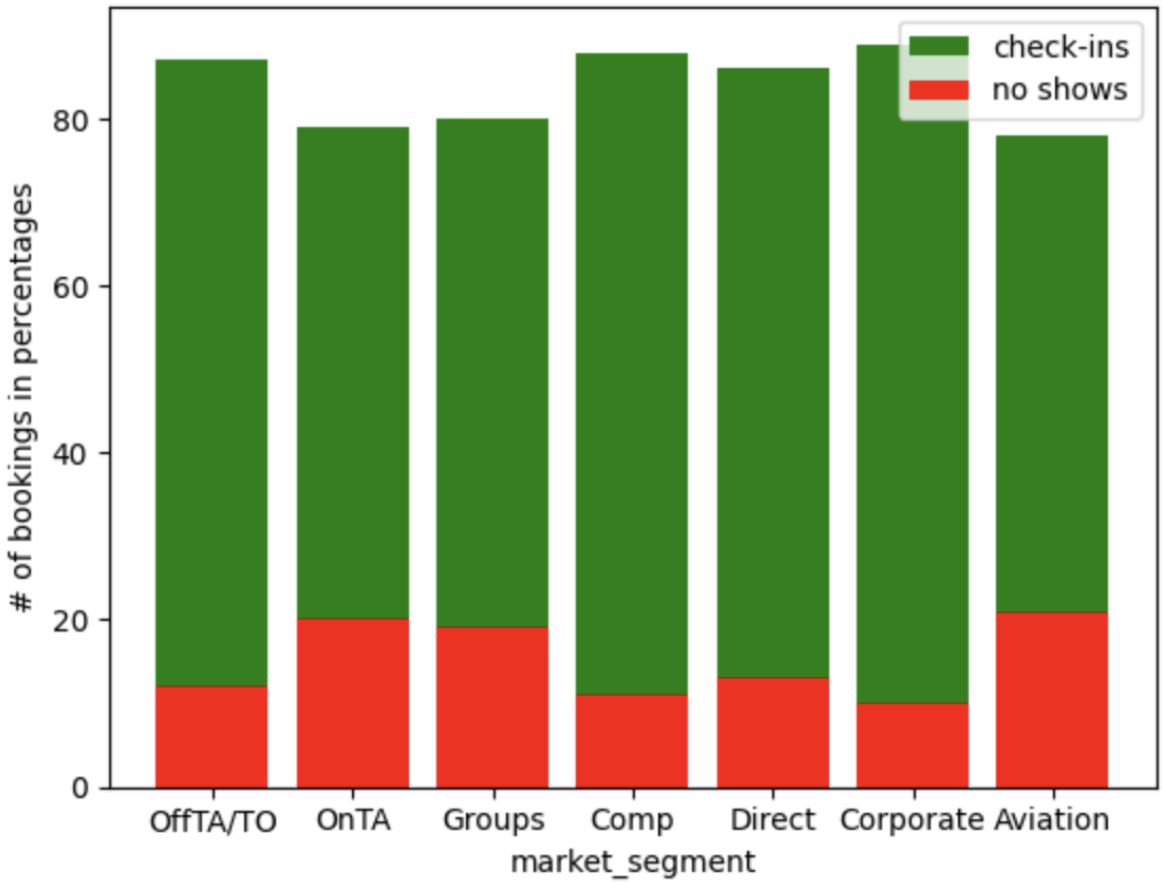

check-ins
no shows
100
80
60
40
20
0
# of bookings in percentages
A
B
D
F
E
G
C
P
reserved_room_type



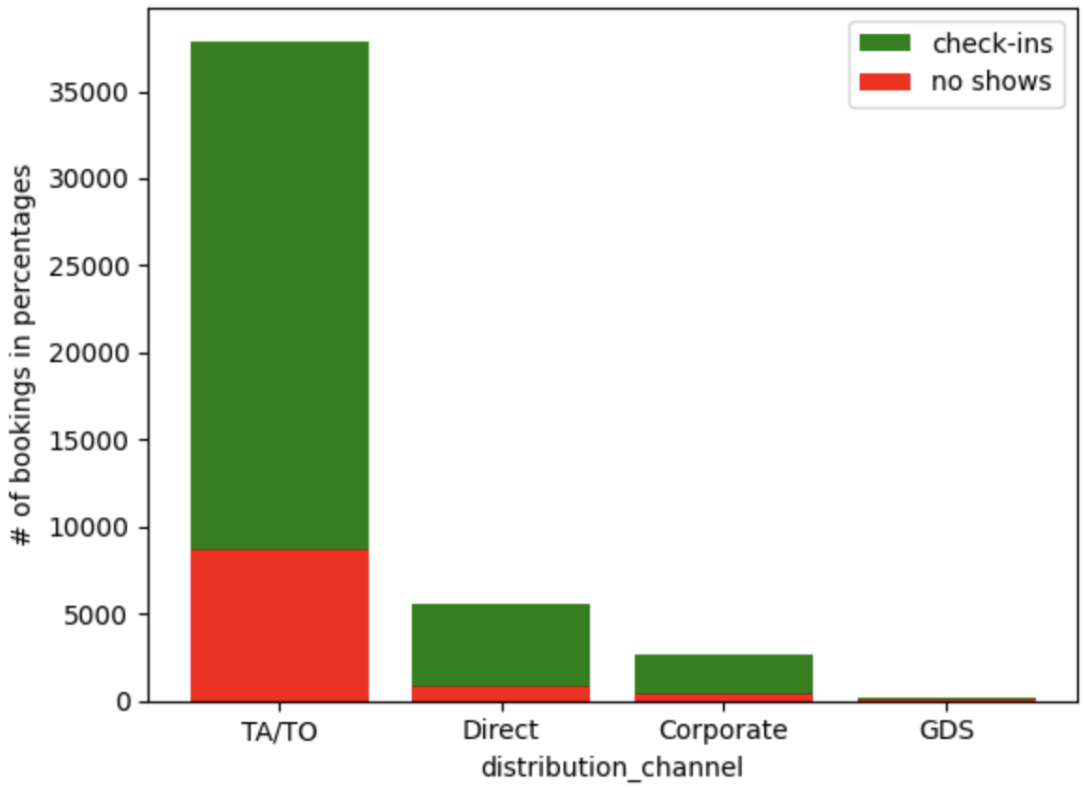

NOTE: While we believed that not getting the reserved room type assigned may influence customer no-shows, the data seems to suggest that the difference does not matter very much.

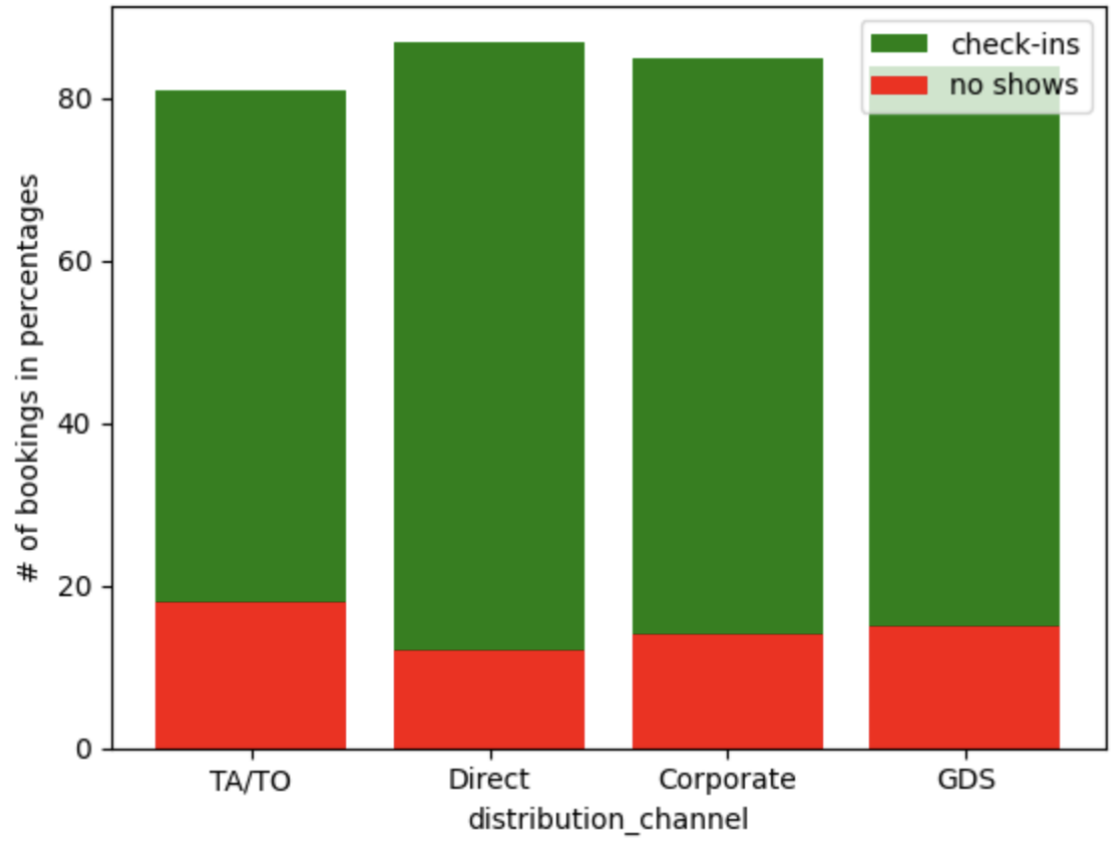
9. market_segment:



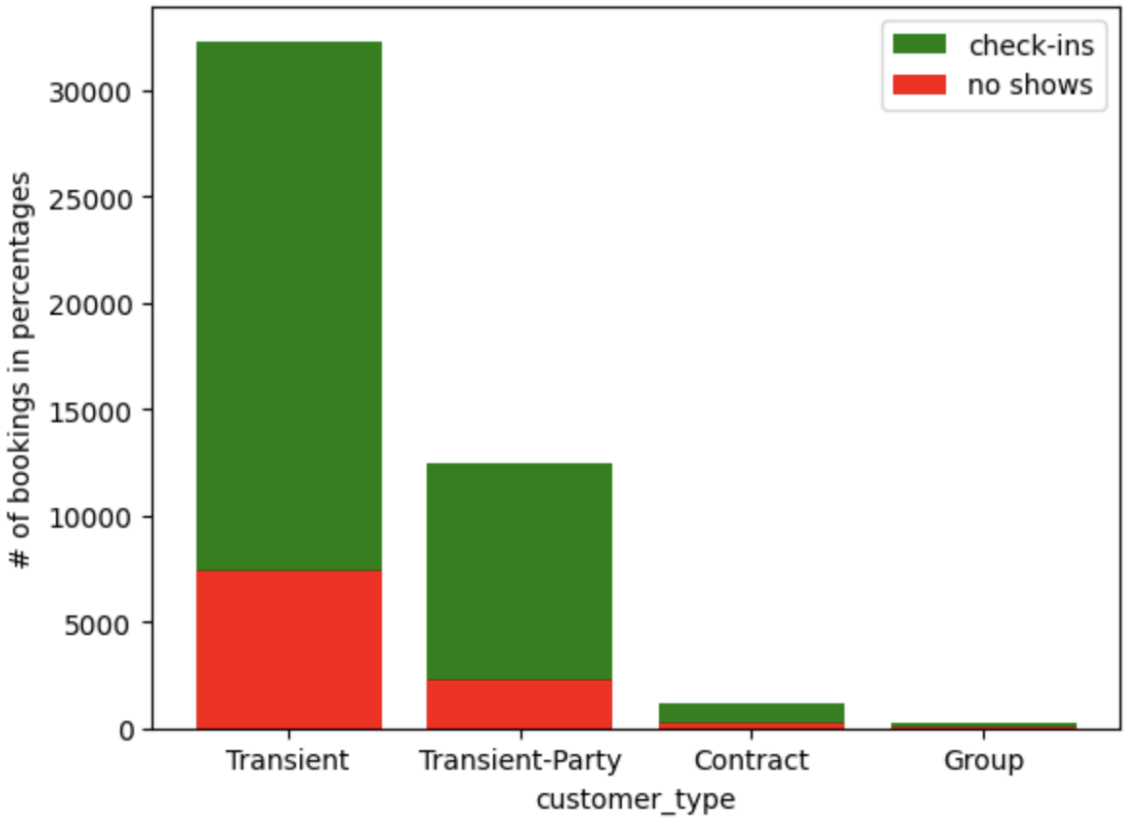



10. distribution_channel:


check-ins
no shows
35000
30000
25000
20000
15000
10000
5000
0
# of bookings in percentages
TA/TO
Direct
Corporate
GDS
distribution_channel

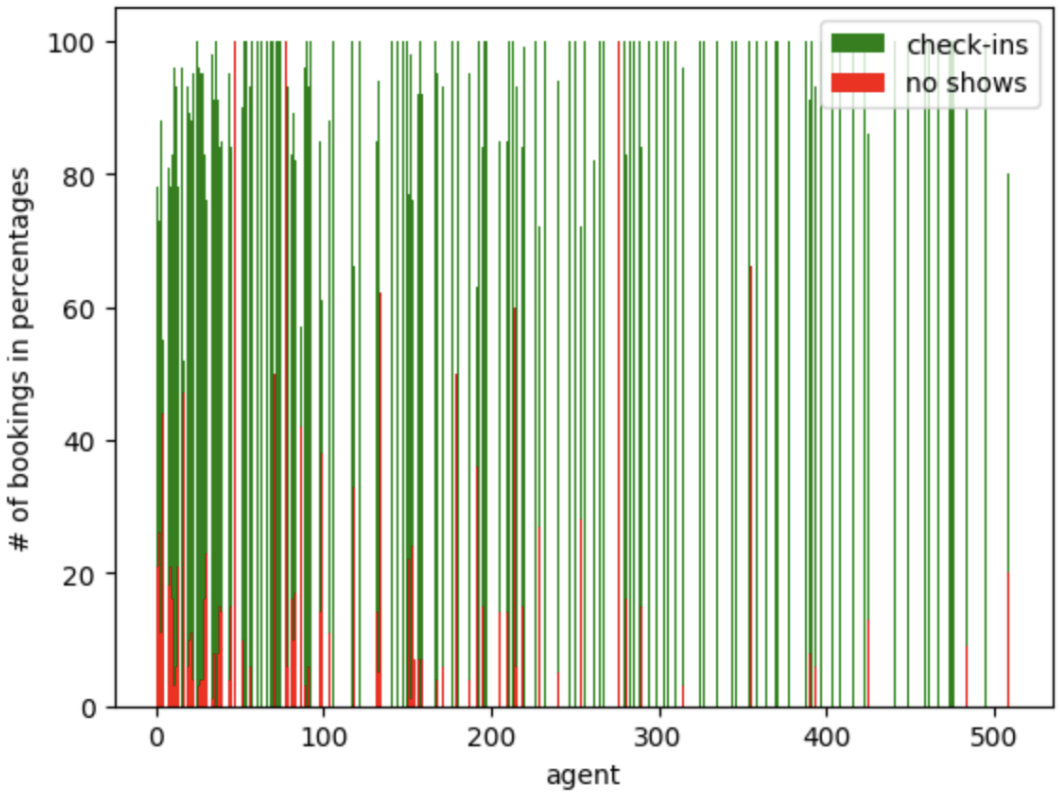

check-ins
no shows
80
60
40
20
0
# of bookings in percentages
TA/TO
Direct
Corporate
GDS
distribution_channel

11. customer_type:

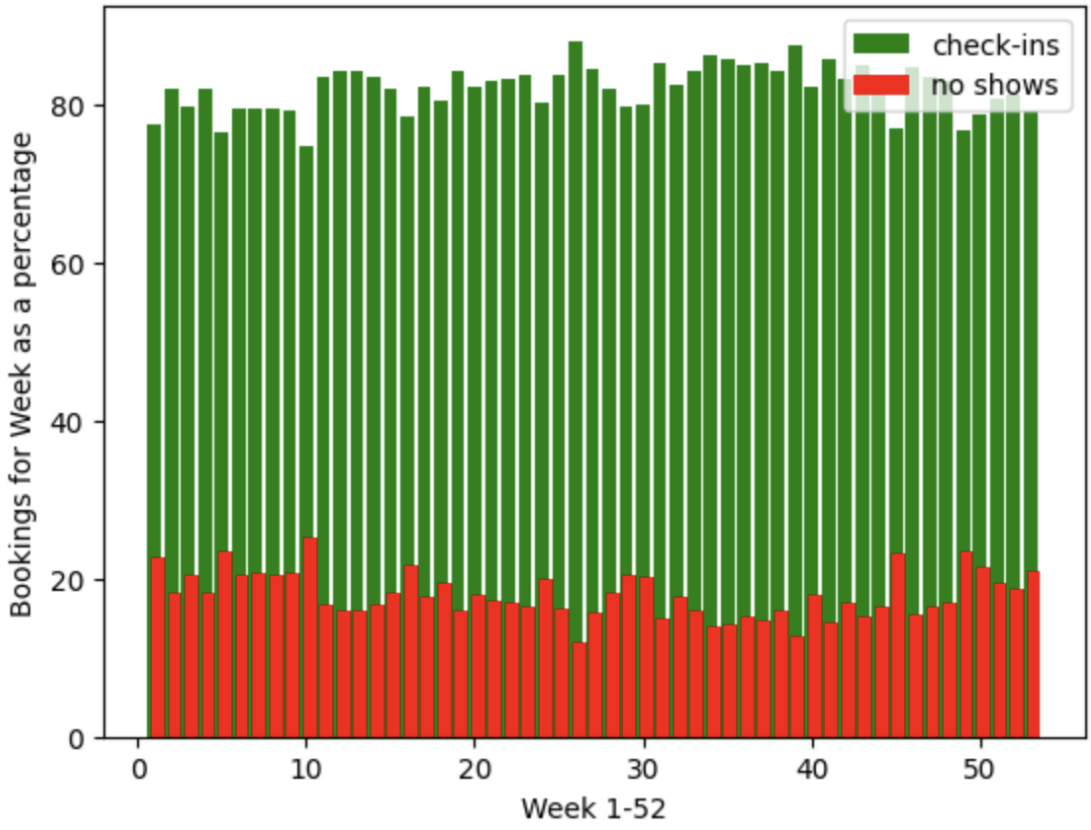



check-ins
no shows
# of bookings in percentages
0
20
40
60
80
Transient
Transient-Party
Contract
Group
customer_type

12. agent:




12. arrival_date_week:




# Modeling

Before we model our data, we will use some of the insights from our analysis in order to transform the dataset.

1. We only consider bookings from the months April through Oct due to high and stable demand during these months.

2. We drop columns “reservation_status_date” and “deposit_type”:

    a. “reservation_status_date”: model chosen cannot handle date type attributes.
    b. “deposit_type”: “Non Refund” instances are dropped from our dataset since it does not pose a loss to the business, which is what we are trying to mitigate in our business problem. Assuming that 100% of the amount paid is refunded, the “Refundable” case reduces to “No deposit” case, in which the business suffers 100% of the room cost as a loss in case of no-show.

3. **LEAKAGE**: As mentioned above, we assume that the business will consider the model recommendations <=30 days out from the arrival_date. This means that for all three cases “Check-Outs”, “No Shows” and “Cancellations”, we will not know the status of the reservation and including this feature will result in leakage.
4. Dummy variables are created for categorical variables using One-Hot Encoding method.
5. After this, the dataset consists of:

```
0    25399
1     5065
Name: is_canceled, dtype: int64
```

6. Imbalanced Dataset: Using oversampling technique SMOTE, we bring the minority class of No Shows to have at least 80% of the majority class of Check-Ins instances.

```
0    25399
1    20319
Name: is_canceled, dtype: int64
```

7. Dataset was split into training (80%) and test (20%).
8. Based on literature review[2], we modeled our data using Logistic Regression with Grid Search CV over hyperparameters by varying regularization - C, penalty.
9. The model produces probability estimates with an **AUC Score of 82%** against a base rate of 0.5, which we use as a “baseline”.

# Decision Logic

We leverage the expected value framework in order to find the optimal overbooking limit that maximizes revenue. This takes into consideration both “underbooked” and “overbooked” scenarios to decide the optimal overbooking limit (over capacity) that yields the best revenue. In order to account for both scenarios, we make assumptions about the penalty in the overbooked scenario i.e., overbooked_penalty = 130% of the room cost. This is based on how businesses compensate customers that showed up but were not accommodated in the hotel by either setting them up in a better hotel, or rescheduling them with complimentary services.

EXPECTED VALUE FRAMEWORK:

| | |
|---|---|
| CAPACITY | c |
| OPPORTUNITY COST | ‘Adr’ of booking |
| OVERBOOKED COST | 130 % ‘Adr’ of booking |
| REVENUE | ‘Adr’ of booking |
| PROBABILITY ESTIMATES | Produced by model |

Additionally, we perform this analysis at the week level i.e., for a given week, the decision logic recommends the overbooking limit that will maximize revenue for that week. We also assumed that all bookings were made at the same time.

We consider the “Week 33” in our dataset as it has the highest number of bookings for our decision logic.

```
Out[1932]: arrival_date_week_number_33    330
           arrival_date_week_number_34    282
           arrival_date_week_number_21    282
           arrival_date_week_number_27    280
           arrival_date_week_number_28    277
           arrival_date_week_number_39    273
           arrival_date_week_number_30    273
           arrival_date_week_number_32    271
           arrival_date_week_number_41    254
           arrival_date_week_number_31    253
           arrival_date_week_number_42    251
           arrival_date_week_number_29    250
           arrival_date_week_number_20    247
           arrival_date_week_number_26    245
           arrival_date_week_number_17    245
           arrival_date_week_number_18    245
           arrival_date_week_number_38    245
           arrival_date_week_number_24    245
           arrival_date_week_number_35    241
           arrival_date_week_number_15    237
           arrival_date_week_number_23    235
           arrival_date_week_number_44    233
           arrival_date_week_number_19    228
           arrival_date_week_number_22    228
           arrival_date_week_number_25    225
           arrival_date_week_number_40    224
           arrival_date_week_number_16    222
```

UNDERBOOKED SCENARIO:

In this scenario, the total check-ins are less than capacity and the expected value for business is calculated as below:

1. In our decision logic algorithm, we vary the check-ins from 0 to C-1 and calculate the expected value.
2. EXPECTED VALUE
   WHILE check-ins VARY FROM 0 TO C-1:
   a. Value generated from each checked-in customer:
   P(check-in of customer) * (Revenue from customer booking)
   This is summed up over all checked-in customers

b. Loss generated from each no-show customer:
P(no-show of customer) * (Opportunity cost from customer booking)
This is summed up over all no-show customers

OVERBOOKED SCENARIO:

In this scenario, the total check-ins are more than capacity and the expected value for business is calculated as below:

3. In our decision logic algorithm, we vary the check-ins from C+1 to total_bookings and calculate the expected value.
4. EXPECTED VALUE
WHILE check-ins VARY FROM C+1 TO total_bookings:
a. Value generated from each checked-in customer:
P(check-in of customer) * (Revenue from customer booking)
This is summed up over all checked-in customers
In this case **the value generated is capped at Capacity**

b. Loss generated from each denied customer that showed-up:
P(check-in of customer) * (Overbooking cost from customer booking)
This is summed up over all no-show customers

For the purposes of this project, we assumed that customers that checked-in did so in the order that the data is (more on this in the limitations section!).

Based on our modeling estimates for week 33, we assumed C = 300; total_booking = 330 and overbooking_cost = 130%room_cost or 'adr'.

FINDINGS:

We observed that expected value steadily increases as customers check-in and the highest expected value in UNDERBOOKED SCENARIO is

```
print(max(ev))
print(len(ev))

#28831.75973929873
# 300
```

```
28735.75973929873
299
```

Revenue for booking at capacity is 28831.759

Expected value increases even more after overbooking by 1 more than the capacity (C+1) and then starts to decrease as overbooking costs start to influence value and the highest expected value in OVERBOOKED SCENARIO is

```
print(max(ev_ob))
print(len(ev_ob))
```

```
28850.088277366947
```

Therefore, the decision logic recommends that the hotel should overbook by 1 more than capacity for the week of 33 in order to maximize value.

## LIMITATIONS OF OUR STUDY

- Perform correlation analysis with respect to target variable. In the future, we recommend leveraging covariance matrix to get numeric features that are highly correlated with the target and performing modeling on those.
- Perform sequential feature selection using wrapper methods such as RandomForestClassifier, etc in order to get feature importance of categorical variables. This would also help with interpretability in the future.
- Logistic Regression: The choice of the model was based on literature review of papers that aimed to solve similar problems. Also, it provided us with probability forecasts of no-shows which could be easily fed into our EV framework. However, in the future, with more resources of time and computational power, we recommend other machine learning techniques such as RandomForests, SVM and other such methods to choose the one that best serves our problem.
- Additionally, in gradient-descent based methods, it is important to scale numeric features, which was not done in our analysis. We would recommend scaling

features like “lead_time” etc in the future in order to reduce the impact of outliers in our dataset.

- Interpretability of our model is limited.
- In our decision logic, we derive the overbooking limit on the week instead of the date in the order that the booking appears in the dataset. Ideally, we should do this at the day-level and order the check-ins based on check-in time (currently, this feature is not available in the dataset) and *then* roll-up to week, month and year.

# REFERENCES


1. https://www.sciencedirect.com/science/article/pii/S2352340918315191?via%3Dihub
2. https://www.sciencedirect.com/science/article/pii/S0360835223002504
3. https://www.kaggle.com/datasets/mojtaba142/hotel-booking?datasetId=1437463&sortBy=voteCount
4. https://ieeexplore.ieee.org/document/8260781
5. https://en.wikipedia.org/wiki/Overselling