HOTEL BOOKING PREDICTION

IMPORTING LIBRARIES

In [1]:
```
# importing libraries

!pip install missingno
!pip install catboost
!pip install lightgbm
!pip install folium
!pip install plotly

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import plotly.express as px
import plotly.offline as pyo

import warnings
warnings.filterwarnings('ignore')


from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import VotingClassifier

import folium
from folium.plugins import HeatMap
import plotly.express as px

plt.style.use('fivethirtyeight')
%matplotlib inline
pd.set_option('display.max_columns', 32)
```

```
Requirement already satisfied: missingno in c:\users\aish\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: matplotlib in c:\users\aish\anaconda3\lib\site-packages (from missingno) (3.7.0)
Requirement already satisfied: scipy in c:\users\aish\anaconda3\lib\site-packages (from missingno) (1.10.0)
Requirement already satisfied: numpy in c:\users\aish\anaconda3\lib\site-packages (from missingno) (1.23.5)
Requirement already satisfied: seaborn in c:\users\aish\anaconda3\lib\site-packages (from missingno) (0.12.2)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (22.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (1.0.5)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (9.4.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: pandas>=0.25 in c:\users\aish\anaconda3\lib\site-packages (from seaborn->missingno) (1.5.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\aish\anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingno) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\aish\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Requirement already satisfied: catboost in c:\users\aish\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: plotly in c:\users\aish\anaconda3\lib\site-packages (from catboost) (5.9.0)
Requirement already satisfied: numpy>=1.16.0 in c:\users\aish\anaconda3\lib\site-packages (from catboost) (1.23.5)
Requirement already satisfied: matplotlib in c:\users\aish\anaconda3\lib\site-packages (from catboost) (3.7.0)
Requirement already satisfied: graphviz in c:\users\aish\anaconda3\lib\site-packages (from catboost) (0.20.1)
Requirement already satisfied: pandas>=0.24 in c:\users\aish\anaconda3\lib\site-packages (from catboost) (1.5.3)
Requirement already satisfied: six in c:\users\aish\anaconda3\lib\site-packages (from catboost) (1.16.0)
Requirement already satisfied: scipy in c:\users\aish\anaconda3\lib\site-packages (from catboost) (1.10.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\aish\anaconda3\lib\site-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\aish\anaconda3\lib\site-packages (from pandas>=0.24->catboost) (2022.7)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (4.25.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: packaging>=20.0 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (22.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: cycler>=0.10 in c:\users\aish\anaconda3\lib\site-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\aish\anaconda3\lib\site-packages (from plotly->catboost) (8.0.1)
Requirement already satisfied: lightgbm in c:\users\aish\anaconda3\lib\site-packages (4.1.0)
Requirement already satisfied: numpy in c:\users\aish\anaconda3\lib\site-packages (from lightgbm) (1.23.5)
Requirement already satisfied: scipy in c:\users\aish\anaconda3\lib\site-packages (from lightgbm) (1.10.0)
Requirement already satisfied: folium in c:\users\aish\anaconda3\lib\site-packages (0.14.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\aish\anaconda3\lib\site-packages (from folium) (0.6.0)
Requirement already satisfied: jinja2>=2.9 in c:\users\aish\anaconda3\lib\site-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in c:\users\aish\anaconda3\lib\site-packages (from folium) (1.23.5)
Requirement already satisfied: requests in c:\users\aish\anaconda3\lib\site-packages (from folium) (2.28.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\aish\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\aish\anaconda3\lib\site-packages (from requests->folium) (2023.5.7)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\aish\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\aish\anaconda3\lib\site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\aish\anaconda3\lib\site-packages (from requests->folium) (1.26.14)
Requirement already satisfied: plotly in c:\users\aish\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\aish\anaconda3\lib\site-packages (from plotly) (8.0.1)
```

1 IMPORTING DATA

In [2]:

```
# reading data
df = pd.read_csv('hotel_bookings.csv')
df.head()
```

Out[2]:

	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	distributi
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Direct	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Corporate	
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	Online TA	

In [3]:

```
df.describe()
```

Out[3]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	is_repeated_guest	previous_cancellatio
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119386.000000	119390.000000	119390.000000	119390.0000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.927599	2.500302	1.856403	0.103890	0.007949	0.031912	0.0871
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613	1.908286	0.579261	0.398561	0.097436	0.175767	0.8443
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.0000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000	2.000000	2.000000	0.000000	0.000000	0.000000	0.0000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000	3.000000	2.000000	0.000000	0.000000	0.000000	0.0000
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000	50.000000	55.000000	10.000000	10.000000	1.000000	26.0000

In [4]:

```
df.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
 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
```

2 CLEANING NULL VALUES

In [5]:
```
# checking for null values

null = pd.DataFrame({'Null Values' : df.isna().sum(), 'Percentage Null Values' : (df.isna().sum()) / (df.shape[0]) * (100)})
null
```

Out[5]:

	Null Values	Percentage Null Values
hotel	0	0.000000
is_canceled	0	0.000000
lead_time	0	0.000000
arrival_date_year	0	0.000000
arrival_date_month	0	0.000000
arrival_date_week_number	0	0.000000
arrival_date_day_of_month	0	0.000000
stays_in_weekend_nights	0	0.000000
stays_in_week_nights	0	0.000000
adults	0	0.000000
children	4	0.003350
babies	0	0.000000
meal	0	0.000000
country	488	0.408744
market_segment	0	0.000000
distribution_channel	0	0.000000
is_repeated_guest	0	0.000000
previous_cancellations	0	0.000000
previous_bookings_not_canceled	0	0.000000
reserved_room_type	0	0.000000
assigned_room_type	0	0.000000
booking_changes	0	0.000000
deposit_type	0	0.000000
agent	16340	13.686238
company	112593	94.306893
days_in_waiting_list	0	0.000000
customer_type	0	0.000000
adr	0	0.000000
required_car_parking_spaces	0	0.000000
total_of_special_requests	0	0.000000
reservation_status	0	0.000000
reservation_status_date	0	0.000000

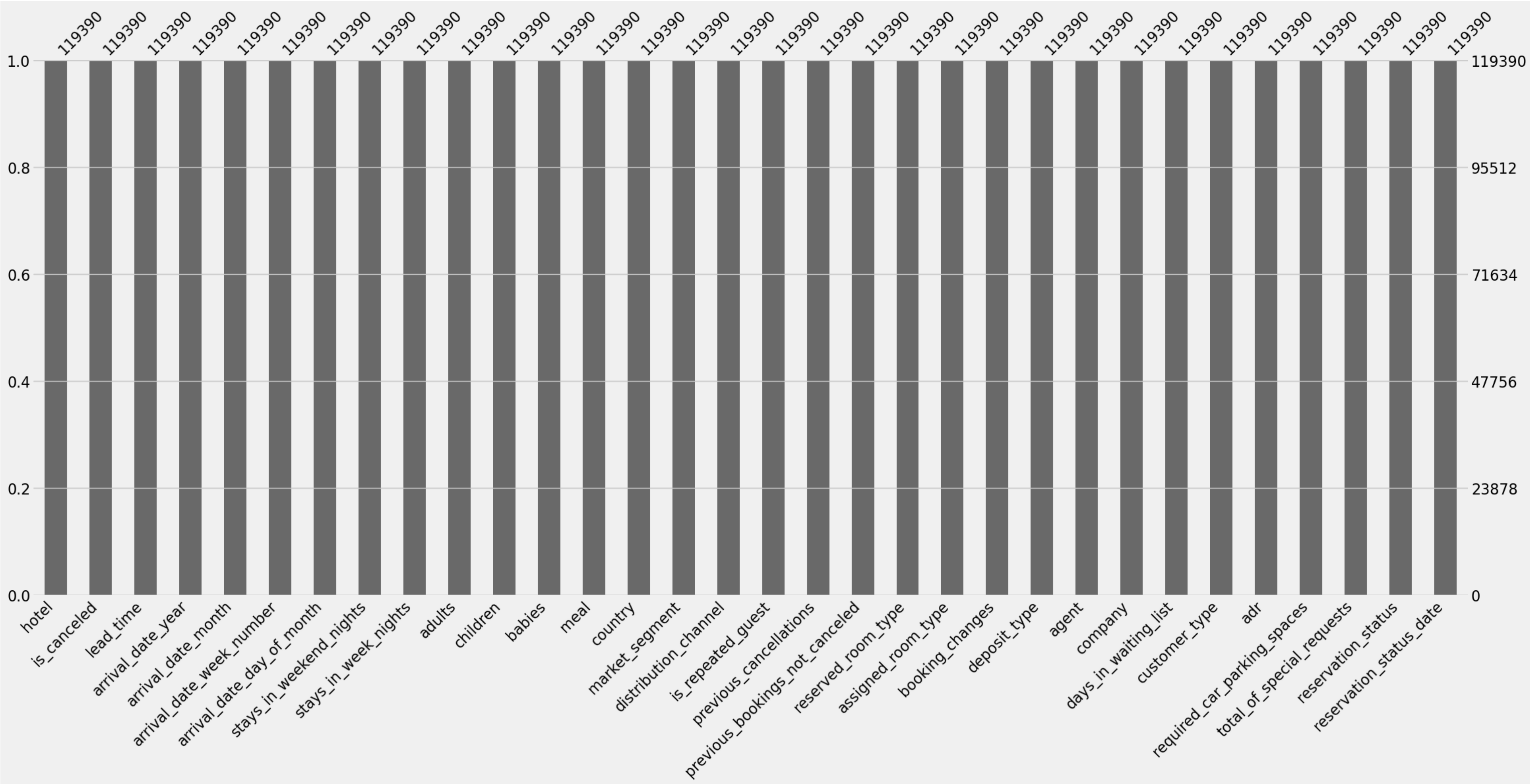
In [6]:

```
# filling null values with zero

df.fillna(0, inplace = True)
```

In [7]:

```
# visualizing null values
msno.bar(df)
plt.show()
```



3 CLEANING THE DATA

In [8]:

```
# adults, babies and children cant be zero at same time, so dropping the rows having all these zero at same time

filter = (df.children == 0) & (df.adults == 0) & (df.babies == 0)
df[filter]
```

Out[8]:

	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	dis
2224	Resort Hotel	0	1	2015	October	41	6	0	3	0	0.0	0	SC	PRT	Corporate	
2409	Resort Hotel	0	0	2015	October	42	12	0	0	0	0.0	0	SC	PRT	Corporate	
3181	Resort Hotel	0	36	2015	November	47	20	1	2	0	0.0	0	SC	ESP	Groups	
3684	Resort Hotel	0	165	2015	December	53	30	1	4	0	0.0	0	SC	PRT	Groups	
3708	Resort Hotel	0	165	2015	December	53	30	2	4	0	0.0	0	SC	PRT	Groups	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
115029	City Hotel	0	107	2017	June	26	27	0	3	0	0.0	0	BB	CHE	Online TA	
115091	City Hotel	0	1	2017	June	26	30	0	1	0	0.0	0	SC	PRT	Complementary	
116251	City Hotel	0	44	2017	July	28	15	1	1	0	0.0	0	SC	SWE	Online TA	
116534	City Hotel	0	2	2017	July	28	15	2	5	0	0.0	0	SC	RUS	Online TA	
117087	City Hotel	0	170	2017	July	30	27	0	2	0	0.0	0	BB	BRA	Offline TA/TO	

180 rows × 32 columns

In [9]:

```
df = df[~filter]
df
```

Out[9]:

	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	dis
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Direct	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Corporate	
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	Online TA	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	35	30	2	5	2	0.0	0	BB	BEL	Offline TA/TO	
119386	City Hotel	0	102	2017	August	35	31	2	5	3	0.0	0	BB	FRA	Online TA	
119387	City Hotel	0	34	2017	August	35	31	2	5	2	0.0	0	BB	DEU	Online TA	
119388	City Hotel	0	109	2017	August	35	31	2	5	2	0.0	0	BB	GBR	Online TA	
119389	City Hotel	0	205	2017	August	35	29	2	7	2	0.0	0	HB	DEU	Online TA	

119210 rows × 32 columns

4 EXPLORATORY DATA ANALYSIS(EDA)

From where the most guests are coming ?

In [10]:

```
country_wise_guests = df[df['is_canceled'] == 0]['country'].value_counts().reset_index()
country_wise_guests.columns = ['country', 'No of guests']
country_wise_guests
```

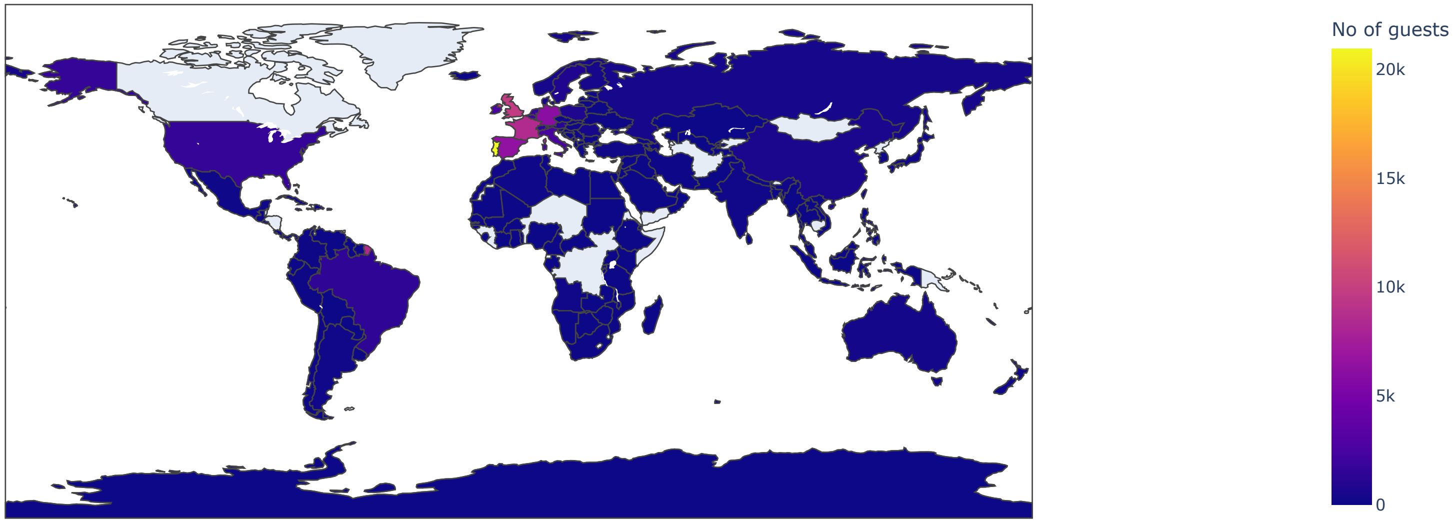
Out[10]:

	country	No of guests
0	PRT	20977
1	GBR	9668
2	FRA	8468
3	ESP	6383
4	DEU	6067
...	...	...
161	BHR	1
162	DJI	1
163	MLI	1
164	NPL	1
165	FRO	1

166 rows × 2 columns

In [11]:

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



People from all over the world are staying in these two hotels. Most guests are from Portugal and other countries in Europe.

How much do guests pay for a room per night?

In [12]:
```
df.head()
```

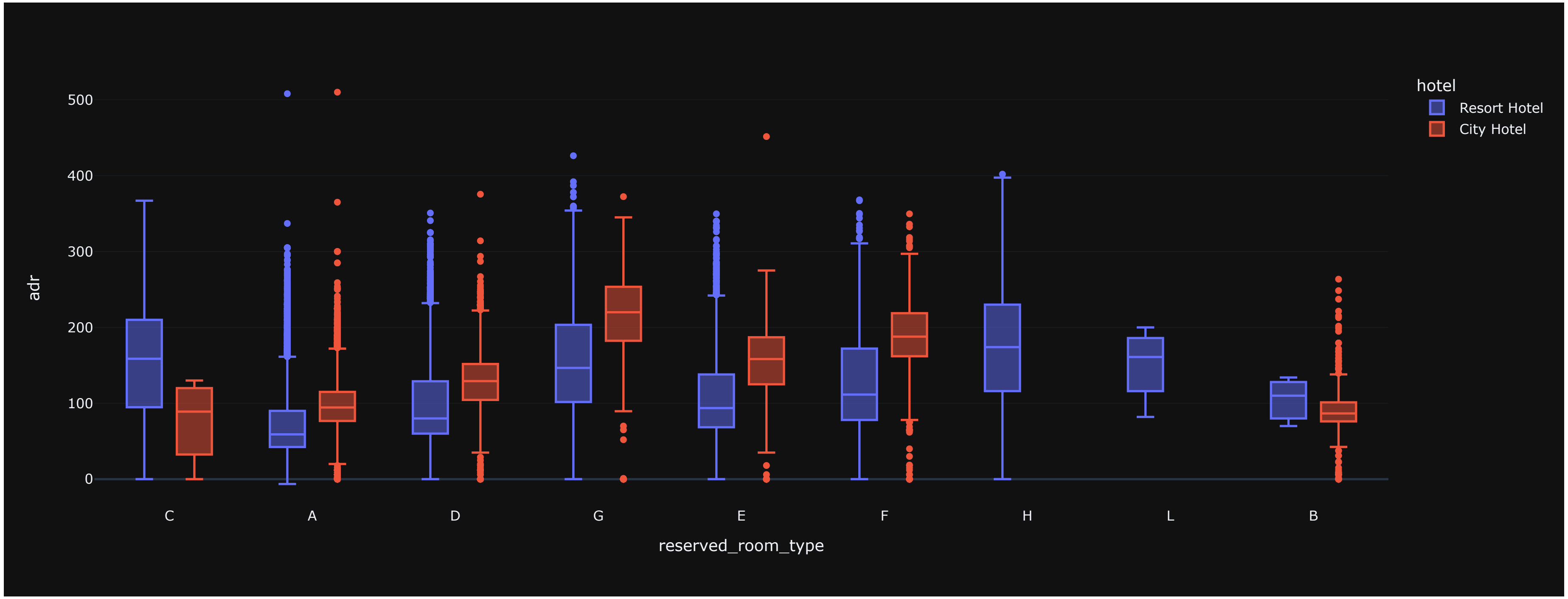
Out[12]:

	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	distributi
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Direct	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Corporate	
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	Online TA	

Both hotels have different room types and different meal arrangements.Seasonal factors are also important, So the prices varies a lot.

In [13]:
```
data = df[df['is_canceled'] == 0]

px.box(data_frame = data, x = 'reserved_room_type', y = 'adr', color = 'hotel', template = 'plotly_dark')
```



The figure shows that the average price per room depends on its type and the standard deviation

How does the price vary per night over the year?

In [14]:
```
data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] == 0)]
data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] == 0)]
```

In [15]:
```
resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'].mean().reset_index()
resort_hotel
```

Out[15]:

	arrival_date_month	adr
0	April	75.867816
1	August	181.205892
2	December	68.410104
3	February	54.147478
4	January	48.761125
5	July	150.122528
6	June	107.974850
7	March	57.056838
8	May	76.657558
9	November	48.706289
10	October	61.775449
11	September	96.416860

In [16]:

```
city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index()
city_hotel
```

Out[16]:

	arrival_date_month	adr
0	April	111.962267
1	August	118.674598
2	December	88.401855
3	February	86.520062
4	January	82.330983
5	July	115.818019
6	June	117.874360
7	March	90.658533
8	May	120.669827
9	November	86.946592
10	October	102.004672
11	September	112.776582

In [17]:

```
final_hotel = resort_hotel.merge(city_hotel, on = 'arrival_date_month')
final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']
final_hotel
```

Out[17]:

	month	price_for_resort	price_for_city_hotel
0	April	75.867816	111.962267
1	August	181.205892	118.674598
2	December	68.410104	88.401855
3	February	54.147478	86.520062
4	January	48.761125	82.330983
5	July	150.122528	115.818019
6	June	107.974850	117.874360
7	March	57.056838	90.658533
8	May	76.657558	120.669827
9	November	48.706289	86.946592
10	October	61.775449	102.004672
11	September	96.416860	112.776582

Now we observe here that month column is not in order, and if we visualize we will get improper conclusions.

So, first we have to provide right hierarchy to month column.

In [18]:

```
!pip install sort-dataframeby-monthorweek

!pip install sorted-months-weekdays
```

```
Requirement already satisfied: sort-dataframeby-monthorweek in c:\users\aish\anaconda3\lib\site-packages (0.4)
Requirement already satisfied: sorted-months-weekdays in c:\users\aish\anaconda3\lib\site-packages (0.2)
```

In [19]:

```
import sort_dataframeby_monthorweek as sd

def sort_month(df, column_name):
    return sd.Sort_Dataframeby_Month(df, column_name)
```

In [20]:

```
final_prices = sort_month(final_hotel, 'month')
final_prices
```

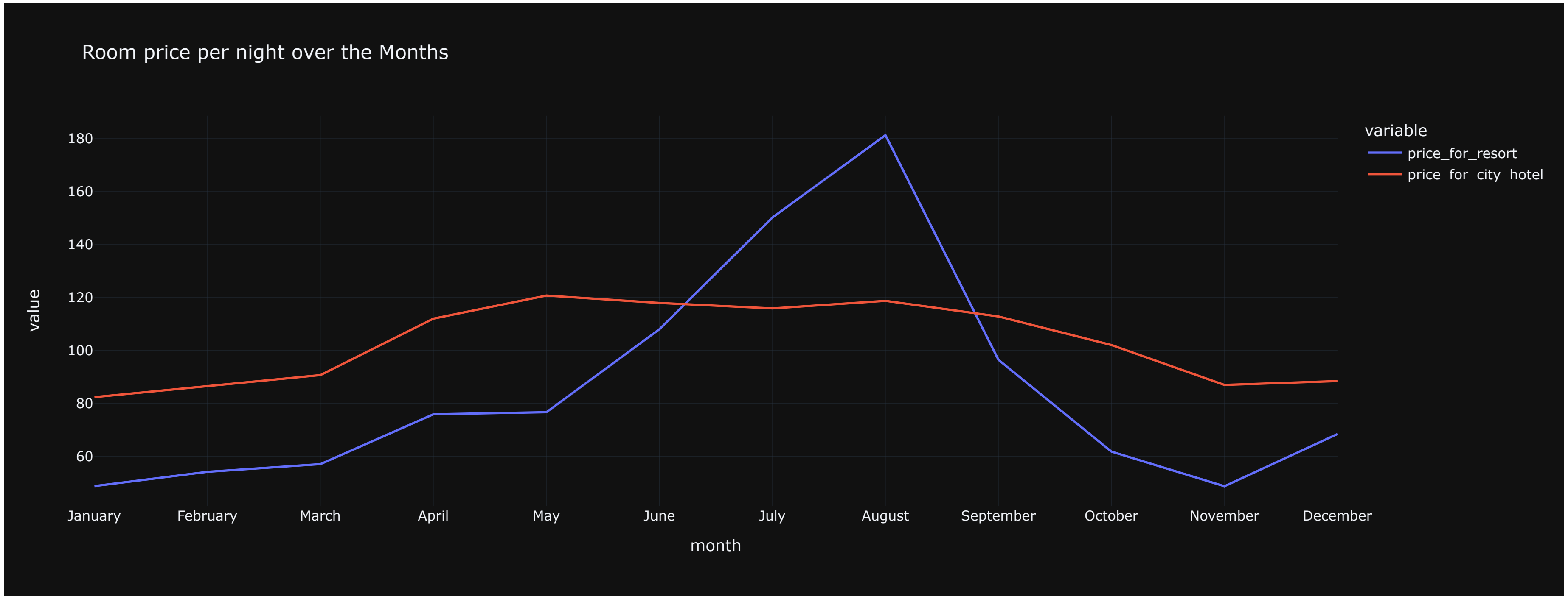
Out[20]:

	month	price_for_resort	price_for_city_hotel
0	January	48.761125	82.330983
1	February	54.147478	86.520062
2	March	57.056838	90.658533
3	April	75.867816	111.962267
4	May	76.657558	120.669827
5	June	107.974850	117.874360
6	July	150.122528	115.818019
7	August	181.205892	118.674598
8	September	96.416860	112.776582
9	October	61.775449	102.004672
10	November	48.706289	86.946592
11	December	68.410104	88.401855

In [21]:

```
plt.figure(figsize = (17, 8))

px.line(final_prices, x = 'month', y = ['price_for_resort','price_for_city_hotel'],
        title = 'Room price per night over the Months', template = 'plotly_dark')
```




```
<Figure size 1700x800 with 0 Axes>
```

This plot clearly shows that prices in the Resort Hotel are much higher during the summer and prices of city hotel varies less and is most expensive during Spring and Autumn .

Which are the most busy months?

In [22]:
```
resort_guests = data_resort['arrival_date_month'].value_counts().reset_index()
resort_guests.columns=['month','no of guests']
resort_guests
```

Out[22]:

	month	no of guests
0	August	3257
1	July	3137
2	October	2575
3	March	2571
4	April	2550
5	May	2535
6	February	2308
7	September	2102
8	June	2037
9	December	2014
10	November	1975
11	January	1866

In [23]:
```
city_guests = data_city['arrival_date_month'].value_counts().reset_index()
city_guests.columns=['month','no of guests']
city_guests
```

Out[23]:

	month	no of guests
0	August	5367
1	July	4770
2	May	4568
3	June	4358
4	October	4326
5	September	4283
6	March	4049
7	April	4010
8	February	3051
9	November	2676
10	December	2377
11	January	2249

In [24]:
```
final_guests = resort_guests.merge(city_guests,on='month')
final_guests.columns=['month','no of guests in resort','no of guest in city hotel']
final_guests
```

Out[24]:

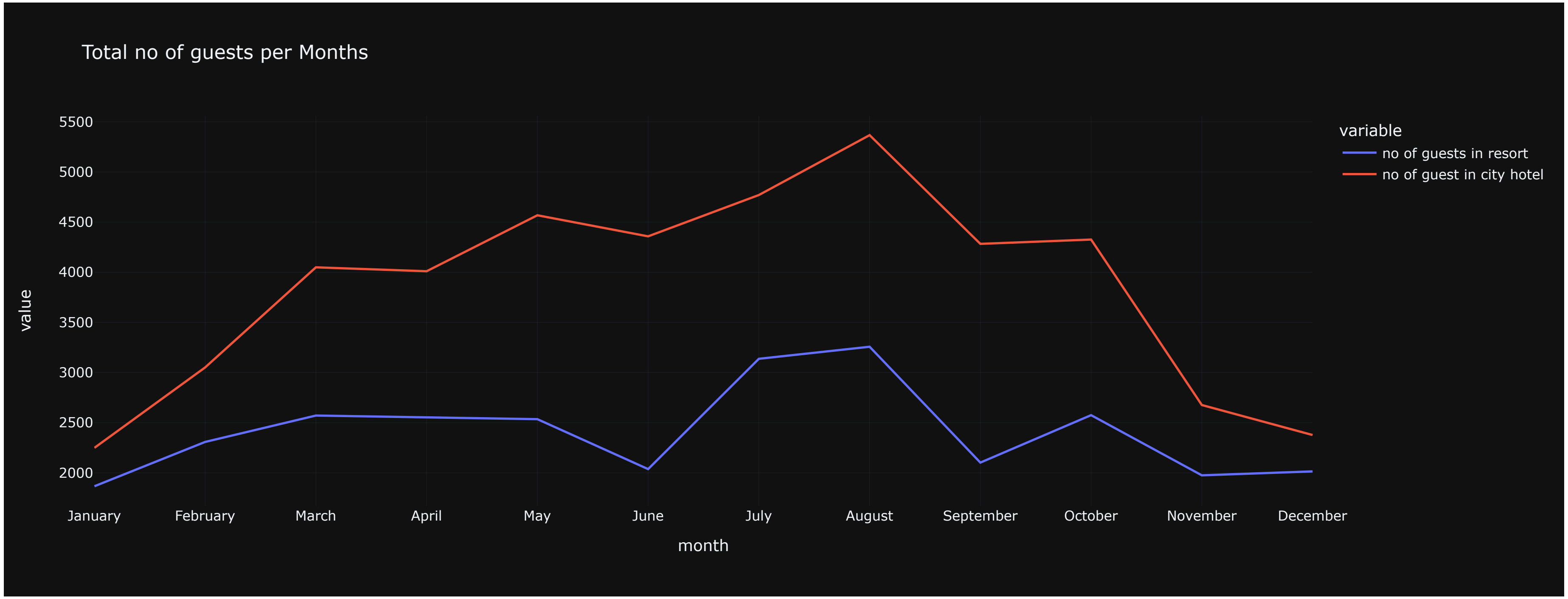
	month	no of guests in resort	no of guest in city hotel
0	August	3257	5367
1	July	3137	4770
2	October	2575	4326
3	March	2571	4049
4	April	2550	4010
5	May	2535	4568
6	February	2308	3051
7	September	2102	4283
8	June	2037	4358
9	December	2014	2377
10	November	1975	2676
11	January	1866	2249

In [25]:
```
final_guests = sort_month(final_guests,'month')
final_guests
```

Out[25]:

	month	no of guests in resort	no of guest in city hotel
0	January	1866	2249
1	February	2308	3051
2	March	2571	4049
3	April	2550	4010
4	May	2535	4568
5	June	2037	4358
6	July	3137	4770
7	August	3257	5367
8	September	2102	4283
9	October	2575	4326
10	November	1975	2676
11	December	2014	2377

In [26]:
```
px.line(final_guests, x = 'month', y = ['no of guests in resort','no of guest in city hotel'],
        title='Total no of guests per Months', template = 'plotly_dark')
```



- The City hotel has more guests during spring and autumn, when the prices are also highest, In July and August there are less visitors, although prices are lower.
- Guest numbers for the Resort hotel go down slighty from June to September, which is also when the prices are highest. Both hotels have the fewest guests during the winter.

How long do people stay at the hotels?

In [27]:
```
filter = df['is_canceled'] == 0
data = df[filter]
data.head()
```

Out[27]:

	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	distributi
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Direct	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Corporate	
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	Online TA	

In [28]:

```
data['total_nights'] = data['stays_in_weekend_nights'] + data['stays_in_week_nights']
data.head()
```

Out[28]:

	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	distributi
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	Direct	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Direct	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Corporate	
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	Online TA	

5 rows × 33 columns

In [29]:

```
stay = data.groupby(['total_nights', 'hotel']).agg('count').reset_index()
stay = stay.iloc[:, :3]
stay = stay.rename(columns={'is_canceled':'Number of stays'})
stay
```

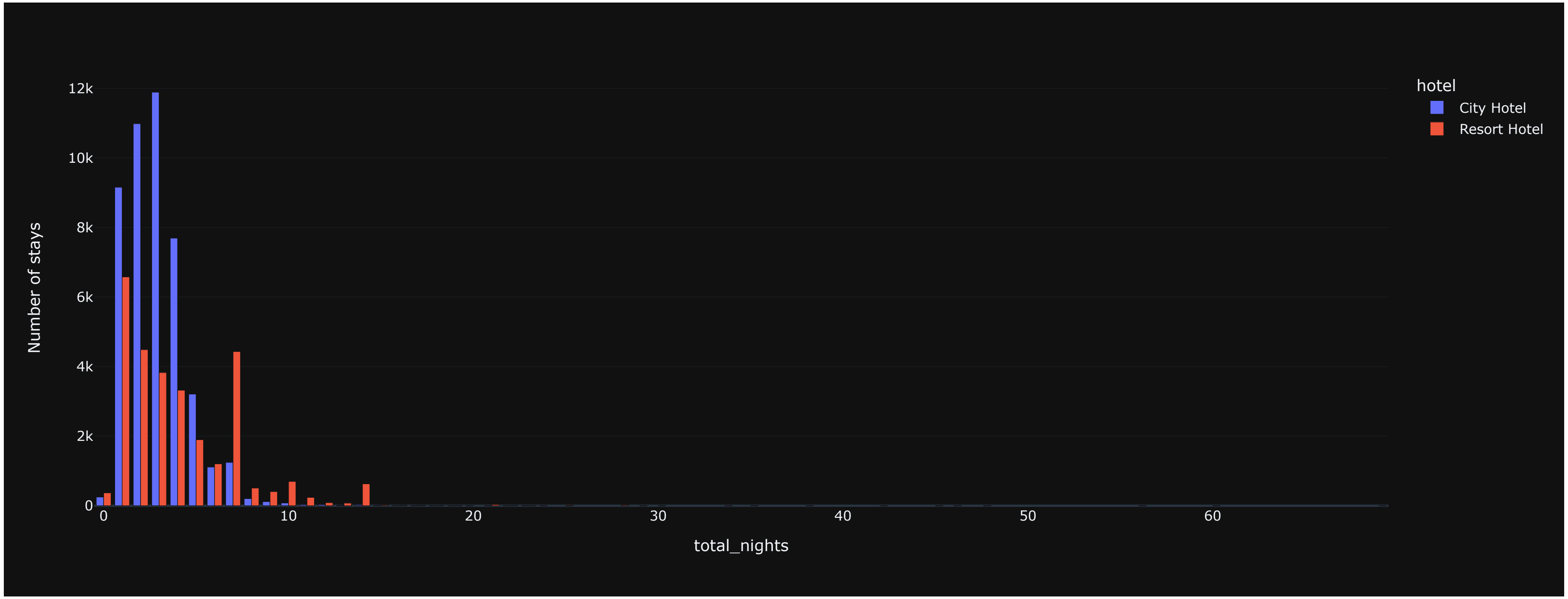
Out[29]:

	total_nights	hotel	Number of stays
0	0	City Hotel	251
1	0	Resort Hotel	371
2	1	City Hotel	9155
3	1	Resort Hotel	6579
4	2	City Hotel	10983
...	...	...	...
57	46	Resort Hotel	1
58	48	City Hotel	1
59	56	Resort Hotel	1
60	60	Resort Hotel	1
61	69	Resort Hotel	1

62 rows × 3 columns

In [30]:

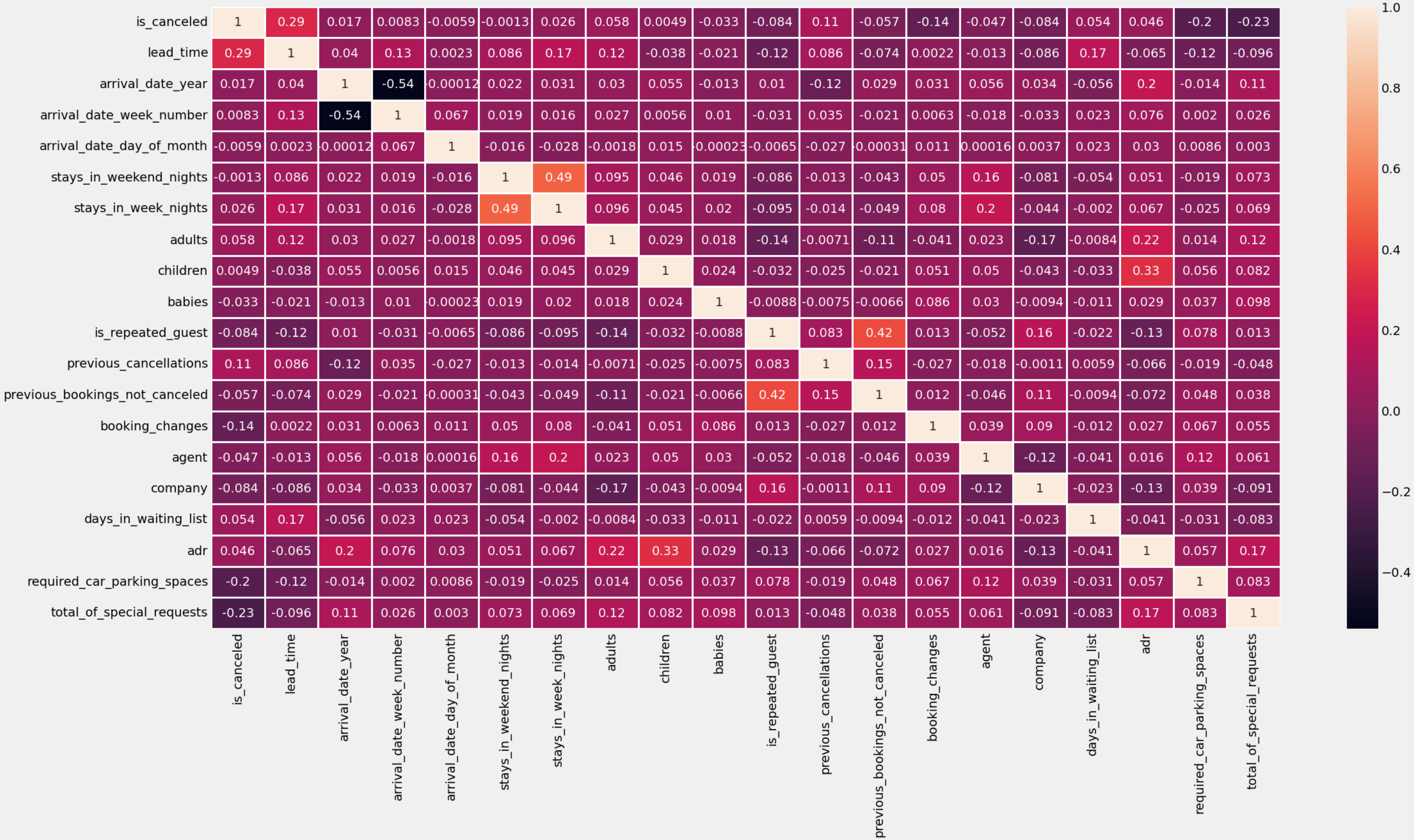
```
px.bar(data_frame = stay, x = 'total_nights', y = 'Number of stays', color = 'hotel', barmode = 'group',
        template = 'plotly_dark')
```



5 DATA PREPROCESSING

In [31]:
```
plt.figure(figsize = (24, 12))

corr = df.corr()
sns.heatmap(corr, annot = True, linewidths = 1)
plt.show()
```



In [32]:
```
correlation = df.corr()['is_canceled'].abs().sort_values(ascending = False)
correlation
```

Out[32]:
```
is_canceled                       1.000000
lead_time                         0.292876
total_of_special_requests         0.234877
required_car_parking_spaces       0.195701
booking_changes                   0.144832
previous_cancellations            0.110139
is_repeated_guest                 0.083745
company                           0.083594
adults                            0.058182
previous_bookings_not_canceled    0.057365
days_in_waiting_list              0.054301
agent                             0.046770
adr                               0.046492
babies                            0.032569
stays_in_week_nights              0.025542
arrival_date_year                 0.016622
arrival_date_week_number          0.008315
arrival_date_day_of_month         0.005948
children                          0.004851
stays_in_weekend_nights           0.001323
Name: is_canceled, dtype: float64
```

In [33]:
```python
# dropping columns that are not useful

useless_col = ['days_in_waiting_list', 'arrival_date_year', 'arrival_date_year', 'assigned_room_type', 'booking_changes',
               'reservation_status', 'country', 'days_in_waiting_list']

df.drop(useless_col, axis = 1, inplace = True)
```

In [34]:
```python
df.head()
```

Out[34]:

	hotel	is_canceled	lead_time	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	meal	market_segment	distribution_channel	is_repeated_gue
0	Resort Hotel	0	342	July	27	1	0	0	2	0.0	0	BB	Direct	Direct	
1	Resort Hotel	0	737	July	27	1	0	0	2	0.0	0	BB	Direct	Direct	
2	Resort Hotel	0	7	July	27	1	0	1	1	0.0	0	BB	Direct	Direct	
3	Resort Hotel	0	13	July	27	1	0	1	1	0.0	0	BB	Corporate	Corporate	
4	Resort Hotel	0	14	July	27	1	0	2	2	0.0	0	BB	Online TA	TA/TO	

In [35]:
```python
# creating numerical and categorical dataframes

cat_cols = [col for col in df.columns if df[col].dtype == 'O']
cat_cols
```

Out[35]:
```
['hotel',
 'arrival_date_month',
 'meal',
 'market_segment',
 'distribution_channel',
 'reserved_room_type',
 'deposit_type',
 'customer_type',
 'reservation_status_date']
```

In [36]:
```python
cat_df = df[cat_cols]
cat_df.head()
```

Out[36]:

	hotel	arrival_date_month	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	reservation_status_date
0	Resort Hotel	July	BB	Direct	Direct	C	No Deposit	Transient	7/1/2015
1	Resort Hotel	July	BB	Direct	Direct	C	No Deposit	Transient	7/1/2015
2	Resort Hotel	July	BB	Direct	Direct	A	No Deposit	Transient	7/2/2015
3	Resort Hotel	July	BB	Corporate	Corporate	A	No Deposit	Transient	7/2/2015
4	Resort Hotel	July	BB	Online TA	TA/TO	A	No Deposit	Transient	7/3/2015

In [37]:

```
cat_df['reservation_status_date'] = pd.to_datetime(cat_df['reservation_status_date'])

cat_df['year'] = cat_df['reservation_status_date'].dt.year
cat_df['month'] = cat_df['reservation_status_date'].dt.month
cat_df['day'] = cat_df['reservation_status_date'].dt.day
```

In [38]:

```
cat_df.drop(['reservation_status_date','arrival_date_month'] , axis = 1, inplace = True)
```

In [39]:

```
cat_df.head()
```

Out[39]:

	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day
0	Resort Hotel	BB	Direct	Direct	C	No Deposit	Transient	2015	7	1
1	Resort Hotel	BB	Direct	Direct	C	No Deposit	Transient	2015	7	1
2	Resort Hotel	BB	Direct	Direct	A	No Deposit	Transient	2015	7	2
3	Resort Hotel	BB	Corporate	Corporate	A	No Deposit	Transient	2015	7	2
4	Resort Hotel	BB	Online TA	TA/TO	A	No Deposit	Transient	2015	7	3

In [40]:

```
# printing unique values of each column
for col in cat_df.columns:
    print(f"{col}: \n{cat_df[col].unique()}\n")
```

```
hotel:
['Resort Hotel' 'City Hotel']

meal:
['BB' 'FB' 'HB' 'SC' 'Undefined']

market_segment:
['Direct' 'Corporate' 'Online TA' 'Offline TA/TO' 'Complementary' 'Groups'
 'Undefined' 'Aviation']

distribution_channel:
['Direct' 'Corporate' 'TA/TO' 'Undefined' 'GDS']

reserved_room_type:
['C' 'A' 'D' 'E' 'G' 'F' 'H' 'L' 'B']

deposit_type:
['No Deposit' 'Refundable' 'Non Refund']

customer_type:
['Transient' 'Contract' 'Transient-Party' 'Group']

year:
[2015 2014 2016 2017]

month:
[ 7  5  4  6  3  8  9  1 11 10 12  2]

day:
[ 1  2  3  6 22 23  5  7  8 11 15 16 29 19 18  9 13  4 12 26 17 10 20 14
 30 28 25 21 27 24 31]
```

In [41]:

```
# encoding categorical variables

cat_df['hotel'] = cat_df['hotel'].map({'Resort Hotel' : 0, 'City Hotel' : 1})

cat_df['meal'] = cat_df['meal'].map({'BB' : 0, 'FB': 1, 'HB': 2, 'SC': 3, 'Undefined': 4})

cat_df['market_segment'] = cat_df['market_segment'].map({'Direct': 0, 'Corporate': 1, 'Online TA': 2, 'Offline TA/TO': 3,
                                                           'Complementary': 4, 'Groups': 5, 'Undefined': 6, 'Aviation': 7})

cat_df['distribution_channel'] = cat_df['distribution_channel'].map({'Direct': 0, 'Corporate': 1, 'TA/TO': 2, 'Undefined': 3,
                                                                       'GDS': 4})

cat_df['reserved_room_type'] = cat_df['reserved_room_type'].map({'C': 0, 'A': 1, 'D': 2, 'E': 3, 'G': 4, 'F': 5, 'H': 6,
                                                                   'L': 7, 'B': 8})

cat_df['deposit_type'] = cat_df['deposit_type'].map({'No Deposit': 0, 'Refundable': 1, 'Non Refund': 3})

cat_df['customer_type'] = cat_df['customer_type'].map({'Transient': 0, 'Contract': 1, 'Transient-Party': 2, 'Group': 3})

cat_df['year'] = cat_df['year'].map({2015: 0, 2014: 1, 2016: 2, 2017: 3})
```

In [42]:

```
num_df = df.drop(columns = cat_cols, axis = 1)
num_df.drop('is_canceled', axis = 1, inplace = True)
num_df
```

Out[42]:

	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	
0	342	27	1	0	0	2	0.0	0	0	0	0	0.0	0.0	
1	737	27	1	0	0	2	0.0	0	0	0	0	0.0	0.0	
2	7	27	1	0	1	1	0.0	0	0	0	0	0.0	0.0	7
3	13	27	1	0	1	1	0.0	0	0	0	0	304.0	0.0	7
4	14	27	1	0	2	2	0.0	0	0	0	0	240.0	0.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
119385	23	35	30	2	5	2	0.0	0	0	0	0	394.0	0.0	9
119386	102	35	31	2	5	3	0.0	0	0	0	0	9.0	0.0	22
119387	34	35	31	2	5	2	0.0	0	0	0	0	9.0	0.0	15
119388	109	35	31	2	5	2	0.0	0	0	0	0	89.0	0.0	10
119389	205	35	29	2	7	2	0.0	0	0	0	0	9.0	0.0	15

119210 rows × 16 columns

In [43]:

```
num_df.var()
```

Out[43]:

```
lead_time                         11422.361808
arrival_date_week_number            184.990111
arrival_date_day_of_month            77.107192
stays_in_weekend_nights               0.990258
stays_in_week_nights                  3.599010
adults                                0.330838
children                              0.159070
babies                                0.009508
is_repeated_guest                     0.030507
previous_cancellations                0.713887
previous_bookings_not_canceled        2.244415
agent                             11485.169679
company                            2897.684308
adr                                2543.589039
required_car_parking_spaces           0.060201
total_of_special_requests             0.628652
dtype: float64
```

In [44]:

```
# normalizing numerical variables

num_df['lead_time'] = np.log(num_df['lead_time'] + 1)
num_df['arrival_date_week_number'] = np.log(num_df['arrival_date_week_number'] + 1)
num_df['arrival_date_day_of_month'] = np.log(num_df['arrival_date_day_of_month'] + 1)
num_df['agent'] = np.log(num_df['agent'] + 1)
num_df['company'] = np.log(num_df['company'] + 1)
num_df['adr'] = np.log(num_df['adr'] + 1)
```

In [45]:

```
num_df.var()
```

Out[45]:
```
lead_time                         2.582757
arrival_date_week_number          0.440884
arrival_date_day_of_month         0.506325
stays_in_weekend_nights           0.990258
stays_in_week_nights              3.599010
adults                            0.330838
children                          0.159070
babies                            0.009508
is_repeated_guest                 0.030507
previous_cancellations            0.713887
previous_bookings_not_canceled    2.244415
agent                             3.535793
company                           1.346883
adr                               0.515480
required_car_parking_spaces       0.060201
total_of_special_requests         0.628652
dtype: float64
```

In [46]:
```
num_df['adr'] = num_df['adr'].fillna(value = num_df['adr'].mean())
```

In [47]:
```
num_df.head()
```

Out[47]:

	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	
0	5.837730	3.332205	0.693147	0	0	2	0.0	0	0	0	0	0.000000	0.0	0.000
1	6.603944	3.332205	0.693147	0	0	2	0.0	0	0	0	0	0.000000	0.0	0.000
2	2.079442	3.332205	0.693147	0	1	1	0.0	0	0	0	0	0.000000	0.0	4.330
3	2.639057	3.332205	0.693147	0	1	1	0.0	0	0	0	0	5.720312	0.0	4.330
4	2.708050	3.332205	0.693147	0	2	2	0.0	0	0	0	0	5.484797	0.0	4.595

In [48]:
```
X = pd.concat([cat_df, num_df], axis = 1)
y = df['is_canceled']
```

In [49]:
```
X.shape, y.shape
```

Out[49]:
```
((119210, 26), (119210,))
```

In [50]:
```
# splitting data into training set and test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

In [51]:
```
X_train.head()
```

Out[51]:

	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day	lead_time	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_night
66840	1	0	5	2	1	3	0	2	11	21	5.056246	2.890372	3.295837	0	
65034	1	0	5	2	1	3	0	0	10	21	6.434547	2.564949	3.178054	0	
3316	0	0	3	2	1	0	0	0	12	7	3.295837	3.912023	1.791759	1	
118781	1	0	2	2	1	0	0	3	8	25	5.129899	3.555348	3.044522	2	
109942	1	3	2	2	1	0	0	3	4	19	2.484907	2.772589	2.708050	2	

In [52]:
```
X_test.head()
```

Out[52]:

	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day	lead_time	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_night
69249	1	0	2	2	2	0	0	3	1	28	4.804021	3.135494	3.367296	2	
21712	0	2	0	0	3	0	2	3	3	19	3.295837	2.484907	2.708050	0	
94153	1	2	3	2	1	0	2	2	8	1	5.710427	3.465736	3.433987	1	
8413	0	2	5	2	1	3	0	2	1	22	5.828946	3.713572	3.295837	3	
109949	1	0	2	2	1	0	0	3	4	20	3.555348	2.833213	2.833213	2	

In [53]:

```
y_train.head(), y_test.head()
```

Out[53]:

```
(66840     1
 65034     1
 3316      0
 118781    0
 109942    0
 Name: is_canceled, dtype: int64,
 69249     1
 21712     0
 94153     0
 8413      1
 109949    0
 Name: is_canceled, dtype: int64)
```

5 MODEL BUILDING

LOGISTIC REGRESSION

In [54]:

```
lr = LogisticRegression()
lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

acc_lr = accuracy_score(y_test, y_pred_lr)
conf = confusion_matrix(y_test, y_pred_lr)
clf_report = classification_report(y_test, y_pred_lr)

print(f"Accuracy Score of Logistic Regression is : {acc_lr}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Logistic Regression is : 0.8113133685652769
Confusion Matrix : 
[[21408  1084]
 [ 5664  7607]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.79      0.95      0.86     22492
           1       0.88      0.57      0.69     13271

    accuracy                           0.81     35763
   macro avg       0.83      0.76      0.78     35763
weighted avg       0.82      0.81      0.80     35763
```

KNN

In [55]:
```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

acc_knn = accuracy_score(y_test, y_pred_knn)
conf = confusion_matrix(y_test, y_pred_knn)
clf_report = classification_report(y_test, y_pred_knn)

print(f"Accuracy Score of KNN is : {acc_knn}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of KNN is : 0.8909207840505551
Confusion Matrix :
[[21763   729]
 [ 3172 10099]]
Classification Report :
              precision    recall  f1-score   support

           0       0.87      0.97      0.92     22492
           1       0.93      0.76      0.84     13271

    accuracy                           0.89     35763
   macro avg       0.90      0.86      0.88     35763
weighted avg       0.90      0.89      0.89     35763
```

DECISION TREE CLASSIFIER

In [56]:
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

y_pred_dtc = dtc.predict(X_test)

acc_dtc = accuracy_score(y_test, y_pred_dtc)
conf = confusion_matrix(y_test, y_pred_dtc)
clf_report = classification_report(y_test, y_pred_dtc)

print(f"Accuracy Score of Decision Tree is : {acc_dtc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Decision Tree is : 0.9444677459944636
Confusion Matrix :
[[21520   972]
 [ 1014 12257]]
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     22492
           1       0.93      0.92      0.93     13271

    accuracy                           0.94     35763
   macro avg       0.94      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

RANDOM FOREST CLASSIFIER

In [57]:
```python
rd_clf = RandomForestClassifier()
rd_clf.fit(X_train, y_train)

y_pred_rd_clf = rd_clf.predict(X_test)

acc_rd_clf = accuracy_score(y_test, y_pred_rd_clf)
conf = confusion_matrix(y_test, y_pred_rd_clf)
clf_report = classification_report(y_test, y_pred_rd_clf)

print(f"Accuracy Score of Random Forest is : {acc_rd_clf}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Random Forest is : 0.9526605709811816
Confusion Matrix : 
[[22289   203]
 [ 1490 11781]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     22492
           1       0.98      0.89      0.93     13271

    accuracy                           0.95     35763
   macro avg       0.96      0.94      0.95     35763
weighted avg       0.95      0.95      0.95     35763
```

ADA BOOST CLASSIFIER

In [58]:
```python
ada = AdaBoostClassifier(base_estimator = dtc)
ada.fit(X_train, y_train)

y_pred_ada = ada.predict(X_test)

acc_ada = accuracy_score(y_test, y_pred_ada)
conf = confusion_matrix(y_test, y_pred_ada)
clf_report = classification_report(y_test, y_pred_ada)

print(f"Accuracy Score of Ada Boost Classifier is : {acc_ada}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9442999748343259
Confusion Matrix : 
[[21491  1001]
 [  991 12280]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     22492
           1       0.92      0.93      0.92     13271

    accuracy                           0.94     35763
   macro avg       0.94      0.94      0.94     35763
weighted avg       0.94      0.94      0.94     35763
```

GRADIENT BOOSTING CLASSIFIER

In [59]:
```
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)

y_pred_gb = gb.predict(X_test)

acc_gb = accuracy_score(y_test, y_pred_gb)
conf = confusion_matrix(y_test, y_pred_gb)
clf_report = classification_report(y_test, y_pred_gb)

print(f"Accuracy Score of Gradient Boosting Classifier is : {acc_gb}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Gradient Boosting Classifier is : 0.9063836926432346
Confusion Matrix :
[[22224   268]
 [ 3080 10191]]
Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.99      0.93     22492
           1       0.97      0.77      0.86     13271

    accuracy                           0.91     35763
   macro avg       0.93      0.88      0.89     35763
weighted avg       0.91      0.91      0.90     35763
```

XGBOOST CLASSIFIER

In [60]:
```
xgb = XGBClassifier(booster = 'gbtree', learning_rate = 0.1, max_depth = 5, n_estimators = 180)
xgb.fit(X_train, y_train)

y_pred_xgb = xgb.predict(X_test)

acc_xgb = accuracy_score(y_test, y_pred_xgb)
conf = confusion_matrix(y_test, y_pred_xgb)
clf_report = classification_report(y_test, y_pred_xgb)

print(f"Accuracy Score of XGBoost Classifier is : {acc_xgb}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of XGBoost Classifier is : 0.9824119900455778
Confusion Matrix :
[[22477    15]
 [  614 12657]]
Classification Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     22492
           1       1.00      0.95      0.98     13271

    accuracy                           0.98     35763
   macro avg       0.99      0.98      0.98     35763
weighted avg       0.98      0.98      0.98     35763
```

CAT BOOST CLASSIFIER

In [61]:
```
cat = CatBoostClassifier(iterations=100)
cat.fit(X_train, y_train)

y_pred_cat = cat.predict(X_test)

acc_cat = accuracy_score(y_test, y_pred_cat)
conf = confusion_matrix(y_test, y_pred_cat)
clf_report = classification_report(y_test, y_pred_cat)
```

```
Learning rate set to 0.5
0:	learn: 0.4635888	total: 216ms	remaining: 21.4s
1:	learn: 0.4053135	total: 251ms	remaining: 12.3s
2:	learn: 0.3751804	total: 283ms	remaining: 9.16s
3:	learn: 0.3320807	total: 318ms	remaining: 7.64s
4:	learn: 0.3097690	total: 354ms	remaining: 6.74s
5:	learn: 0.2849091	total: 386ms	remaining: 6.05s
6:	learn: 0.2509375	total: 417ms	remaining: 5.54s
7:	learn: 0.2379493	total: 441ms	remaining: 5.07s
8:	learn: 0.2211308	total: 469ms	remaining: 4.74s
9:	learn: 0.1834510	total: 500ms	remaining: 4.5s
10:	learn: 0.1678027	total: 531ms	remaining: 4.29s
11:	learn: 0.1576576	total: 562ms	remaining: 4.12s
12:	learn: 0.1524325	total: 592ms	remaining: 3.96s
13:	learn: 0.1408135	total: 623ms	remaining: 3.83s
14:	learn: 0.1322983	total: 657ms	remaining: 3.72s
15:	learn: 0.1220751	total: 686ms	remaining: 3.6s
16:	learn: 0.1102169	total: 717ms	remaining: 3.5s
17:	learn: 0.1043466	total: 748ms	remaining: 3.4s
18:	learn: 0.1014863	total: 775ms	remaining: 3.31s
19:	learn: 0.0978337	total: 805ms	remaining: 3.22s
20:	learn: 0.0944187	total: 838ms	remaining: 3.15s
21:	learn: 0.0934548	total: 868ms	remaining: 3.08s
22:	learn: 0.0888218	total: 897ms	remaining: 3s
23:	learn: 0.0858455	total: 929ms	remaining: 2.94s
24:	learn: 0.0844769	total: 959ms	remaining: 2.88s
25:	learn: 0.0817192	total: 990ms	remaining: 2.82s
26:	learn: 0.0793475	total: 1.02s	remaining: 2.76s
27:	learn: 0.0759774	total: 1.05s	remaining: 2.7s
28:	learn: 0.0726175	total: 1.08s	remaining: 2.65s
29:	learn: 0.0689331	total: 1.11s	remaining: 2.6s
30:	learn: 0.0673316	total: 1.14s	remaining: 2.54s
31:	learn: 0.0645945	total: 1.17s	remaining: 2.49s
32:	learn: 0.0623087	total: 1.2s	remaining: 2.44s
33:	learn: 0.0599571	total: 1.23s	remaining: 2.39s
34:	learn: 0.0579193	total: 1.26s	remaining: 2.34s
35:	learn: 0.0567986	total: 1.29s	remaining: 2.3s
36:	learn: 0.0552332	total: 1.32s	remaining: 2.25s
37:	learn: 0.0539433	total: 1.35s	remaining: 2.2s
38:	learn: 0.0534948	total: 1.38s	remaining: 2.15s
39:	learn: 0.0523618	total: 1.41s	remaining: 2.11s
40:	learn: 0.0513314	total: 1.44s	remaining: 2.06s
41:	learn: 0.0489160	total: 1.47s	remaining: 2.02s
42:	learn: 0.0474929	total: 1.5s	remaining: 1.99s
43:	learn: 0.0469116	total: 1.53s	remaining: 1.94s
44:	learn: 0.0462935	total: 1.56s	remaining: 1.91s
45:	learn: 0.0447441	total: 1.59s	remaining: 1.86s
46:	learn: 0.0443588	total: 1.62s	remaining: 1.82s
47:	learn: 0.0435778	total: 1.65s	remaining: 1.78s
48:	learn: 0.0421251	total: 1.68s	remaining: 1.75s
49:	learn: 0.0403468	total: 1.71s	remaining: 1.71s
50:	learn: 0.0393920	total: 1.74s	remaining: 1.67s
51:	learn: 0.0391944	total: 1.77s	remaining: 1.63s
52:	learn: 0.0381996	total: 1.79s	remaining: 1.59s
53:	learn: 0.0378151	total: 1.82s	remaining: 1.55s
54:	learn: 0.0369584	total: 1.85s	remaining: 1.52s
55:	learn: 0.0359041	total: 1.89s	remaining: 1.48s
56:	learn: 0.0354511	total: 1.92s	remaining: 1.45s
57:	learn: 0.0349777	total: 1.94s	remaining: 1.41s
58:	learn: 0.0344471	total: 1.97s	remaining: 1.37s
59:	learn: 0.0342530	total: 2s	remaining: 1.33s
60:	learn: 0.0331533	total: 2.03s	remaining: 1.29s
61:	learn: 0.0317613	total: 2.06s	remaining: 1.26s
62:	learn: 0.0314239	total: 2.09s	remaining: 1.23s
63:	learn: 0.0300242	total: 2.12s	remaining: 1.19s
64:	learn: 0.0298043	total: 2.15s	remaining: 1.16s
```

```
65:	learn: 0.0295827	total: 2.17s	remaining: 1.12s
66:	learn: 0.0288499	total: 2.2s	remaining: 1.08s
67:	learn: 0.0282404	total: 2.23s	remaining: 1.05s
68:	learn: 0.0276580	total: 2.26s	remaining: 1.02s
69:	learn: 0.0273049	total: 2.29s	remaining: 983ms
70:	learn: 0.0259852	total: 2.33s	remaining: 950ms
71:	learn: 0.0257488	total: 2.35s	remaining: 916ms
72:	learn: 0.0248966	total: 2.39s	remaining: 883ms
73:	learn: 0.0241093	total: 2.42s	remaining: 850ms
74:	learn: 0.0239690	total: 2.45s	remaining: 816ms
75:	learn: 0.0236815	total: 2.48s	remaining: 782ms
76:	learn: 0.0235566	total: 2.5s	remaining: 748ms
77:	learn: 0.0232616	total: 2.53s	remaining: 715ms
78:	learn: 0.0228137	total: 2.57s	remaining: 682ms
79:	learn: 0.0220771	total: 2.6s	remaining: 649ms
80:	learn: 0.0218928	total: 2.62s	remaining: 615ms
81:	learn: 0.0213718	total: 2.65s	remaining: 582ms
82:	learn: 0.0212126	total: 2.68s	remaining: 549ms
83:	learn: 0.0210830	total: 2.71s	remaining: 516ms
84:	learn: 0.0209067	total: 2.74s	remaining: 483ms
85:	learn: 0.0204775	total: 2.77s	remaining: 451ms
86:	learn: 0.0198129	total: 2.8s	remaining: 419ms
87:	learn: 0.0192032	total: 2.83s	remaining: 386ms
88:	learn: 0.0187687	total: 2.86s	remaining: 353ms
89:	learn: 0.0182871	total: 2.89s	remaining: 321ms
90:	learn: 0.0179069	total: 2.92s	remaining: 288ms
91:	learn: 0.0178731	total: 2.94s	remaining: 256ms
92:	learn: 0.0172815	total: 2.98s	remaining: 224ms
93:	learn: 0.0168985	total: 3.01s	remaining: 192ms
94:	learn: 0.0167478	total: 3.04s	remaining: 160ms
95:	learn: 0.0166471	total: 3.07s	remaining: 128ms
96:	learn: 0.0163321	total: 3.09s	remaining: 95.6ms
97:	learn: 0.0156219	total: 3.13s	remaining: 63.8ms
98:	learn: 0.0151412	total: 3.16s	remaining: 31.9ms
99:	learn: 0.0146907	total: 3.19s	remaining: 0us
```

In [62]:
```python
print(f"Accuracy Score of Ada Boost Classifier is : {acc_cat}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9955261023963314
Confusion Matrix : 
[[22477    15]
 [  145 13126]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     22492
           1       1.00      0.99      0.99     13271

    accuracy                           1.00     35763
   macro avg       1.00      0.99      1.00     35763
weighted avg       1.00      1.00      1.00     35763
```

EXTRA TREES CLASSIFIER

In [63]:
```python
etc = ExtraTreesClassifier()
etc.fit(X_train, y_train)

y_pred_etc = etc.predict(X_test)

acc_etc = accuracy_score(y_test, y_pred_etc)
conf = confusion_matrix(y_test, y_pred_etc)
```

```
clf_report = classification_report(y_test, y_pred_etc)

print(f"Accuracy Score of Extra Trees Classifier is : {acc_etc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Extra Trees Classifier is : 0.9520733719207002
Confusion Matrix : 
[[22286   206]
 [ 1508 11763]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     22492
           1       0.98      0.89      0.93     13271

    accuracy                           0.95     35763
   macro avg       0.96      0.94      0.95     35763
weighted avg       0.95      0.95      0.95     35763
```

LGBM CLASSIFIER

In [64]:
```
lgbm = LGBMClassifier(learning_rate = 1)
lgbm.fit(X_train, y_train)

y_pred_lgbm = lgbm.predict(X_test)

acc_lgbm = accuracy_score(y_test, y_pred_lgbm)
conf = confusion_matrix(y_test, y_pred_lgbm)
clf_report = classification_report(y_test, y_pred_lgbm)

print(f"Accuracy Score of LGBM Classifier is : {acc_lgbm}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
[LightGBM] [Info] Number of positive: 30928, number of negative: 52519
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.016095 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1202
[LightGBM] [Info] Number of data points in the train set: 83447, number of used features: 26
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.370630 -> initscore=-0.529513
[LightGBM] [Info] Start training from score -0.529513
Accuracy Score of LGBM Classifier is : 0.9619159466487711
Confusion Matrix : 
[[21864   628]
 [  734 12537]]
Classification Report : 
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     22492
           1       0.95      0.94      0.95     13271

    accuracy                           0.96     35763
   macro avg       0.96      0.96      0.96     35763
weighted avg       0.96      0.96      0.96     35763
```

VOTING CLASSIFIER

In [65]:
```
classifiers = [('Gradient Boosting Classifier', gb), ('Cat Boost Classifier', cat), ('XGboost', xgb),  ('Decision Tree', dtc),
               ('Extra Tree', etc), ('Light Gradient', lgbm), ('Random Forest', rd_clf), ('Ada Boost', ada), ('Logistic', lr),
               ('Knn', knn)]
vc = VotingClassifier(estimators = classifiers)
vc.fit(X_train, y_train)
```

```
Learning rate set to 0.5
0:	learn: 0.4635888	total: 32ms	remaining: 3.16s
1:	learn: 0.4053135	total: 52.3ms	remaining: 2.56s
2:	learn: 0.3751804	total: 69.1ms	remaining: 2.23s
3:	learn: 0.3320807	total: 92.5ms	remaining: 2.22s
4:	learn: 0.3097690	total: 114ms	remaining: 2.16s
5:	learn: 0.2849091	total: 134ms	remaining: 2.1s
6:	learn: 0.2509375	total: 155ms	remaining: 2.06s
7:	learn: 0.2379493	total: 176ms	remaining: 2.02s
8:	learn: 0.2211308	total: 201ms	remaining: 2.03s
9:	learn: 0.1834510	total: 235ms	remaining: 2.12s
10:	learn: 0.1678027	total: 260ms	remaining: 2.1s
11:	learn: 0.1576576	total: 280ms	remaining: 2.05s
12:	learn: 0.1524325	total: 301ms	remaining: 2.02s
13:	learn: 0.1408135	total: 322ms	remaining: 1.98s
14:	learn: 0.1322983	total: 342ms	remaining: 1.94s
15:	learn: 0.1220751	total: 365ms	remaining: 1.92s
16:	learn: 0.1102169	total: 386ms	remaining: 1.89s
17:	learn: 0.1043466	total: 407ms	remaining: 1.85s
18:	learn: 0.1014863	total: 429ms	remaining: 1.83s
19:	learn: 0.0978337	total: 458ms	remaining: 1.83s
20:	learn: 0.0944187	total: 481ms	remaining: 1.81s
21:	learn: 0.0934548	total: 505ms	remaining: 1.79s
22:	learn: 0.0888218	total: 524ms	remaining: 1.75s
23:	learn: 0.0858455	total: 545ms	remaining: 1.73s
24:	learn: 0.0844769	total: 565ms	remaining: 1.7s
25:	learn: 0.0817192	total: 593ms	remaining: 1.69s
26:	learn: 0.0793475	total: 615ms	remaining: 1.66s
27:	learn: 0.0759774	total: 638ms	remaining: 1.64s
28:	learn: 0.0726175	total: 664ms	remaining: 1.63s
29:	learn: 0.0689331	total: 698ms	remaining: 1.63s
30:	learn: 0.0673316	total: 725ms	remaining: 1.61s
31:	learn: 0.0645945	total: 756ms	remaining: 1.61s
32:	learn: 0.0623087	total: 789ms	remaining: 1.6s
33:	learn: 0.0599571	total: 819ms	remaining: 1.59s
34:	learn: 0.0579193	total: 850ms	remaining: 1.58s
35:	learn: 0.0567986	total: 879ms	remaining: 1.56s
36:	learn: 0.0552332	total: 908ms	remaining: 1.55s
37:	learn: 0.0539433	total: 936ms	remaining: 1.53s
38:	learn: 0.0534948	total: 964ms	remaining: 1.51s
39:	learn: 0.0523618	total: 996ms	remaining: 1.49s
40:	learn: 0.0513314	total: 1.03s	remaining: 1.48s
41:	learn: 0.0489160	total: 1.06s	remaining: 1.47s
42:	learn: 0.0474929	total: 1.09s	remaining: 1.44s
43:	learn: 0.0469116	total: 1.12s	remaining: 1.43s
44:	learn: 0.0462935	total: 1.15s	remaining: 1.41s
45:	learn: 0.0447441	total: 1.18s	remaining: 1.38s
46:	learn: 0.0443588	total: 1.21s	remaining: 1.36s
47:	learn: 0.0435778	total: 1.24s	remaining: 1.34s
48:	learn: 0.0421251	total: 1.27s	remaining: 1.33s
49:	learn: 0.0403468	total: 1.3s	remaining: 1.3s
50:	learn: 0.0393920	total: 1.33s	remaining: 1.28s
51:	learn: 0.0391944	total: 1.36s	remaining: 1.25s
52:	learn: 0.0381996	total: 1.39s	remaining: 1.23s
53:	learn: 0.0378151	total: 1.42s	remaining: 1.21s
54:	learn: 0.0369584	total: 1.45s	remaining: 1.18s
55:	learn: 0.0359041	total: 1.48s	remaining: 1.16s
56:	learn: 0.0354511	total: 1.5s	remaining: 1.13s
57:	learn: 0.0349777	total: 1.53s	remaining: 1.11s
58:	learn: 0.0344471	total: 1.56s	remaining: 1.08s
59:	learn: 0.0342530	total: 1.59s	remaining: 1.06s
60:	learn: 0.0331533	total: 1.62s	remaining: 1.03s
61:	learn: 0.0317613	total: 1.65s	remaining: 1.01s
62:	learn: 0.0314239	total: 1.68s	remaining: 986ms
63:	learn: 0.0300242	total: 1.71s	remaining: 961ms
64:	learn: 0.0298043	total: 1.74s	remaining: 936ms
```

```
65:	learn: 0.0295827	total: 1.76s	remaining: 910ms
66:	learn: 0.0288499	total: 1.79s	remaining: 885ms
67:	learn: 0.0282404	total: 1.83s	remaining: 860ms
68:	learn: 0.0276580	total: 1.86s	remaining: 835ms
69:	learn: 0.0273049	total: 1.89s	remaining: 809ms
70:	learn: 0.0259852	total: 1.92s	remaining: 785ms
71:	learn: 0.0257488	total: 1.95s	remaining: 757ms
72:	learn: 0.0248966	total: 1.98s	remaining: 731ms
73:	learn: 0.0241093	total: 2.01s	remaining: 705ms
74:	learn: 0.0239690	total: 2.04s	remaining: 679ms
75:	learn: 0.0236815	total: 2.07s	remaining: 653ms
76:	learn: 0.0235566	total: 2.1s	remaining: 626ms
77:	learn: 0.0232616	total: 2.12s	remaining: 599ms
78:	learn: 0.0228137	total: 2.15s	remaining: 573ms
79:	learn: 0.0220771	total: 2.18s	remaining: 546ms
80:	learn: 0.0218928	total: 2.21s	remaining: 519ms
81:	learn: 0.0213718	total: 2.24s	remaining: 492ms
82:	learn: 0.0212126	total: 2.27s	remaining: 465ms
83:	learn: 0.0210830	total: 2.3s	remaining: 438ms
84:	learn: 0.0209067	total: 2.33s	remaining: 411ms
85:	learn: 0.0204775	total: 2.36s	remaining: 384ms
86:	learn: 0.0198129	total: 2.39s	remaining: 357ms
87:	learn: 0.0192032	total: 2.42s	remaining: 330ms
88:	learn: 0.0187687	total: 2.45s	remaining: 303ms
89:	learn: 0.0182871	total: 2.48s	remaining: 276ms
90:	learn: 0.0179069	total: 2.51s	remaining: 248ms
91:	learn: 0.0178731	total: 2.54s	remaining: 221ms
92:	learn: 0.0172815	total: 2.57s	remaining: 193ms
93:	learn: 0.0168985	total: 2.59s	remaining: 166ms
94:	learn: 0.0167478	total: 2.62s	remaining: 138ms
95:	learn: 0.0166471	total: 2.65s	remaining: 110ms
96:	learn: 0.0163321	total: 2.68s	remaining: 83ms
97:	learn: 0.0156219	total: 2.71s	remaining: 55.4ms
98:	learn: 0.0151412	total: 2.75s	remaining: 27.8ms
99:	learn: 0.0146907	total: 2.78s	remaining: 0us
[LightGBM] [Info] Number of positive: 30928, number of negative: 52519
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.013730 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1202
[LightGBM] [Info] Number of data points in the train set: 83447, number of used features: 26
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.370630 -> initscore=-0.529513
[LightGBM] [Info] Start training from score -0.529513
```

Out[65]:

VotingClassifier

Gradient Boosting Classifier	Cat Boost Classifier	XGboost	Decision Tree	Extra Tree	Light Gradient	Random Forest	Ada Boost	Logis
GradientBoostingClassifier	CatBoostClassifier	XGBClassifier	DecisionTreeClassifier	ExtraTreesClassifier	LGBMClassifier	RandomForestClassifier	base_estimator: DecisionTreeClassifier DecisionTreeClassifier	LogisticRe

In [66]:

```python
y_pred_vc = vc.predict(X_test)

acc_vtc = accuracy_score(y_test, y_pred_vc)
conf = confusion_matrix(y_test, y_pred_vc)
clf_report = classification_report(y_test, y_pred_vc)

print(f"Accuracy Score of Voting Classifier is : {acc_vtc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Voting Classifier is : 0.9637614294102844
Confusion Matrix :
[[22472    20]
 [ 1276 11995]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     22492
           1       1.00      0.90      0.95     13271

    accuracy                           0.96     35763
   macro avg       0.97      0.95      0.96     35763
weighted avg       0.97      0.96      0.96     35763
```

MODELS COMPARISON

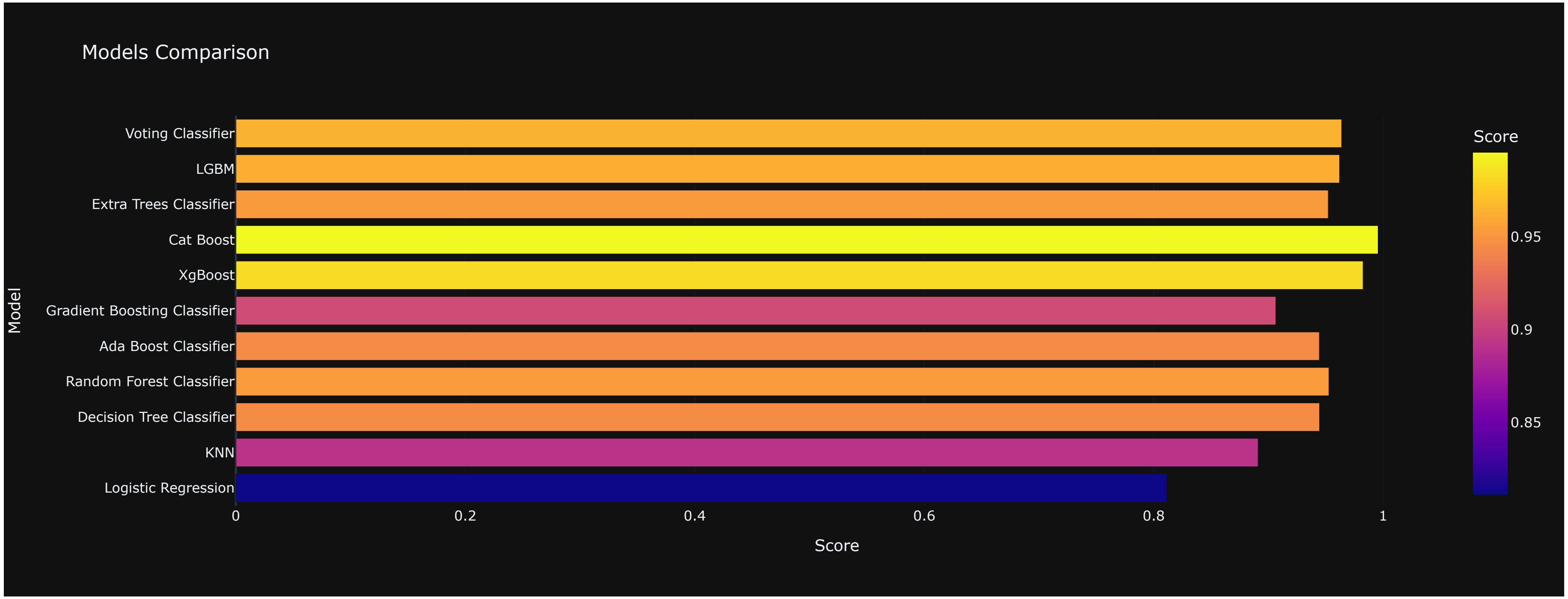
In [67]:
```
models = pd.DataFrame({
    'Model' : ['Logistic Regression', 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier','Ada Boost Classifier',
             'Gradient Boosting Classifier', 'XgBoost', 'Cat Boost', 'Extra Trees Classifier', 'LGBM', 'Voting Classifier'
               ],
    'Score' : [acc_lr, acc_knn, acc_dtc, acc_rd_clf, acc_ada, acc_gb, acc_xgb, acc_cat, acc_etc, acc_lgbm, acc_vtc]
})


models.sort_values(by = 'Score', ascending = False)
```

Out[67]:

	Model	Score
7	Cat Boost	0.995526
6	XgBoost	0.982412
10	Voting Classifier	0.963761
9	LGBM	0.961916
3	Random Forest Classifier	0.952661
8	Extra Trees Classifier	0.952073
2	Decision Tree Classifier	0.944468
4	Ada Boost Classifier	0.944300
5	Gradient Boosting Classifier	0.906384
1	KNN	0.890921
0	Logistic Regression	0.811313

In [68]:
```
fig = px.bar(data_frame = models, x = 'Score', y = 'Model', color = 'Score', template = 'plotly_dark', title = 'Models Comparison')
pyo.iplot(fig)
```



We got accuracy score of 99.5%