Tittle of the Project

Hotel Booking Cancel Prediction

Submitted By,

Abhishek Prasad Nonia

21125892001

Business Understanding

A. Hotel booking cancellation impacts on the operation problem for the hotel.
B. There is a problem with overbooking for the hotels which lead to loss of sales and fall in customer loyalty
C. And customer is accustomed to free cancellation policies, which impact the loss of sales and fall in business reputation.

Problem Statement

A. In order to fight the negative effects of cancellation, hotels need to be able to identify which booking are likely to be cancelled.
B. Does Leadtime and Market segment related to booking cancellation?
C. Can we predict with accuracy if a booking will be cancelled based on the attributes?
D. What incentives we can provide to customers to reduce the booking cancellation?
E. How many days should the hotel keep between booking date and check in date for free cancellation?

Approach

The Hotel Booking Cancellation prediction will be forecasted using machine learning models and statistical techniques. Regression models like Linear, Logistic regression, Random Forest, boosting techniques will be used in forecast of the project. Jupyter Notebook will be used as a machine learning tool. These models will be compared with each other and which results into better accuracy of the model would be taken into consideration as resulting machine learning algorithm.

Data Understanding

The Dataset

data.csv

Checking The Data Information

```
In [76]:  1  hotel_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
```

```
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

These are columns attached to get a better understanding of the dataset

Summary of the Dataset

The dataset has **119390** rows and **32** columns in total. This csv files have rows representing numbers and values of the particular columns. The columns represent →

'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_statusus','reservation_status_date'.

Attribute Information

- Hotel: Hotel (H1 = Resort Hotel or H2 = City Hotel)
- is_canceled: Value indicating if the booking was cancelled (1) or not (0)
- lead_time: Number of days that elapsed between the entering date of the booking into the PMS and the arrival date
- arrival_date_year: Year of arrival date
- arrival_date_month: Month of arrival date
- arrival_date_week_number: Week number of year for arrival date
- arrival_date_day_of_month: Day of arrival date
- stays_in_weekend_nights: Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- stays_in_week_nights: Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- adults: Number of adults
- children: Number of children
- babies: Number of babies
- meal: 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)
- country: Country of origin. Categories are represented in the ISO 3155–3:2013 format
- market_segment: Market segment designation. In categories, the term "TA" means "Travel Agents" and "TO" means "Tour Operators"
- distribution_channel: Booking distribution channel. The term "TA" means "Travel Agents" and "TO" means "Tour Operators"
- is_repeated_guest: Value indicating if the booking name was from a repeated guest (1) or not (0)
- previous_cancellations: Number of previous bookings that were cancelled by the customer prior to the current booking
- previous_bookings_not_canceled: Number of previous bookings not cancelled by the customer prior to the current booking
- reserved_room_type: Code of room type reserved. Code is presented instead of designation for anonymity reasons
- assigned_room_type: 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

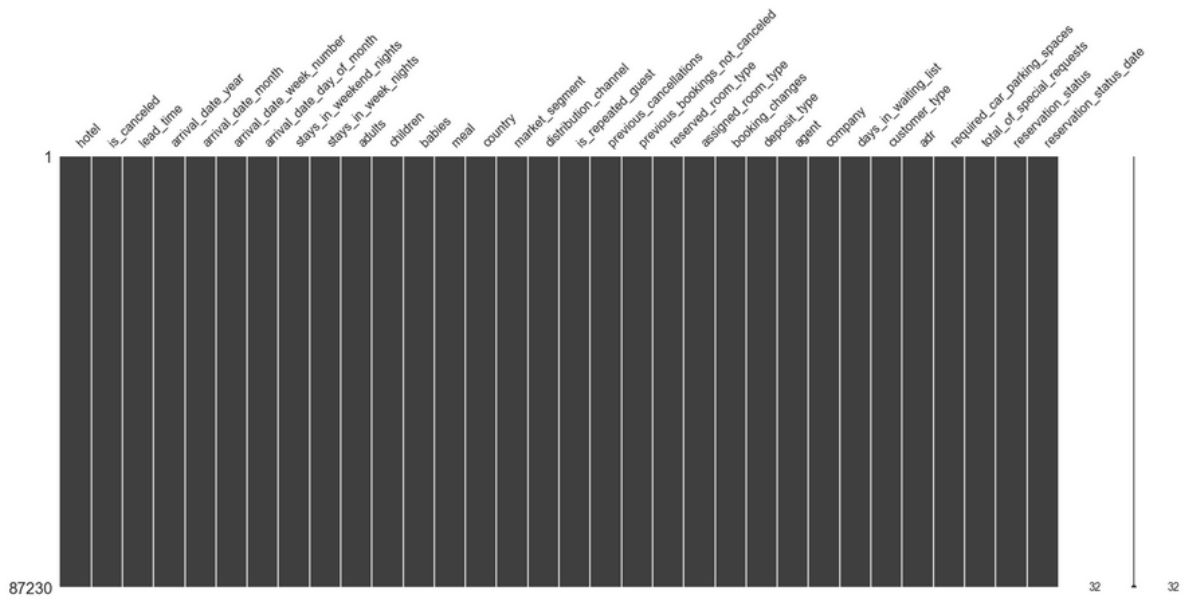
- booking_changes: Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
- deposit_type: Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories: No Deposit – 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
- agent: ID of the travel agency that made the booking
- company: ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons
- days_in_waiting_list: Number of days the booking was in the waiting list before it was confirmed to the customer
- customer_type: 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
- adr: Average Daily Rate as defined by dividing the sum of all lodging transactions by the total number of staying nights
- required_car_parking_spaces: Number of car parking spaces required by the customer
- total_of_special_requests: Number of special requests made by the customer (e.g. twin bed or high floor)
- reservation_status: Reservation last status, assuming one of three categories: Canceled – booking was cancelled 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
- reservation_status_date: Date at which the last status was set. This variable can be used in conjunction with the Reservation Status to understand when was the booking cancelled or when did the customer checked-out of the hotel

Exploratory Data Analysis

Missing Value Matrix

```
ms.matrix(hotel_data_clean)
```

<AxesSubplot:>



Checking categorical variables. ¶

```python
k,v=[],[]
for i in hotel_data_clean.select_dtypes('object').columns:
    k.append(i)
    v.append(list(hotel_data_clean[i].unique()))
categorydf=pd.DataFrame({'Category':k,'Sub-categoty':v})
print(categorydf)
```

```
                  Category                                        Sub-categoty
0                    hotel                           [Resort Hotel, City Hotel]
1       arrival_date_month  [July, August, September, October, November, D...
2                     meal                                    [BB, FB, HB, SC]
3                  country  [PRT, GBR, USA, ESP, IRL, FRA, Unknown, ROU, N...
4           market_segment  [Direct, Corporate, Online TA, Offline TA/TO, ...
5     distribution_channel          [Direct, Corporate, TA/TO, Undefined, GDS]
6       reserved_room_type                     [C, A, D, E, G, F, H, L, B]
7       assigned_room_type                  [C, A, D, E, G, F, I, B, H, L, K]
8             deposit_type              [No Deposit, Refundable, Non Refund]
9            customer_type           [Transient, Contract, Transient-Party, Group]
10      reservation_status                   [Check-Out, Canceled, No-Show]
11 reservation_status_date  [01-07-2015, 02-07-2015, 03-07-2015, 06-05-201...
```

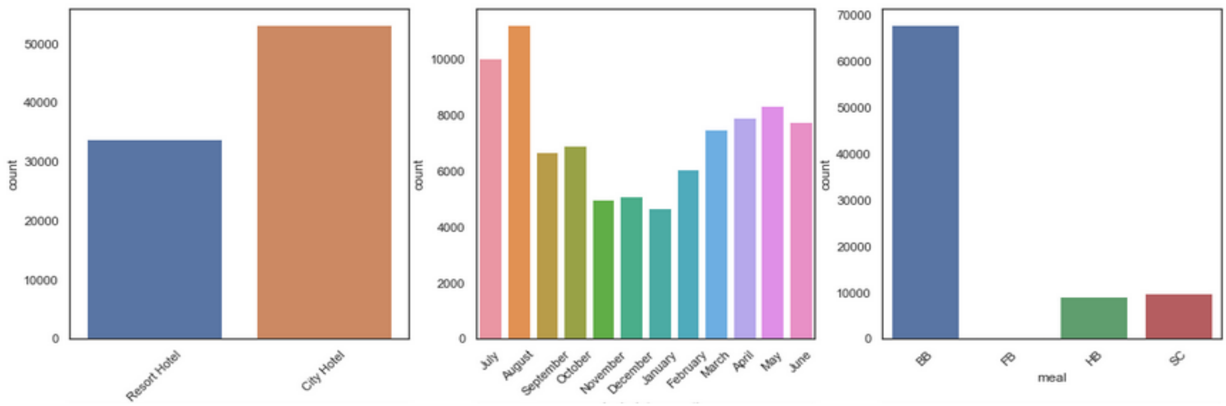
Exploratory Data Analysis

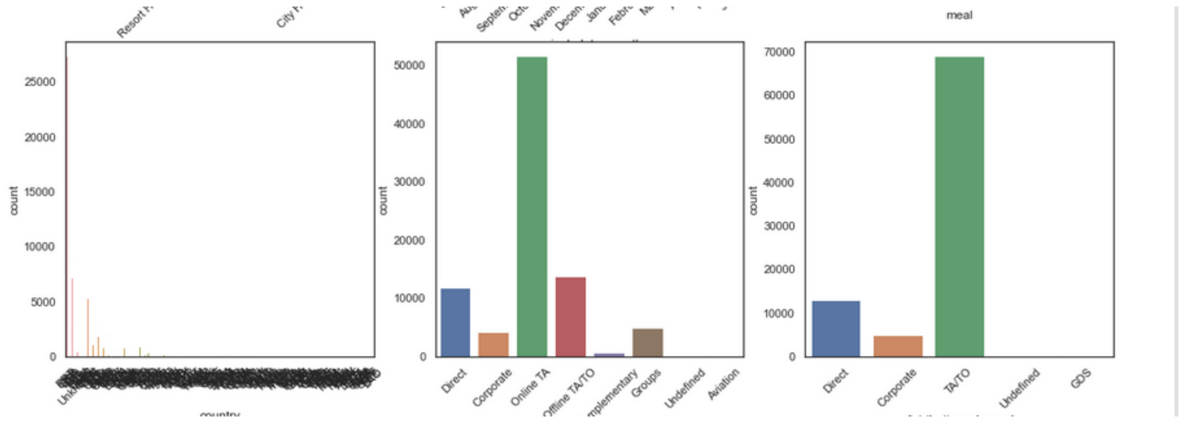
```
hotel_data_clean.select_dtypes('object').describe().T
```

	count	unique	top	freq
hotel	87230	2	City Hotel	53274
arrival_date_month	87230	12	August	11242
meal	87230	4	BB	67907
country	87230	178	PRT	27355
market_segment	87230	8	Online TA	51553
distribution_channel	87230	5	TA/TO	69028
reserved_room_type	87230	9	A	56436
assigned_room_type	87230	11	A	46283
deposit_type	87230	3	No Deposit	86085
customer_type	87230	4	Transient	71862
reservation_status	87230	3	Check-Out	63221
reservation_status_date	87230	926	14-02-2016	211

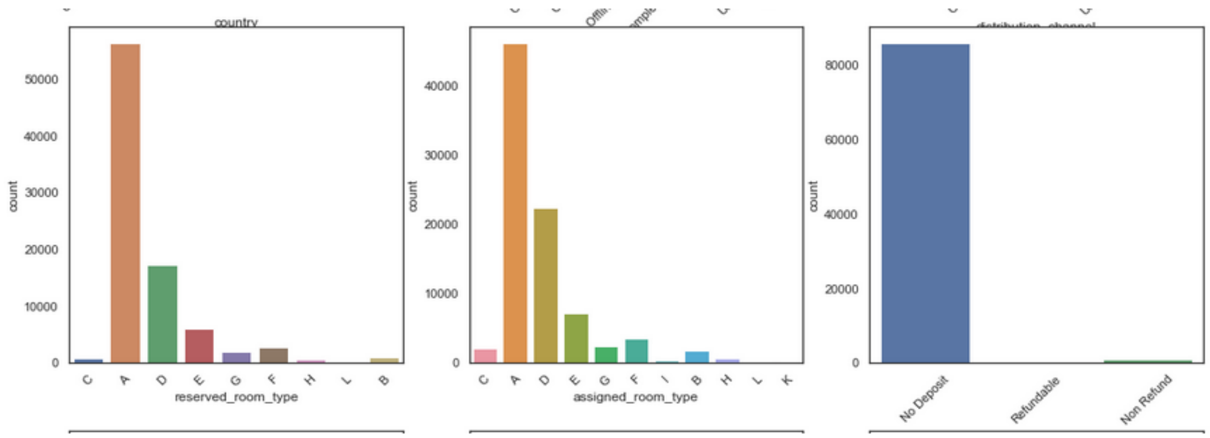
Plotting countplot of categorical variable to visualize distribution.

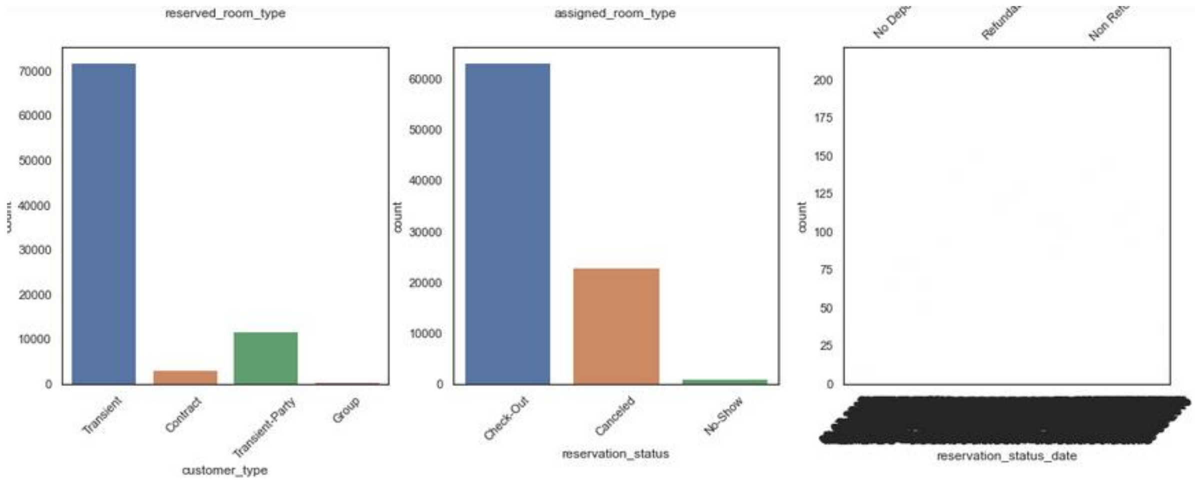
```
plt.figure(figsize=(18,80),facecolor='white')
plot_num=1
for i in k:
    ax=plt.subplot(12,3,plot_num)
    sns.countplot(x=i,data=hotel_data_clean)
    plt.xticks(rotation=45)
    plot_num+=1
plt.show()
```





Exploratory Data Analysis





Exploratory Data Analysis

Analyzing numerical variables.

4]:

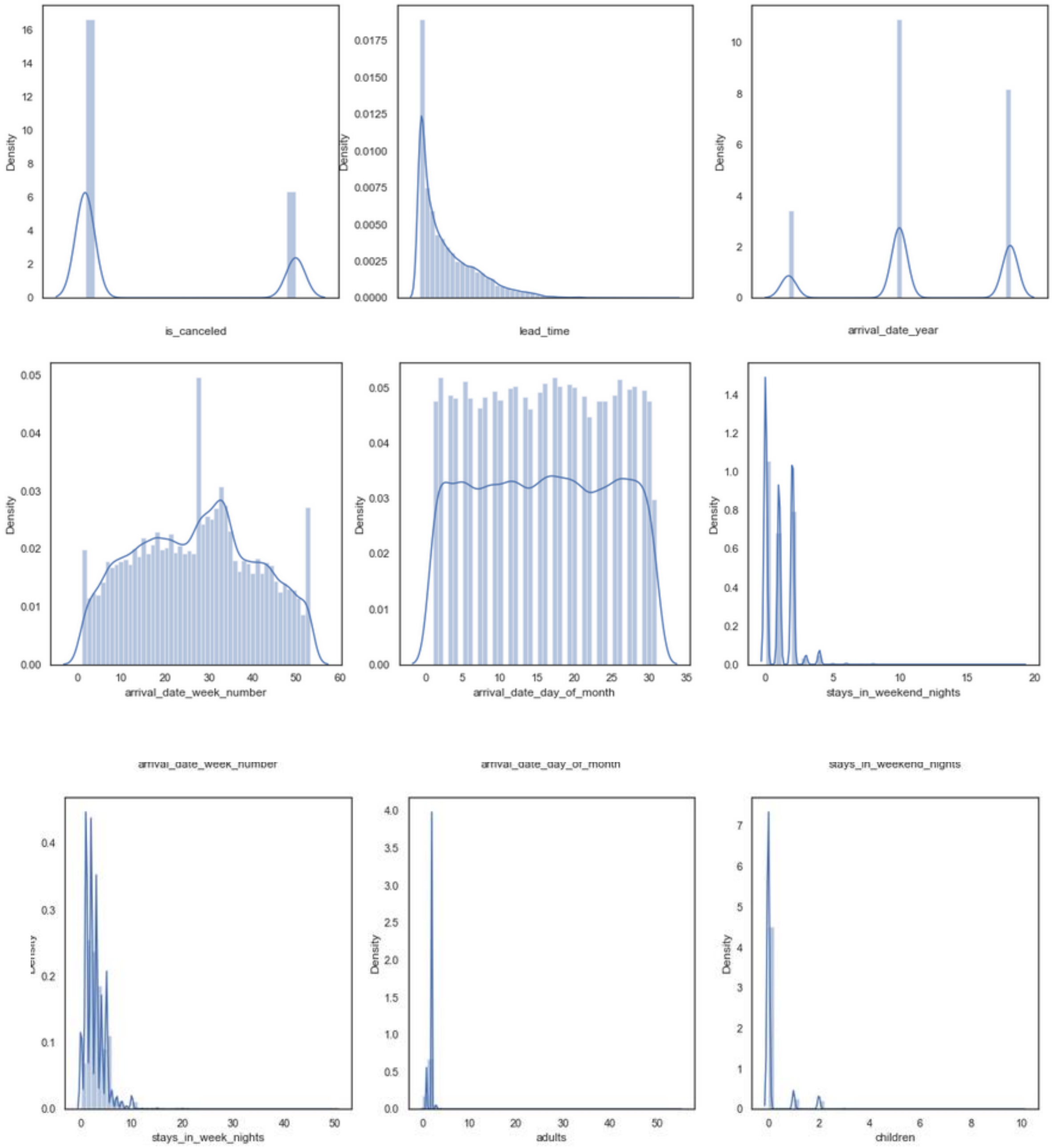
```python
k1=[]
for i in hotel_data_clean.select_dtypes(exclude='object').columns:
    k1.append(i)
```

5]:

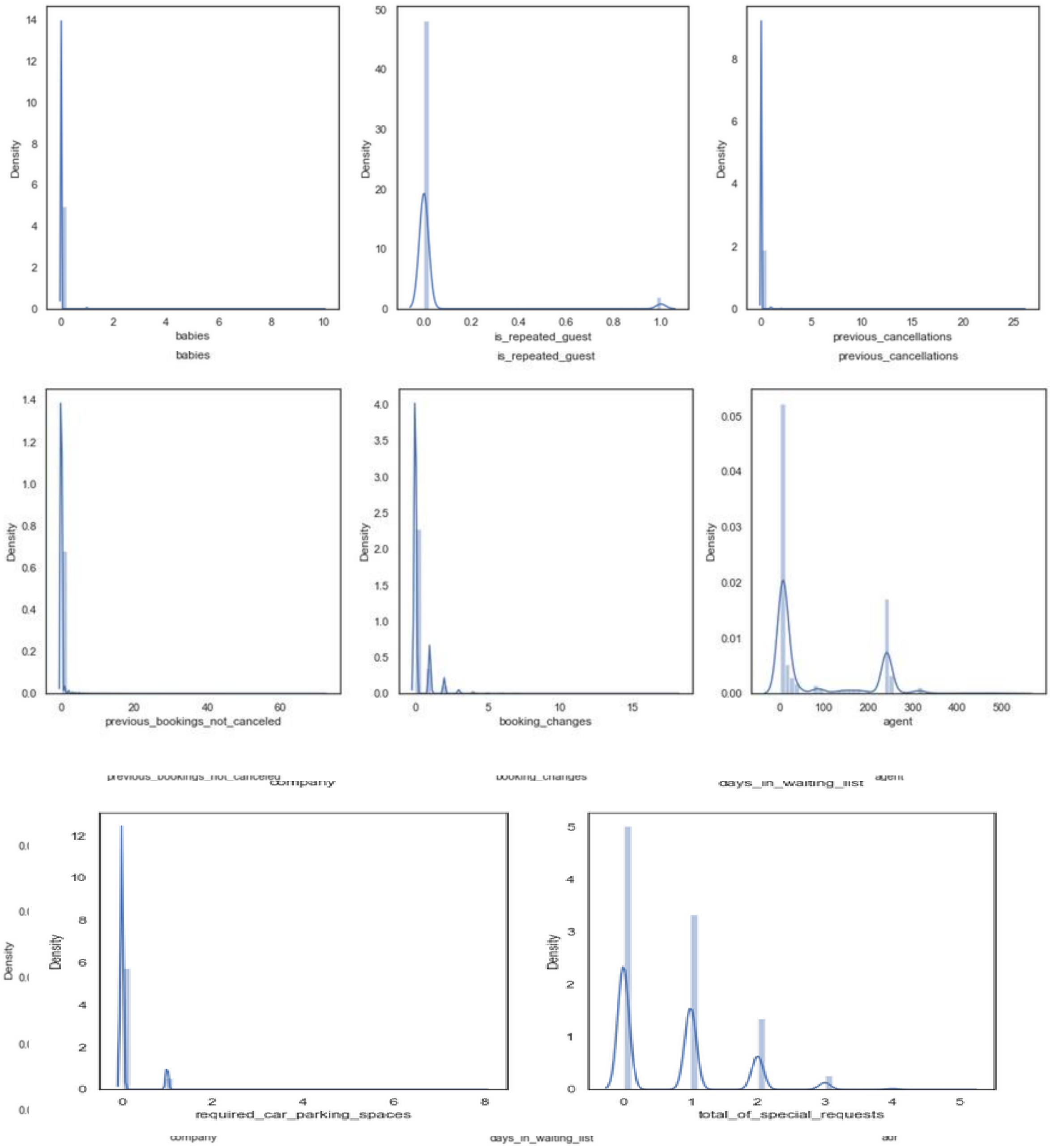
```python
# Plotting distplot to visualize distribution.
```

6]:

```python
plt.figure(figsize=(18,80),facecolor='white')
plot_num=1
for i in k1:
    ax=plt.subplot(12,3,plot_num)
    sns.distplot(hotel_data_clean[i])
    plot_num+=1
plt.show()
```



Exploratory Data Analysis



Data Preparation

Data processing was initiated by checking the description, info, column names and shape of the dataset.

Find the Shape or Size of Dataset

```
In [4]:   1 hotel_data.shape
Out[4]: (119390, 32)
```

Checking The Data Information

```
In [76]:   1 hotel_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
```

```
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

```
In [5]:   1 hotel_data.columns

Out[5]: 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'],
      dtype='object')
```

Data Preparation

In [74]:
```
hotel_data.describe().T
```

Out[74]:

	count	mean	std	min	25%	50%	75%	max
is_canceled	119390.0	0.370416	0.482918	0.00	0.00	0.000	1.0	1.0
lead_time	119390.0	104.011416	106.863097	0.00	18.00	69.000	160.0	737.0
arrival_date_year	119390.0	2016.156554	0.707476	2015.00	2016.00	2016.000	2017.0	2017.0
arrival_date_week_number	119390.0	27.165173	13.605138	1.00	16.00	28.000	38.0	53.0
arrival_date_day_of_month	119390.0	15.798241	8.780829	1.00	8.00	16.000	23.0	31.0
stays_in_weekend_nights	119390.0	0.927599	0.998613	0.00	0.00	1.000	2.0	19.0
stays_in_week_nights	119390.0	2.500302	1.908286	0.00	1.00	2.000	3.0	50.0
adults	119390.0	1.856403	0.579261	0.00	2.00	2.000	2.0	55.0
children	119386.0	0.103890	0.398561	0.00	0.00	0.000	0.0	10.0
babies	119390.0	0.007949	0.097436	0.00	0.00	0.000	0.0	10.0
is_repeated_guest	119390.0	0.031912	0.175767	0.00	0.00	0.000	0.0	1.0
previous_cancellations	119390.0	0.087118	0.844336	0.00	0.00	0.000	0.0	26.0
previous_bookings_not_canceled	119390.0	0.137097	1.497437	0.00	0.00	0.000	0.0	72.0
booking_changes	119390.0	0.221124	0.652306	0.00	0.00	0.000	0.0	21.0
agent	103050.0	86.693382	110.774548	1.00	9.00	14.000	229.0	535.0
company	6797.0	189.266735	131.655015	6.00	62.00	179.000	270.0	543.0
days_in_waiting_list	119390.0	2.321149	17.594721	0.00	0.00	0.000	0.0	391.0
adr	119390.0	101.831122	50.535790	-6.38	69.29	94.575	126.0	5400.0
required_car_parking_spaces	119390.0	0.062518	0.245291	0.00	0.00	0.000	0.0	8.0
total_of_special_requests	119390.0	0.571363	0.792798	0.00	0.00	0.000	1.0	5.0

Data Visualisation

In [109]:

```
# Plotting heatmap to see the intensity of relation.
plt.figure(figsize=(15,6))
sns.heatmap(hotel_data_clean.corr(),annot=True)
plt.xticks(rotation=25)
plt.show()
```

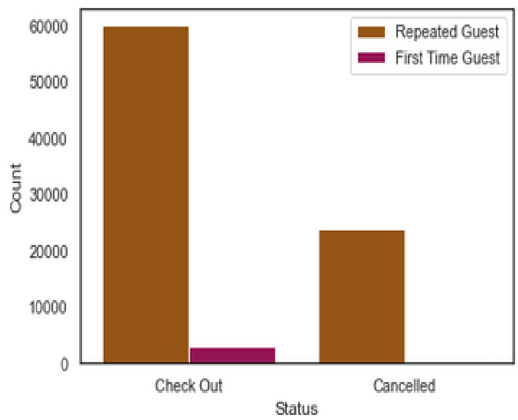
Cancellation by repeated guests

Let's check how many have cancelled their booking in the respective hotels

n [110]:

```
ax = sns.countplot(x="is_canceled", hue="is_repeated_guest", data=hotel_data_clean, palette = 'brg_r')
ax.set(xlabel='Status', ylabel='Count')
positions = (0, 1)
labels = ("Check Out", "Cancelled")
ax.set_xticklabels(labels)
LAB = {'Repeated Guest','First Time Guest'}
ax.legend(labels=LAB)
```

ut[110]: <matplotlib.legend.Legend at 0x1ff1c4cfa60>



Data Visualisation

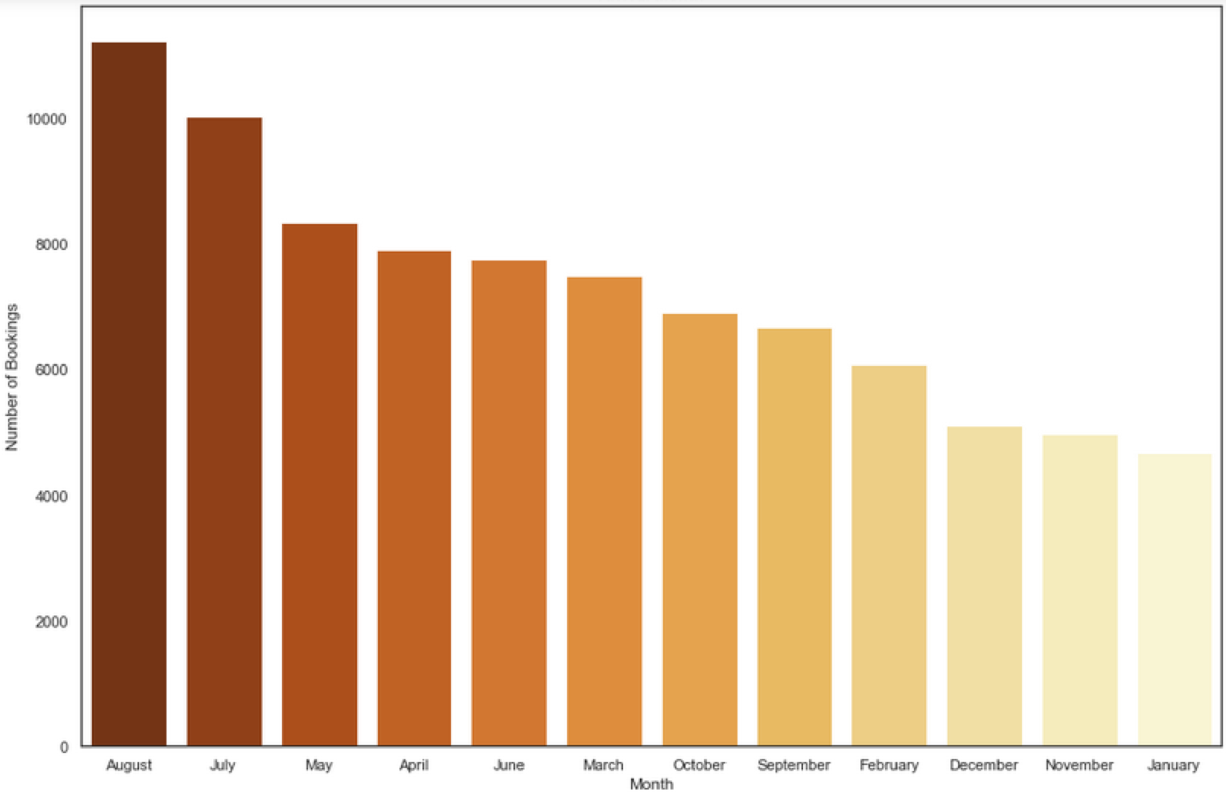
Most Busy Month

`arrival_date_month` exploration

In [112]:

```
plt.figure(figsize=(15,10))
sns.countplot(x='arrival_date_month', data = hotel_data_clean,
              order=pd.value_counts(hotel_data_clean['arrival_date_month']).index, palette='YlOrBr_r')
plt.title('Arrival Month', weight='bold')
plt.xlabel('Month', fontsize=12)
plt.ylabel('Number of Bookings', fontsize=12)
```

Out[112]: `Text(0, 0.5, 'Number of Bookings')`



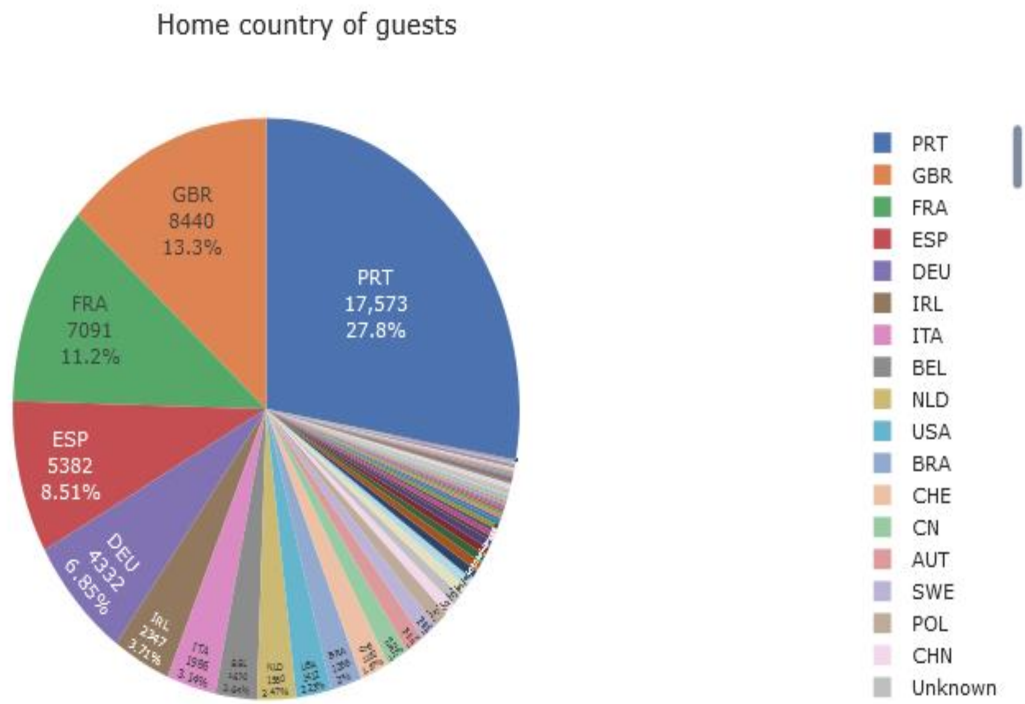
Data Visualisation

Where do the guests come from?

In [45]:

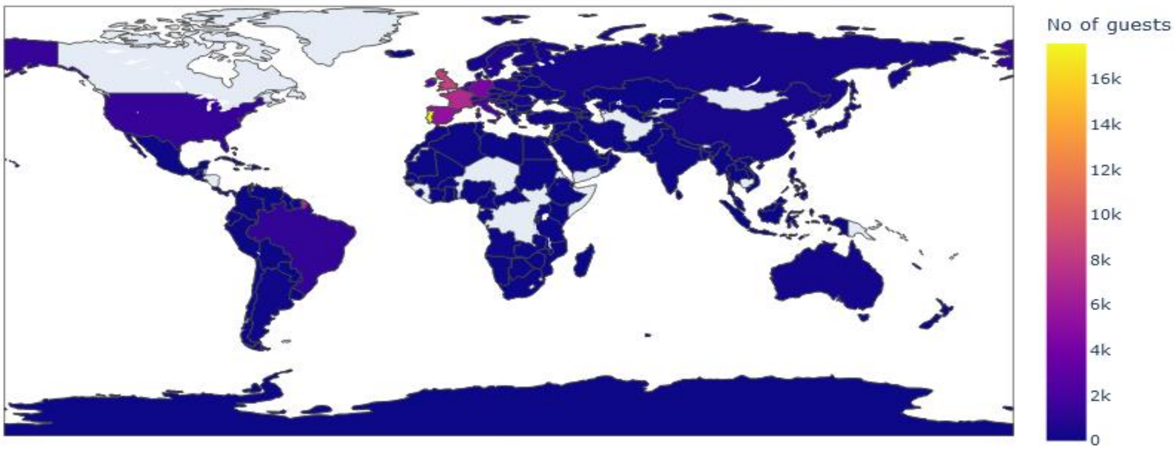
```
# get number of acutal guests by country
country_data = pd.DataFrame(hotel_data_clean[hotel_data_clean["is_canceled"] == 0]["country"].value_counts())
#country_data.index.name = "country"
country_data.rename(columns={"country": "Number of Guests"}, inplace=True)
total_guests = country_data["Number of Guests"].sum()
country_data["Guests in %"] = round(country_data["Number of Guests"] / total_guests * 100, 2)
country_data["country"] = country_data.index

# pie plot
fig = px.pie(country_data,
             values="Number of Guests",
             names="country",
             title="Home country of guests",
             template="seaborn")
fig.update_traces(textposition="inside", textinfo="value+percent+label")
fig.show()
```

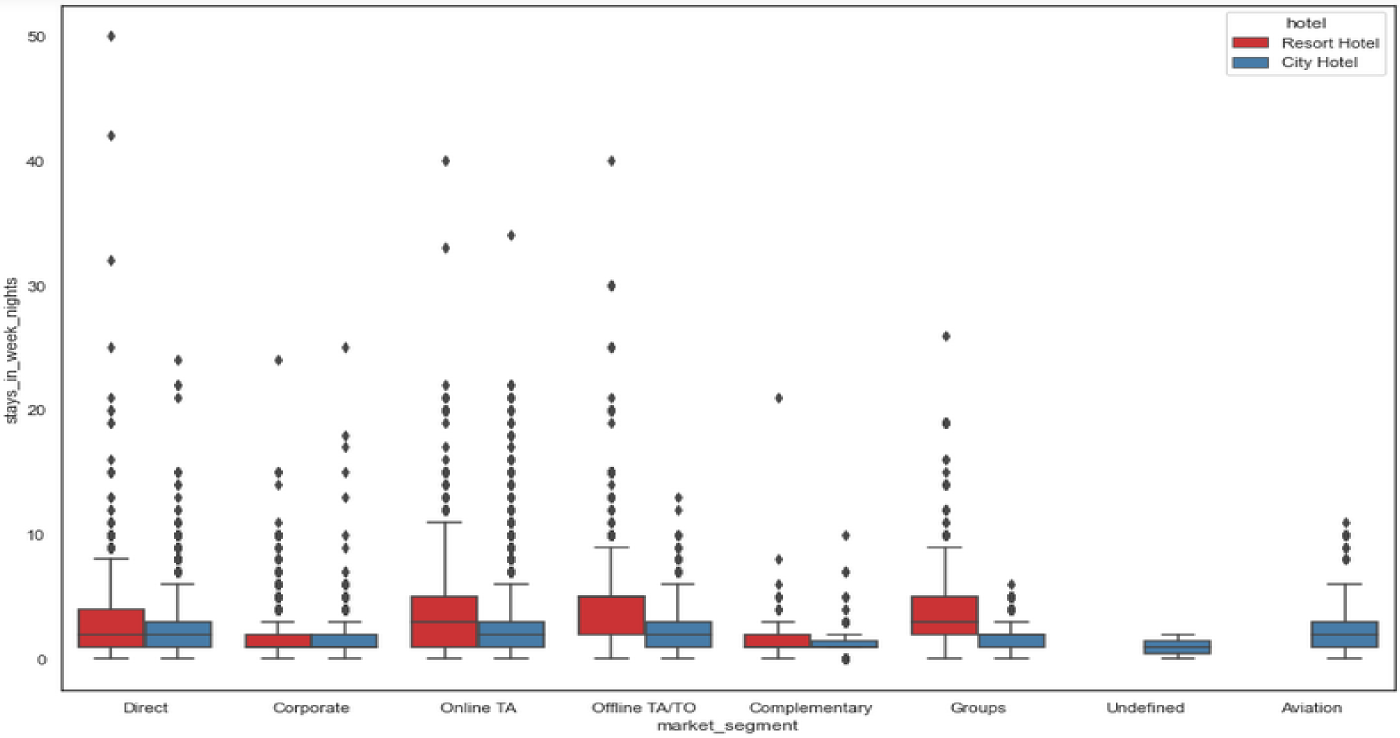


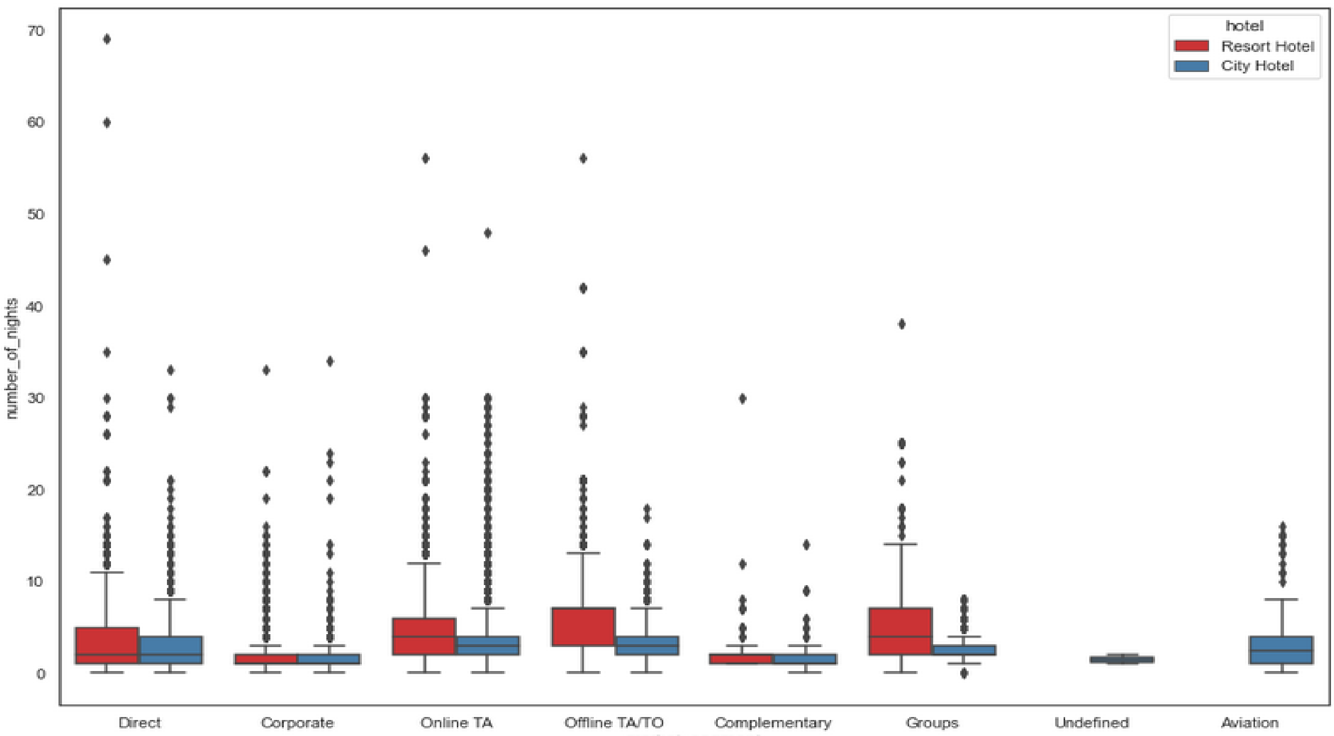
In [121]:

```
basemap = folium.Map()
guests_map = px.choropleth(guest_city, locations = guest_city['Country'],
                           color = guest_city['No of guests'], hover_name = guest_city['Country'])
guests_map.show()
```

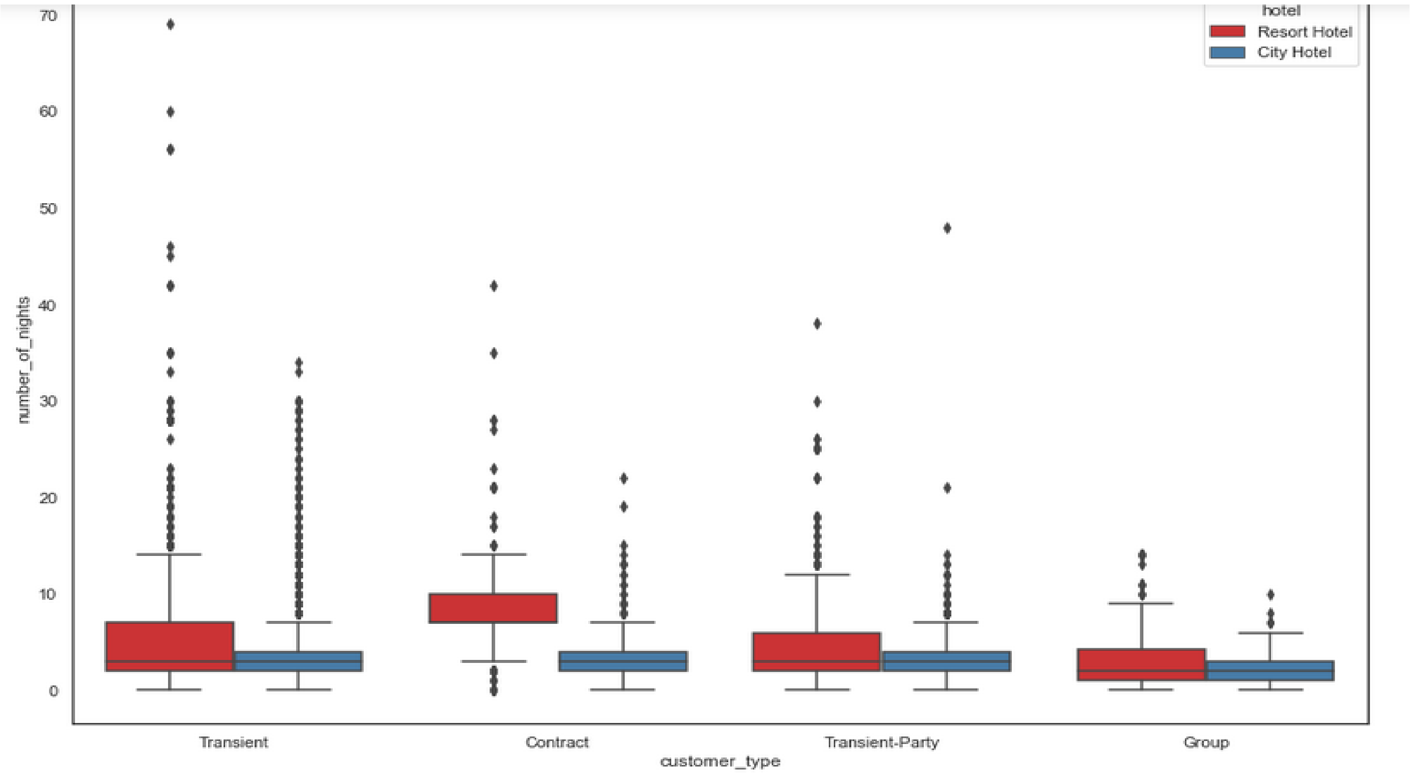


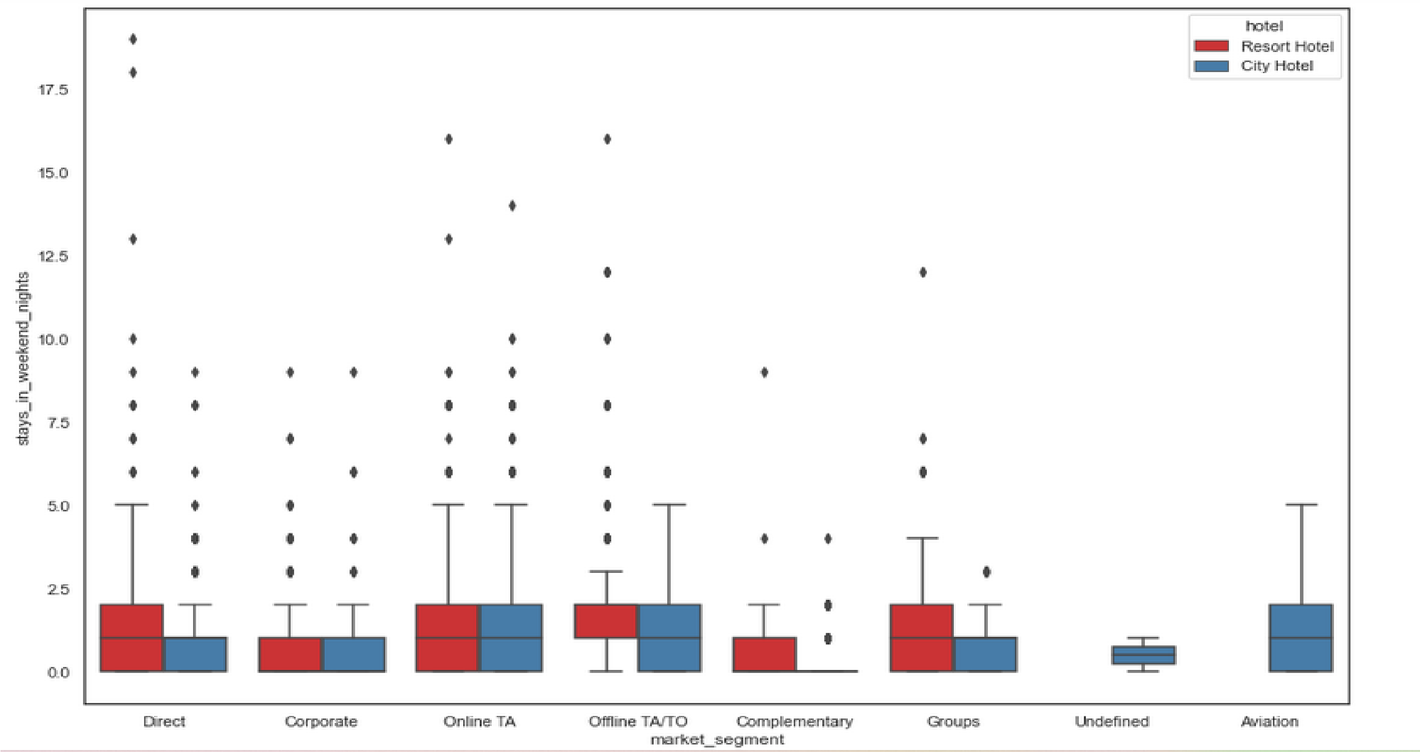
Data Visualisation



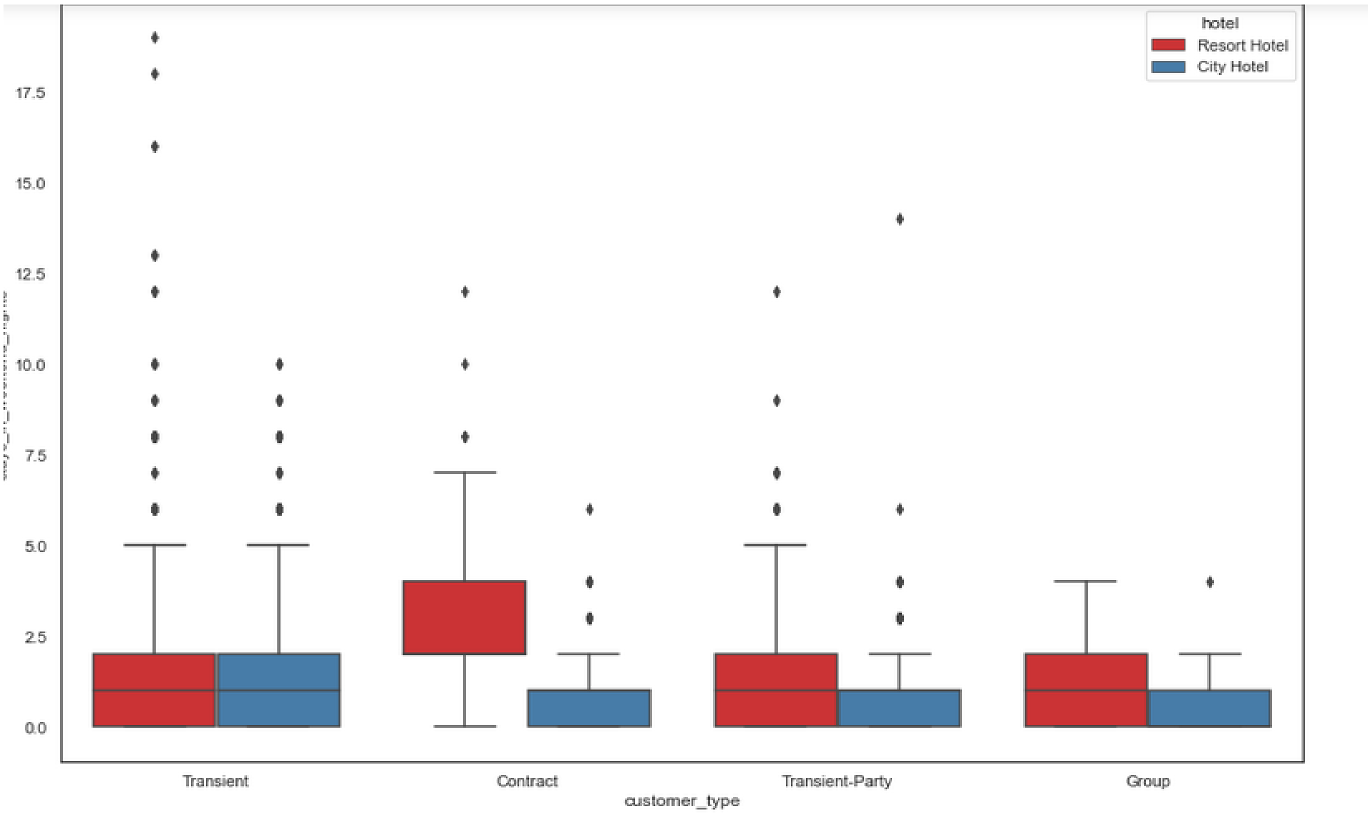


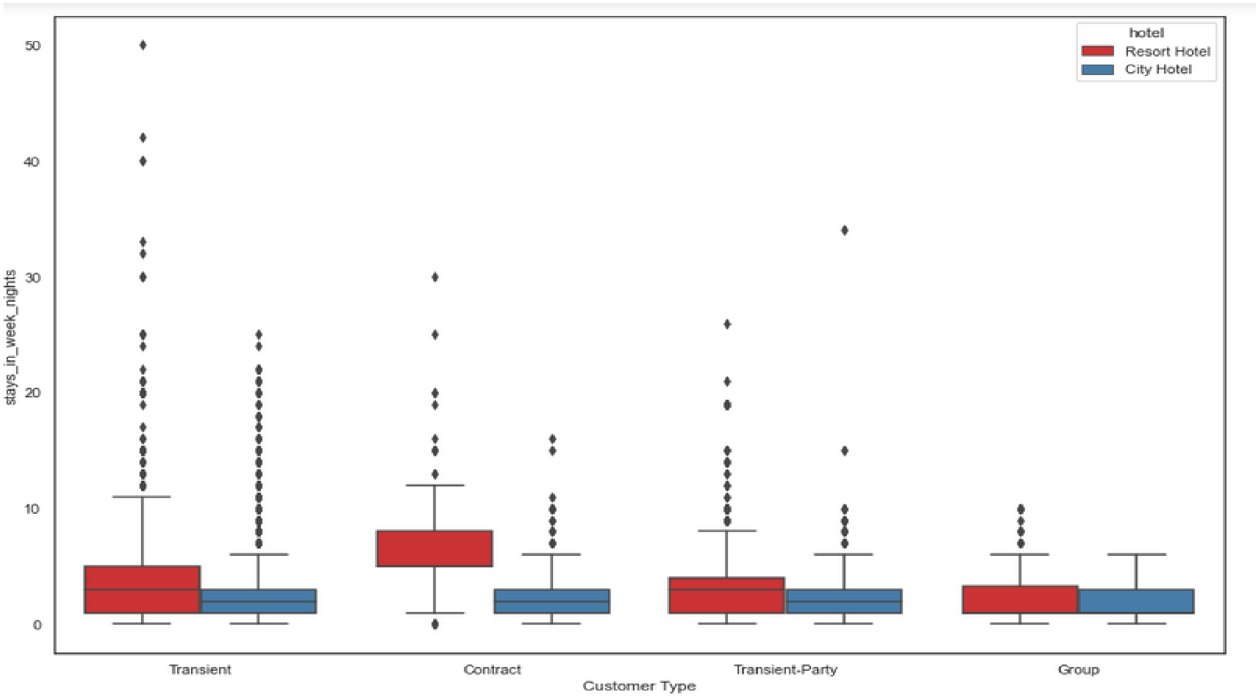
Data Visualisation





Data Visualisation

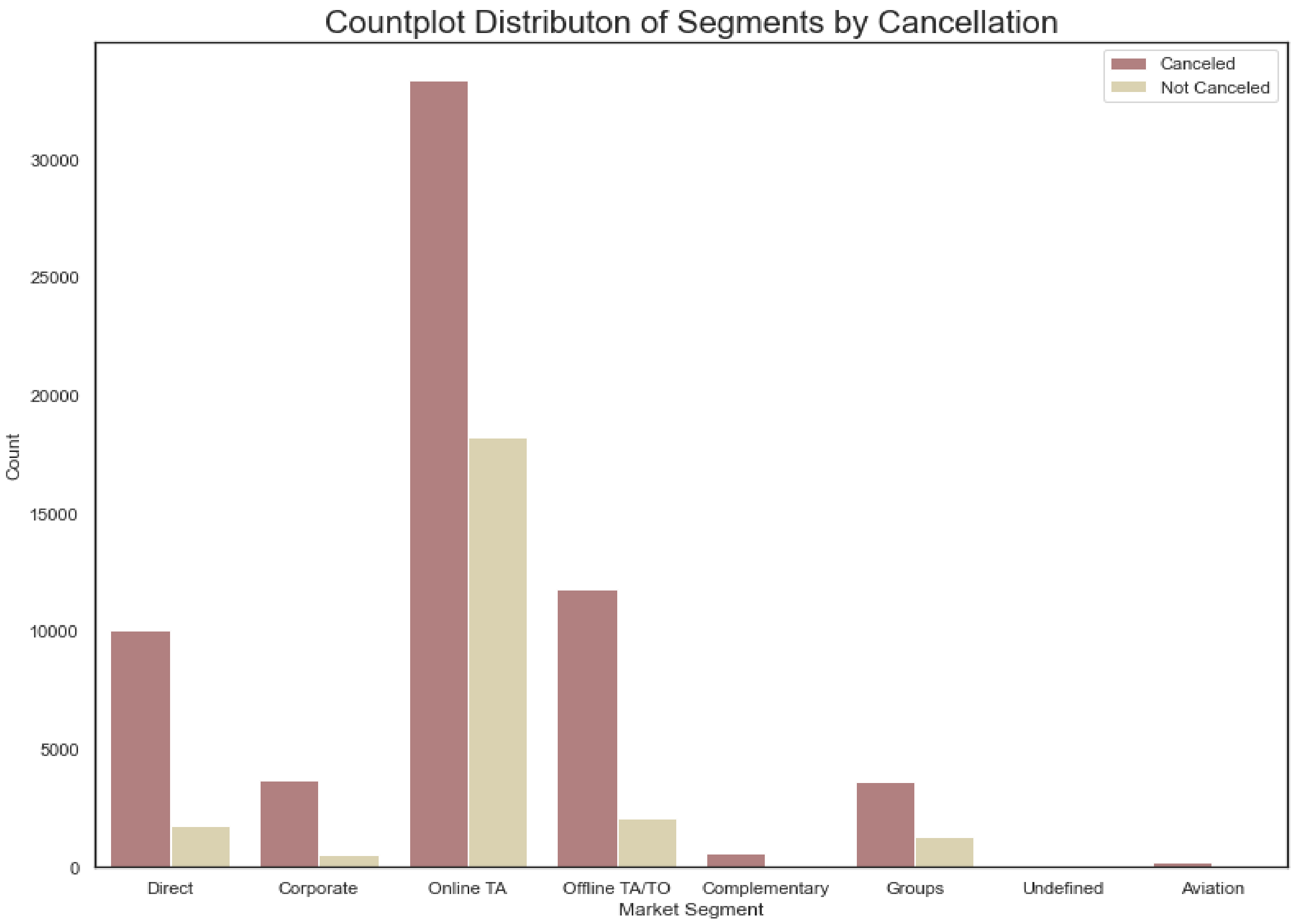




Data Visualisation

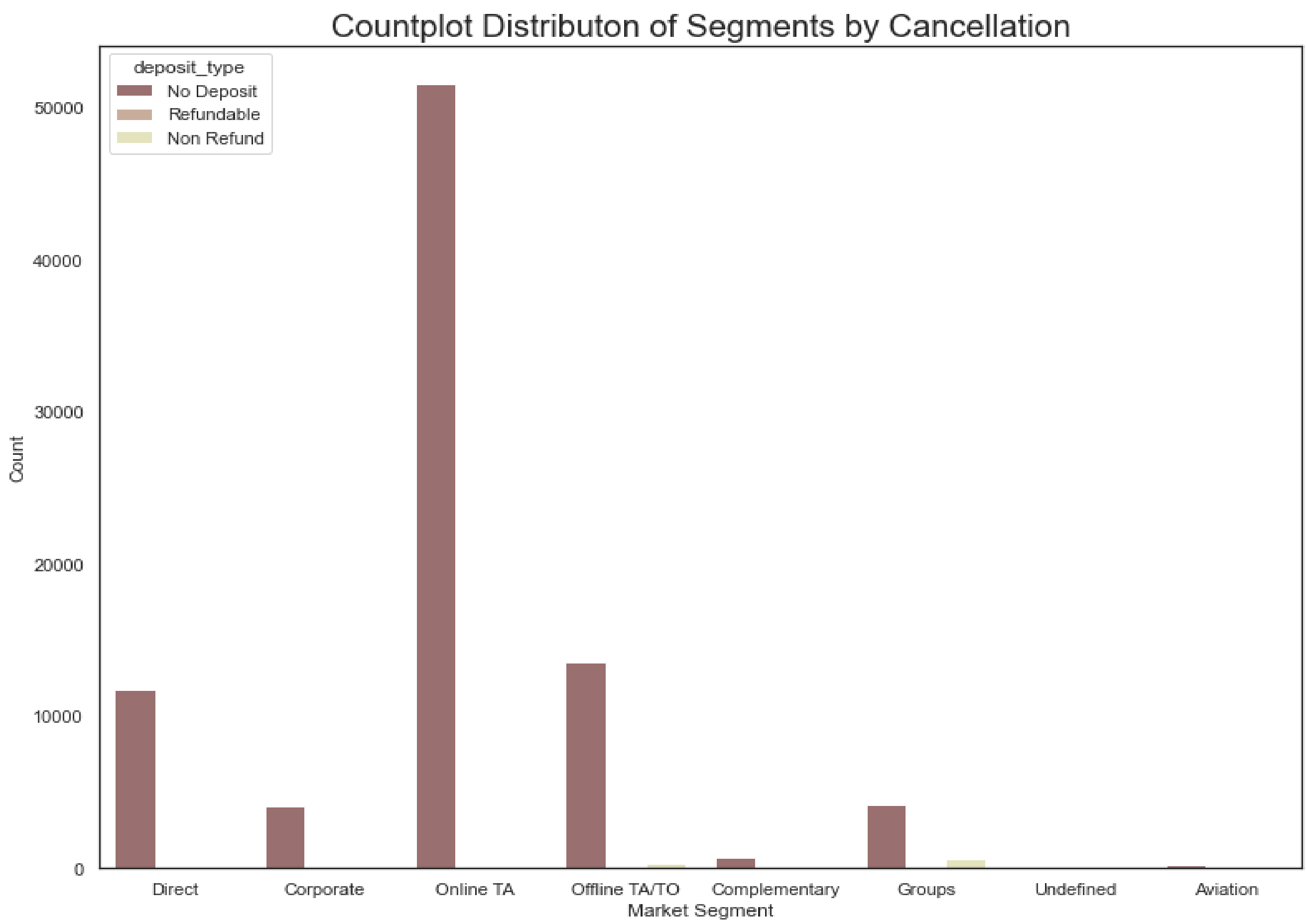
In [121]:
```python
plt.figure(figsize = (13,10))
ax = sns.countplot(x="market_segment", hue="is_canceled", data=hotel_data_clean, palette = 'pink')
ax.set(xlabel='Market Segment', ylabel='Count')
plt.title("Countplot Distributon of Segments by Cancellation", fontdict = {'fontsize':20})
LAB = {'Canceled','Not Canceled'}
ax.legend(labels=LAB)
```

Out[121]: `<matplotlib.legend.Legend at 0x1ff04aaedf0>`

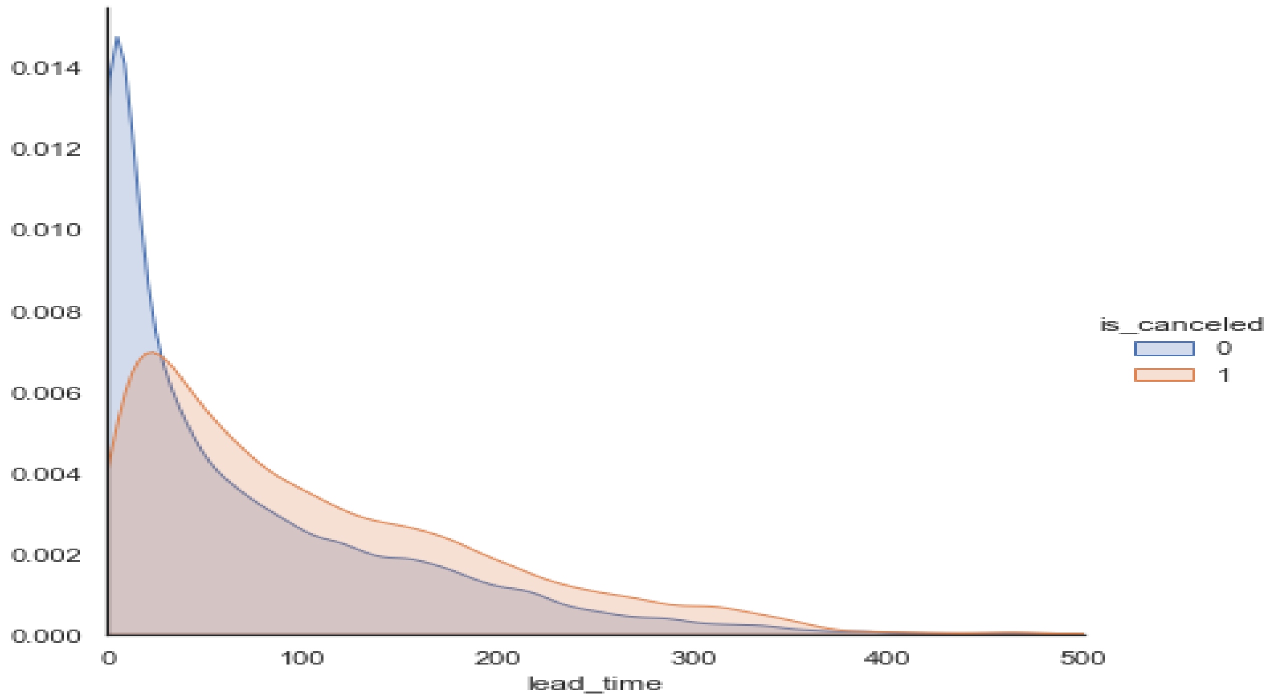


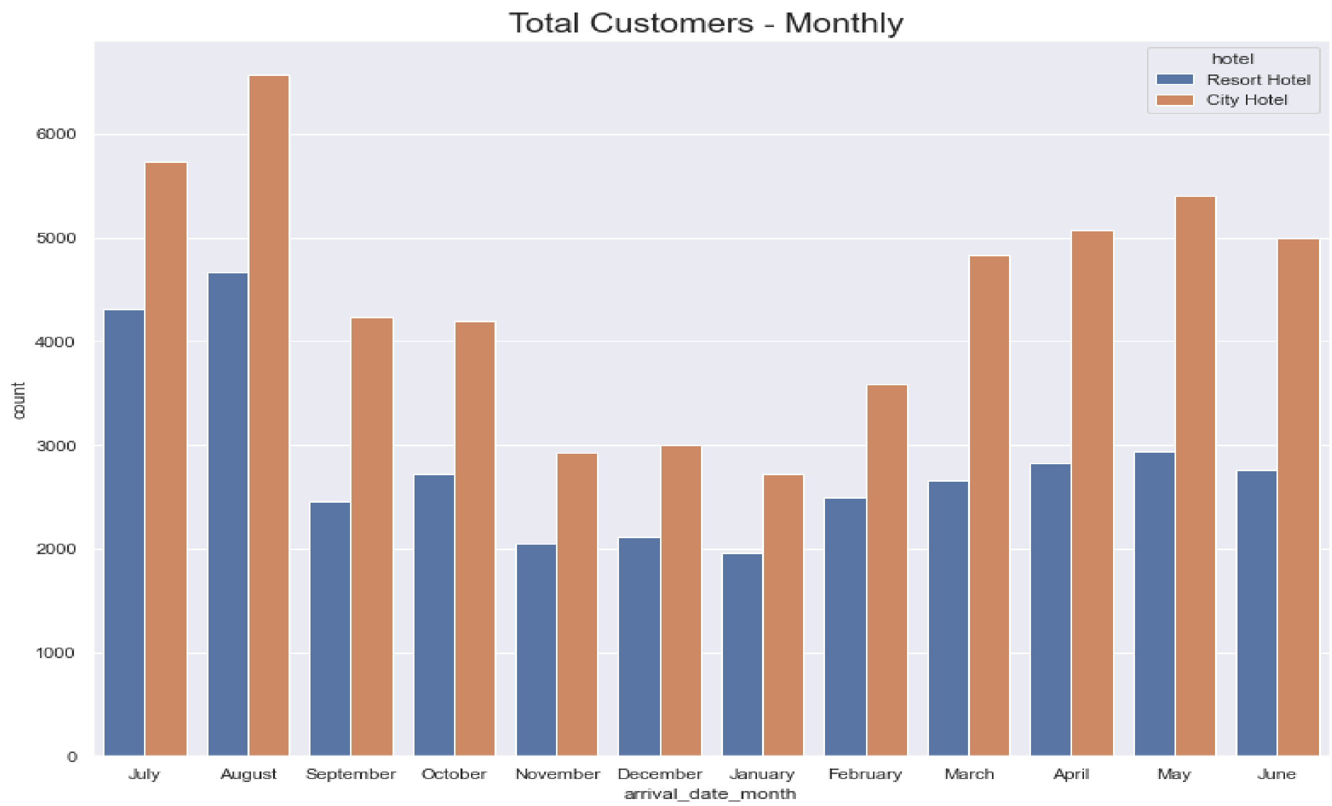
Data Visualisation

In [122]:
```
plt.figure(figsize = (13,10))
ax = sns.countplot(x="market_segment", hue="deposit_type", data=hotel_data_clean, palette = 'pink')
ax.set(xlabel='Market Segment', ylabel='Count')
plt.title("Countplot Distributon of Segments by Cancellation", fontdict = {'fontsize':20})
```
Out[122]:
```
Text(0.5, 1.0, 'Countplot Distributon of Segments by Cancellation')
```

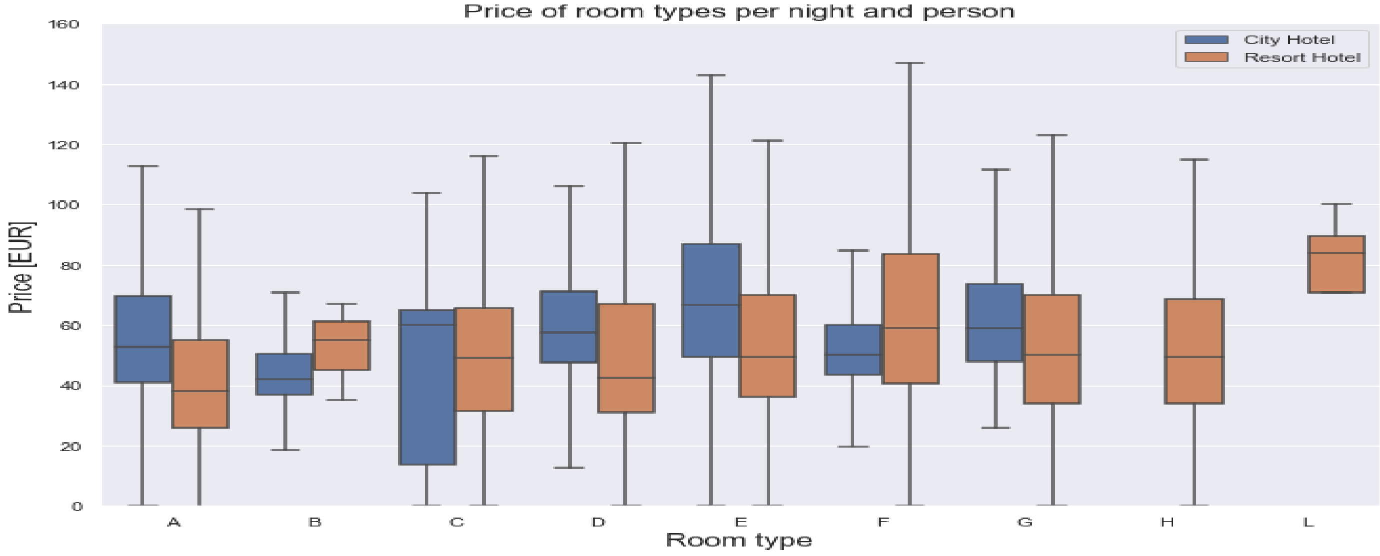


Data Visualisation

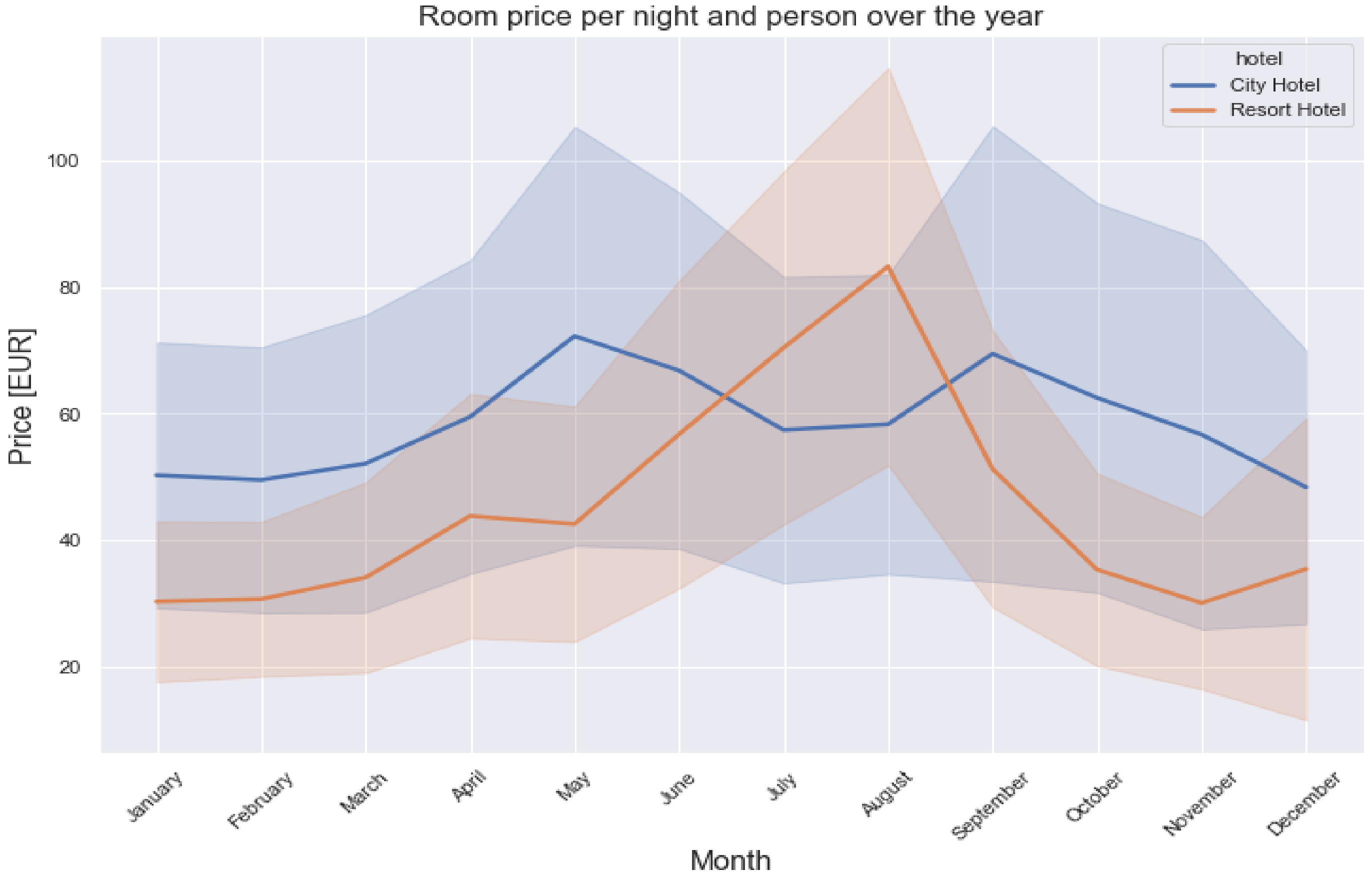




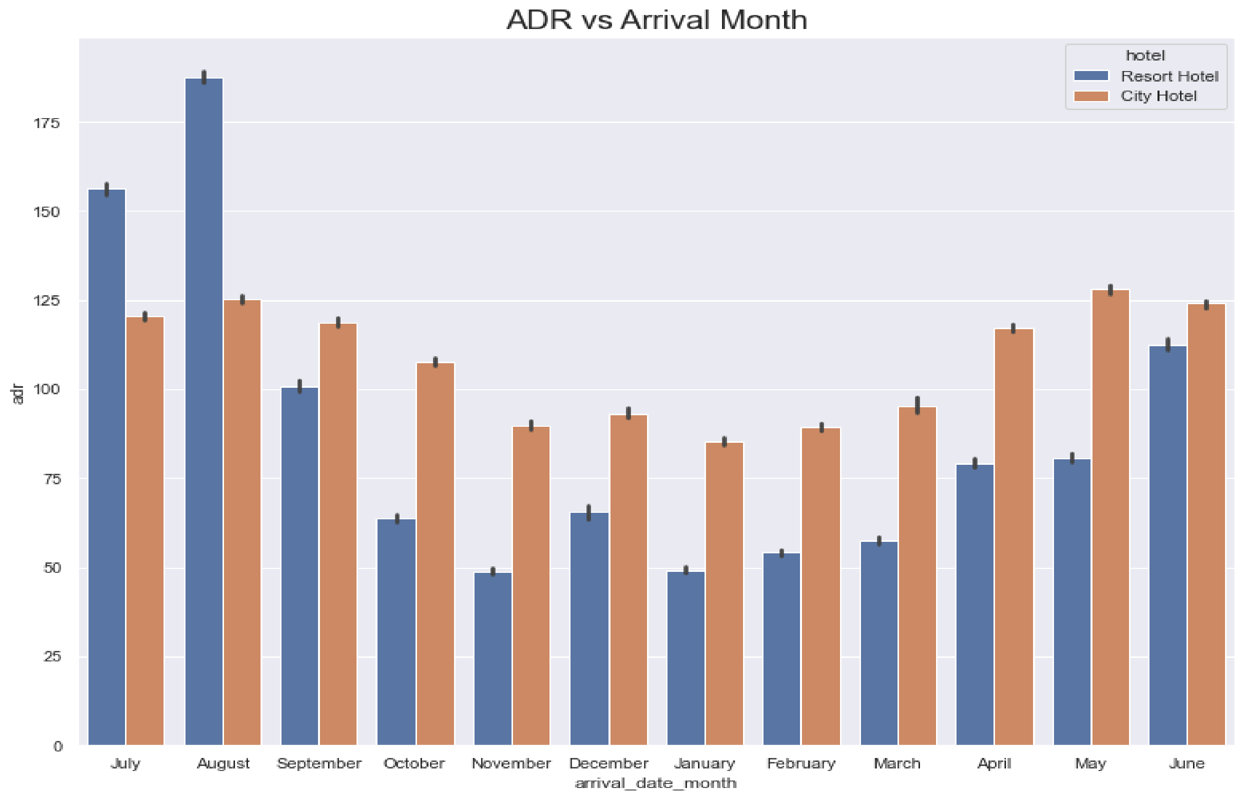
Data Visualisation



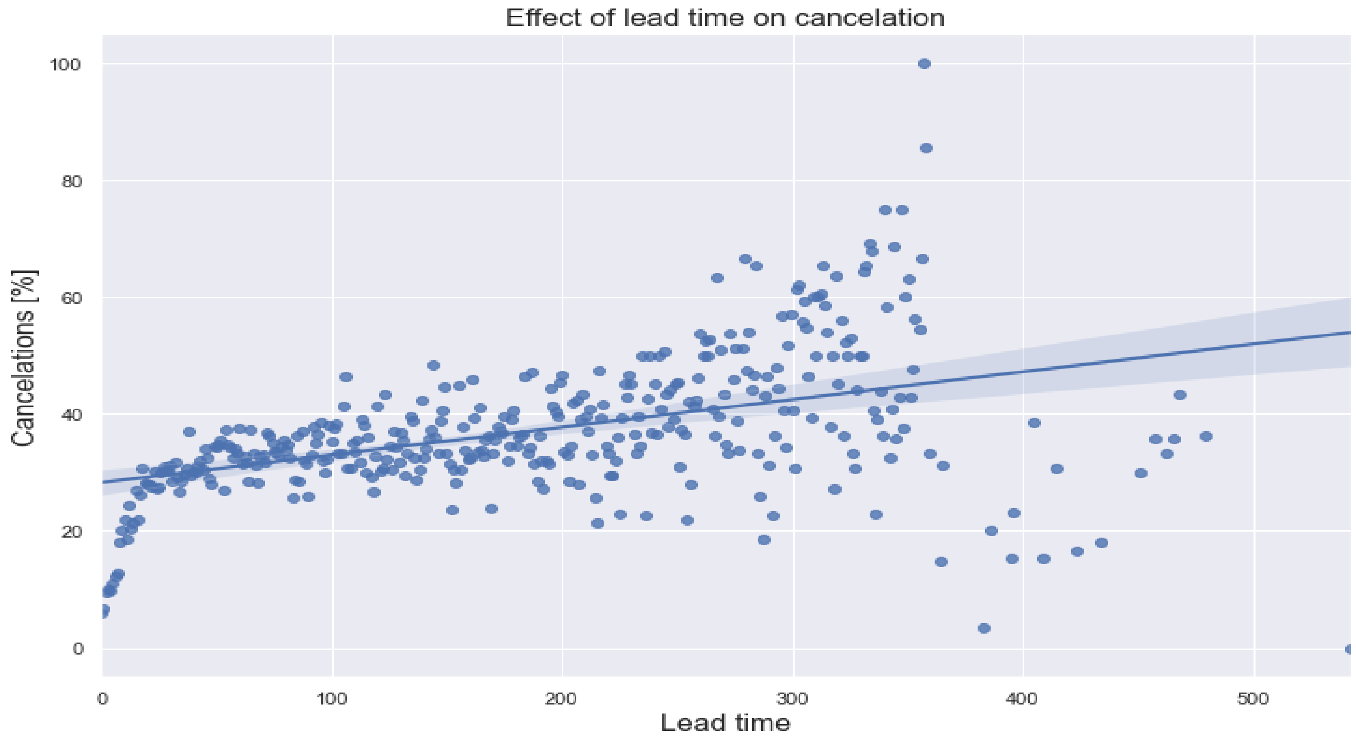
Order by Month



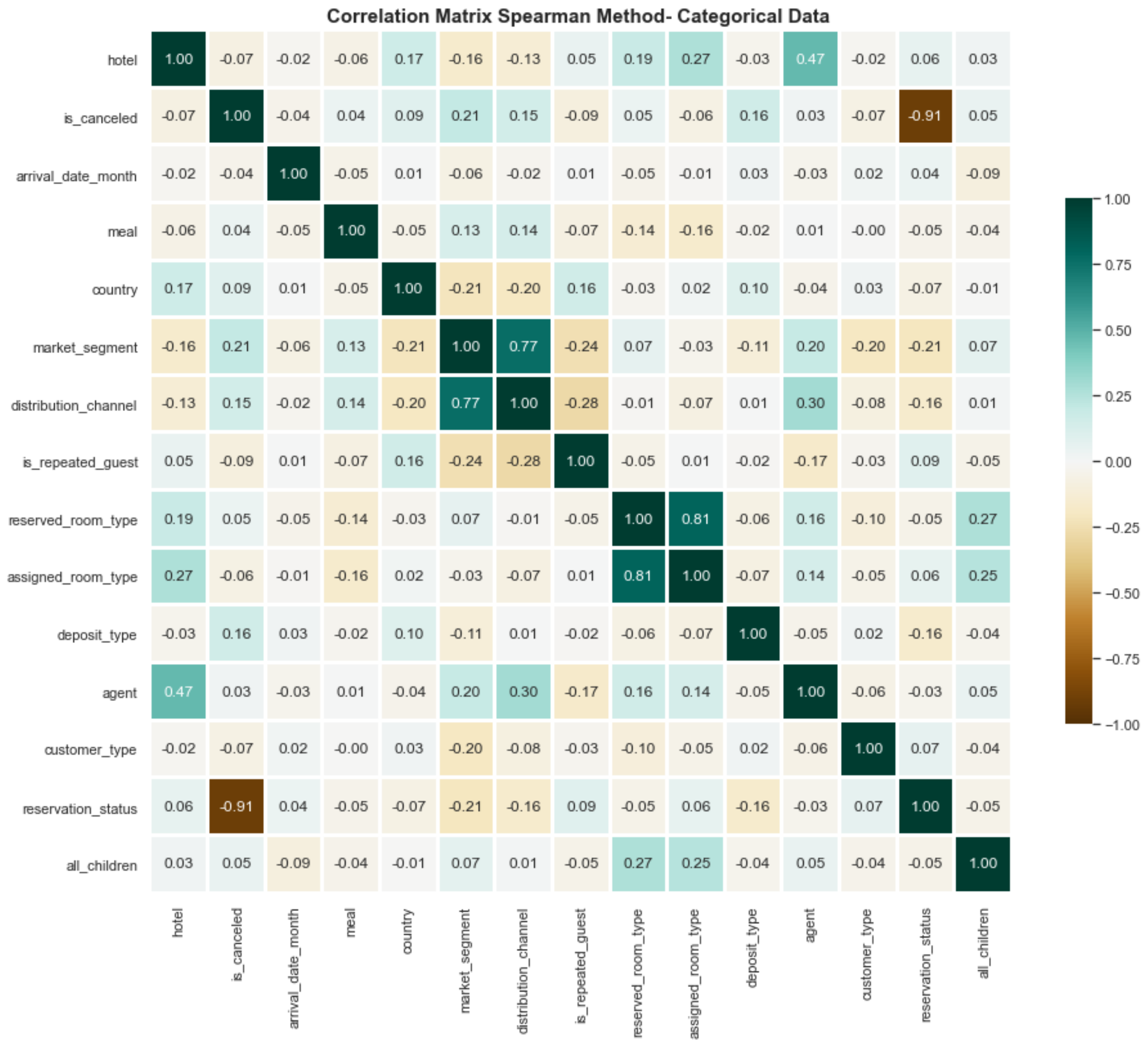
Data Visualisation



Cancelation Vs Lead Time



Correlation Matrix Spearman Method – Categorical Data



Predict Cancellation

In [139]:
```
cancel_corr = hotel_data.corr()["is_canceled"]
cancel_corr.abs().sort_values(ascending=False)[1:]
```

Out[139]:
```
lead_time                         0.293123
total_of_special_requests         0.234658
required_car_parking_spaces       0.195498
booking_changes                   0.144381
previous_cancellations            0.110133
is_repeated_guest                 0.084793
agent                             0.083114
adults                            0.060017
previous_bookings_not_canceled    0.057358
days_in_waiting_list              0.054186
adr                               0.047557
babies                            0.032491
stays_in_week_nights              0.024765
company                           0.020642
arrival_date_year                 0.016660
arrival_date_week_number          0.008148
arrival_date_day_of_month         0.006130
children                          0.005048
stays_in_weekend_nights           0.001791
Name: is_canceled, dtype: float64
```

Name: is_canceled, dtype: float64

In [140]:
```
hotel_data.groupby("is_canceled")["reservation_status"].value_counts()
```

Out[140]:
```
is_canceled  reservation_status
0            Check-Out             75166
1            Canceled              43017
             No-Show                1207
Name: reservation_status, dtype: int64
```

Data Pre-processing for Feature Engineering

In [143]:

```
num_features = ["lead_time","arrival_date_week_number","arrival_date_day_of_month",
                "stays_in_weekend_nights","stays_in_week_nights","adults","children",
                "babies","is_repeated_guest", "previous_cancellations",
                "previous_bookings_not_canceled","agent","company",
                "required_car_parking_spaces", "total_of_special_requests", "adr"]

cat_features = ["hotel","arrival_date_month","meal","market_segment",
                "distribution_channel","reserved_room_type","deposit_type","customer_type"]

# Separate features and predicted value
features = num_features + cat_features
X = hotel_data.drop(["is_canceled"], axis=1)[features]
y = hotel_data["is_canceled"]

#Creating Pipeline for the full_data
num_transformer = SimpleImputer(strategy="constant")

#Creating Pipeline for both kinds of data
# Preprocessing for categorical features:
cat_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="constant", fill_value="Unknown")),
    ("onehot", OneHotEncoder(handle_unknown='ignore'))])

# Bundle preprocessing for numerical and categorical features:
preprocessor = ColumnTransformer(transformers=[("num", num_transformer, num_features),
                                               ("cat", cat_transformer, cat_features)])
```

Modelling

To choose a model, there are a lot of factors to consider rather than focusing on performance every time. There are other factors like size of the dataset, how much time does a model take to train, accuracy, precision, and r2 score of the output, complexity of the data and still it leads to better results. Model selection is a process of choosing the best model that fits the data and addresses the problem.

What is a good model?

- Model that meets the requirements and constraints of the project
- Model that has a good training time or speed
- Model that is better than old ways of prediction
- Model that is skilled to get better results when compiled on test set.

SELECTION OF THE ALGORITHMS USED IN MODELLING –

I. Decision Tree _model
II. Random Forest Classifier model
III. Logistic Regression _model
IV. XGB Classifier model

Modelling

In [144]:
```
# define models to test:
base_models = [("DecisionTree_model", DecisionTreeClassifier(random_state=42)),
               ("RandomForestClassifier_model", RandomForestClassifier(random_state=42,n_jobs=-1)),
               ("LogisticRegression_model", LogisticRegression(random_state=42,n_jobs=-1)),
               ("XGBClassifier_model", XGBClassifier(random_state=42, n_jobs=-1))]


kfolds = 4
split = KFold(n_splits=kfolds, shuffle=True, random_state=42)

# Preprocessing, fitting, making predictions and scoring for every model:
for name, model in base_models:

    model_steps = Pipeline(steps=[('preprocessor', preprocessor),
                              ('model', model)])

    cv_results = cross_val_score(model_steps,
                                 X, y,
                                 cv=split,
                                 scoring="accuracy",
                                 n_jobs=-1)

    min_score = round(min(cv_results), 4)
    max_score = round(max(cv_results), 4)
    mean_score = round(np.mean(cv_results), 4)
    std_dev = round(np.std(cv_results), 4)
    print(f"{name} cross validation accuarcy score: {mean_score} +/- {std_dev} (std) min: {min_score}, max: {max_score}, "
```

```
DecisionTree_model cross validation accuarcy score: 0.8246 +/- 0.0016 (std) min: 0.8221, max: 0.8263,
RandomForestClassifier_model cross validation accuarcy score: 0.8664 +/- 0.0012 (std) min: 0.8646, max: 0.8676,
LogisticRegression_model cross validation accuarcy score: 0.7935 +/- 0.0012 (std) min: 0.7919, max: 0.7951,
XGBClassifier_model cross validation accuarcy score: 0.8473 +/- 0.0011 (std) min: 0.8456, max: 0.8487,
```

Enhanced RF model with the best parameters I found:

In [145]:
```
rf_model_enh = RandomForestClassifier(n_estimators=160,
                                      max_features=0.4,
                                      min_samples_split=2,
                                      n_jobs=-1,
                                      random_state=0)

split = KFold(n_splits=kfolds, shuffle=True, random_state=42)
model_pipe = Pipeline(steps=[('preprocessor', preprocessor),
                             ('model', rf_model_enh)])
cv_results = cross_val_score(model_pipe,
                             X, y,
                             cv=split,
                             scoring="accuracy",
                             n_jobs=-1)
# output:
min_score = round(min(cv_results), 4)
max_score = round(max(cv_results), 4)
mean_score = round(np.mean(cv_results), 4)
std_dev = round(np.std(cv_results), 4)
print(f"Enhanced RF model cross validation accuarcy score: {mean_score} +/- {std_dev} (std) min: {min_score}, max: {max_sc
```

```
Enhanced RF model cross validation accuarcy score: 0.8681 +/- 0.0006 (std) min: 0.8673, max: 0.869
```

Conclusion

As you probably noticed, explainable machine learning gives a lot opportunities to validate predictive models, find most insightful facts about data and set new directions to improve our results.

Explainable machine learning methods can be compared to the pointing fingers on every model weakness and to the compass guiding where we should go to improve final outcome.

8 Tips to Reduce Last Minute Hotel Cancellations and No Shows

- 1. Make Sure You have a Solid Cancellation Policy in Place
- 2. Require Credit / Debit Card Deposits
- 3. Set Discounted or Advance Purchase Rates
- 4. Use Length of Stay Restrictions
- 5. Sweeten the Deal for Direct Bookings (Offer discounts)
- 6. Send Your Guests Email Reminders About their Booking
- 7. Adopt A Cautious Overbooking Strategy
- 8. Be Responsive and Proactive

Minimizing the overbooking Problem

a. The Solution is found in reducing the need to overbook rooms.
b. Guest will want the best of both worlds; the ability to extend or shorten a stay while not being charged additionally for either.
c. The hotel industry is increasing their restrictive policies, similar to the airline industry.