Hotel Booking Cancellation EDA and Prediction

In [2]:
```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno

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
```

In [3]:
```
# reading data
df = pd.read_csv('../input/hotel-booking-demand/hotel_bookings.csv')
df.head()
```

Out[3]:

	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_se
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Co
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	On

In [4]:
```
df.describe()
```

Out[4]:

	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	pr
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119386.000000	119390.000000	119390.000000	
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.927599	2.500302	1.856403	0.103890	0.007949	0.031912	
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613	1.908286	0.579261	0.398561	0.097436	0.175767	
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000	2.000000	2.000000	0.000000	0.000000	0.000000	
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000	3.000000	2.000000	0.000000	0.000000	0.000000	
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000	50.000000	55.000000	10.000000	10.000000	1.000000	

In [5]:
```
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
```

In [6]:
```
# checking for null values

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

Out[6]:

	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 [7]:
```
# filling null values with zero

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

In [9]:
```
# 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[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	mar
2224	Resort Hotel	0	1	2015	October	41	6	0	3	0	0.0	0	SC	PRT	
2409	Resort Hotel	0	0	2015	October	42	12	0	0	0	0.0	0	SC	PRT	
3181	Resort Hotel	0	36	2015	November	47	20	1	2	0	0.0	0	SC	ESP	
3684	Resort Hotel	0	165	2015	December	53	30	1	4	0	0.0	0	SC	PRT	
3708	Resort Hotel	0	165	2015	December	53	30	2	4	0	0.0	0	SC	PRT	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
115029	City Hotel	0	107	2017	June	26	27	0	3	0	0.0	0	BB	CHE	
115091	City Hotel	0	1	2017	June	26	30	0	1	0	0.0	0	SC	PRT	Co
116251	City Hotel	0	44	2017	July	28	15	1	1	0	0.0	0	SC	SWE	
116534	City Hotel	0	2	2017	July	28	15	2	5	0	0.0	0	SC	RUS	
117087	City Hotel	0	170	2017	July	30	27	0	2	0	0.0	0	BB	BRA	

180 rows × 32 columns

In [10]:
```
df = df[~filter]
df
```

Out[10]:

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

119210 rows × 32 columns

Exploratory Data Analysis (EDA)

From where the most guests are coming?

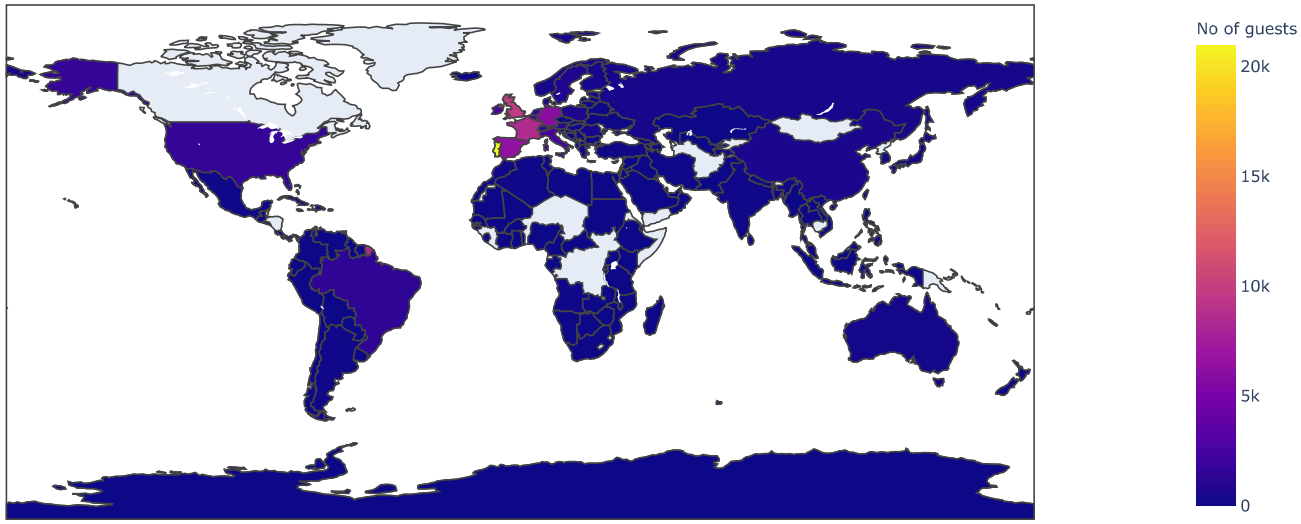
In [11]:
```
country_wise_guests = df[df['is_canceled'] == 0]['country'].value_counts().reset_index()
country_wise_guests.columns = ['country', 'No of guests']
country_wise_guests
```

Out[11]:

	country	No of guests
0	PRT	20977
1	GBR	9668
2	FRA	8468
3	ESP	6383
4	DEU	6067
...	...	...
161	BDI	1
162	DJI	1
163	DMA	1
164	LCA	1
165	MAC	1

166 rows × 2 columns

In [12]:
```
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()
```



How much do guests pay for a room per night?

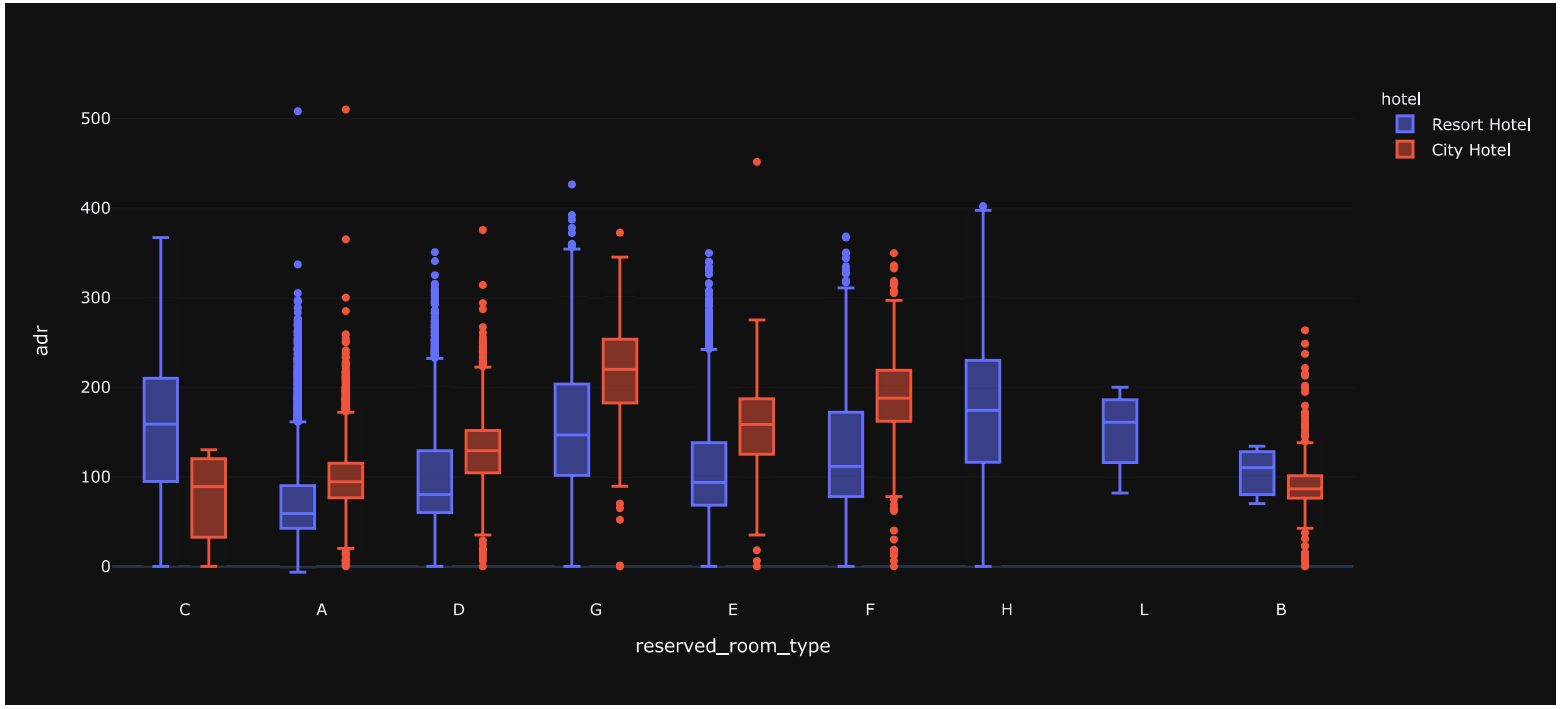
In [13]:
```
df.head()
```

Out[13]:

	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_se
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Co
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	On

In [14]:
```
data = df[df['is_canceled'] == 0]

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



How does the price vary per night over the year?

In [15]:
```
data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] == 0)]
data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] == 0)]
```

In [16]:
```
resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'].mean().reset_index()
resort_hotel
```

Out[16]:

	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 [17]:
```
city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index()
city_hotel
```

Out[17]:

	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 [18]:
```
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[18]:

	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

In [20]:
```
def sort_month(df, column_name):
    return sd.Sort_Dataframeby_Month(df, column_name)
```

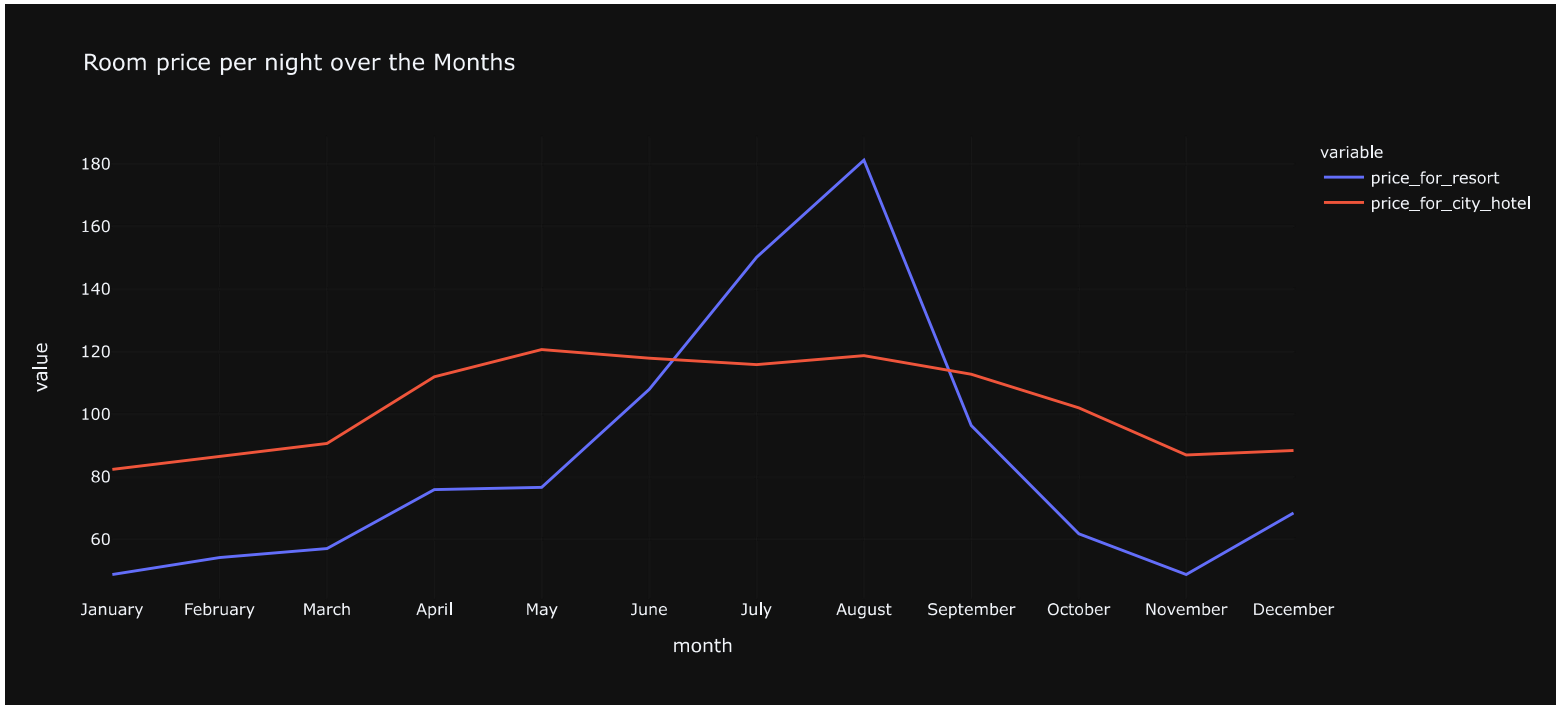
In [21]:
```
final_prices = sort_month(final_hotel, 'month')
final_prices
```

Out[21]:

	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 [22]:
```
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 1224x576 with 0 Axes>

Which are the most busy months?

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

Out[23]:

	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 [24]:
```
city_guests = data_city['arrival_date_month'].value_counts().reset_index()
city_guests.columns=['month','no of guests']
city_guests
```

Out[24]:

	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 [25]:
```
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[25]:

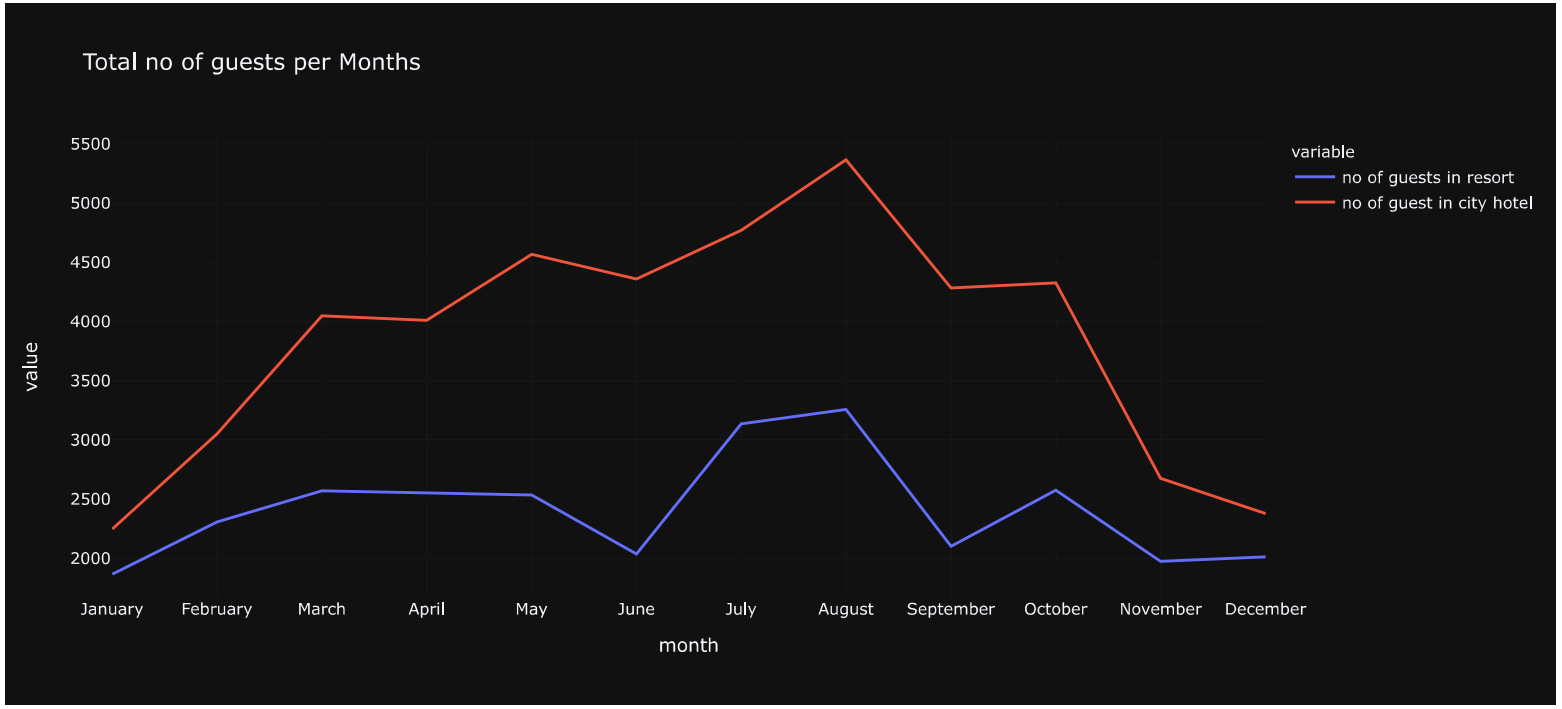
	month	no of guests in resort	no of guest in city hotel
0	August	3257	5367
1	July	3137	4770
2	October	2575	4326
3	March	2571	4049
4	April	2550	4010
5	May	2535	4568
6	February	2308	3051
7	September	2102	4283
8	June	2037	4358
9	December	2014	2377
10	November	1975	2676
11	January	1866	2249

In [26]:
```
final_guests = sort_month(final_guests,'month')
final_guests
```

Out[26]:

	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 [27]:
```
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')
```



How long do people stay at the hotels?

In [28]:
```
filter = df['is_canceled'] == 0
data = df[filter]
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_se
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Co
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	On

In [29]:
```
data['total_nights'] = data['stays_in_weekend_nights'] + data['stays_in_week_nights']
data.head()
```

Out[29]:

	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_se
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	0.0	0	BB	PRT	
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	0.0	0	BB	GBR	
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	0.0	0	BB	GBR	Co
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	0.0	0	BB	GBR	On

5 rows × 33 columns

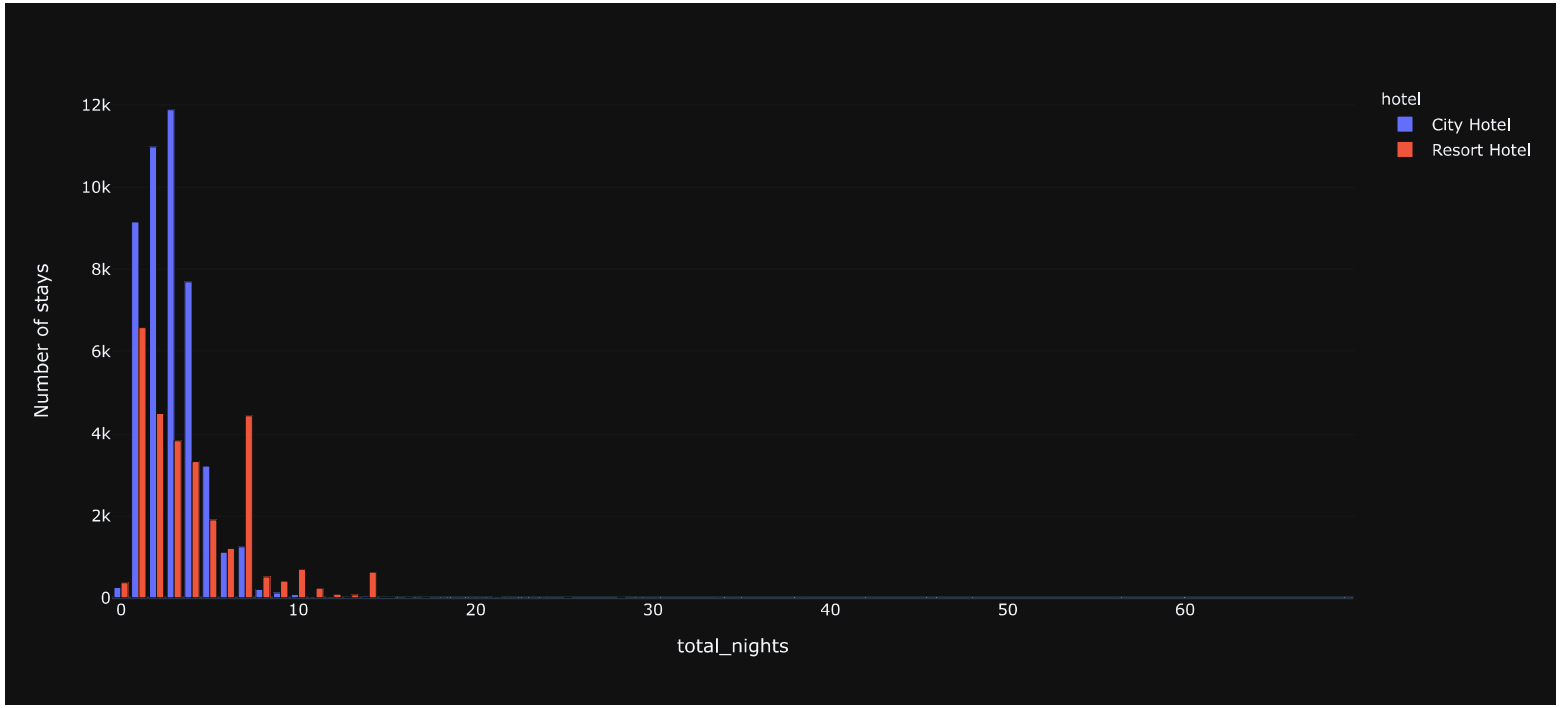
In [30]:
```
stay = data.groupby(['total_nights', 'hotel']).agg('count').reset_index()
stay = stay.iloc[:, :3]
stay = stay.rename(columns={'is_canceled':'Number of stays'})
stay
```

Out[30]:

	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 [31]:
```
px.bar(data_frame = stay, x = 'total_nights', y = 'Number of stays', color = 'hotel', barmode = 'group',
       template = 'plotly_dark')
```

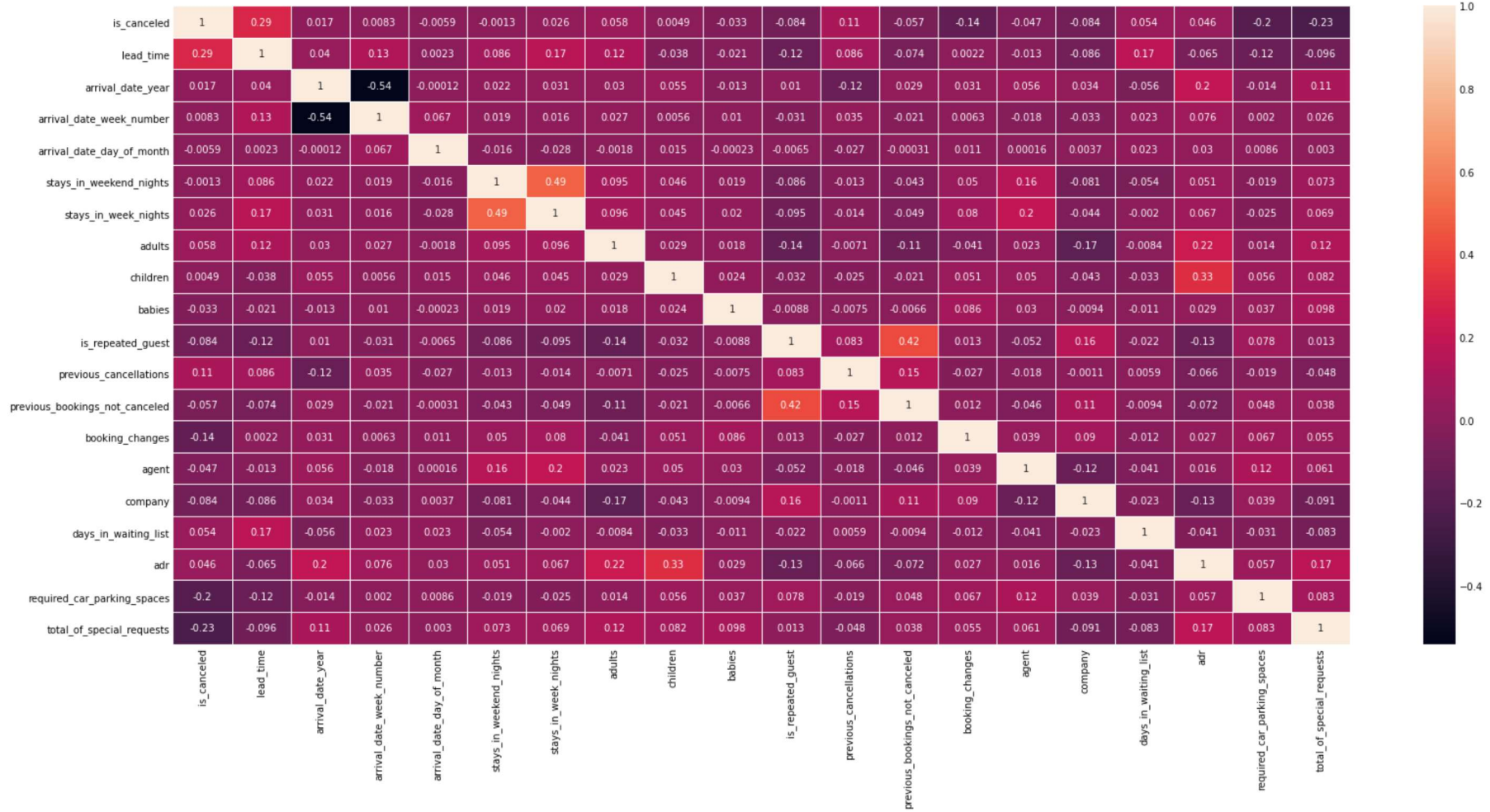


Data Pre Processing

In [32]:
```
plt.figure(figsize = (24, 12))

corr = df.corr()
```

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



In [33]:
```
correlation = df.corr()['is_canceled'].abs().sort_values(ascending = False)
correlation
```

Out[33]:
```
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 [34]:
```
# 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 [35]:
```
df.head()
```

Out[35]:

	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
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 [36]:
```
# creating numerical and categorical dataframes

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

Out[36]:
```
['hotel',
 'arrival_date_month',
 'meal',
 'market_segment',
 'distribution_channel',
 'reserved_room_type',
 'deposit_type',
 'customer_type',
 'reservation_status_date']
```

In [37]:
```
cat_df = df[cat_cols]
cat_df.head()
```

Out[37]:

	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	2015-07-01
1	Resort Hotel	July	BB	Direct	Direct	C	No Deposit	Transient	2015-07-01
2	Resort Hotel	July	BB	Direct	Direct	A	No Deposit	Transient	2015-07-02
3	Resort Hotel	July	BB	Corporate	Corporate	A	No Deposit	Transient	2015-07-02
4	Resort Hotel	July	BB	Online TA	TA/TO	A	No Deposit	Transient	2015-07-03

In [38]:
```
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 [39]:
```
cat_df.drop(['reservation_status_date','arrival_date_month'] , axis = 1, inplace = True)
```

In [40]:
```
cat_df.head()
```

Out[40]:

	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 [41]:
```
# 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 [42]:
```
# 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 [43]:
```
cat_df.head()
```

Out[43]:

	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day
0	0	0	0	0	0	0	0	0	7	1
1	0	0	0	0	0	0	0	0	7	1
2	0	0	0	0	1	0	0	0	7	2
3	0	0	1	1	1	0	0	0	7	2
4	0	0	2	2	1	0	0	0	7	3

In [44]:
```
num_df = df.drop(columns = cat_cols, axis = 1)
num_df.drop('is_canceled', axis = 1, inplace = True)
num_df
```

Out[44]:

	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	age
0	342	27	1	0	0	2	0.0	0	0	0	0	
1	737	27	1	0	0	2	0.0	0	0	0	0	
2	7	27	1	0	1	1	0.0	0	0	0	0	
3	13	27	1	0	1	1	0.0	0	0	0	0	30
4	14	27	1	0	2	2	0.0	0	0	0	0	24
...	...	...	...	...	...	...	...	...	...	...	...	
119385	23	35	30	2	5	2	0.0	0	0	0	0	39
119386	102	35	31	2	5	3	0.0	0	0	0	0	
119387	34	35	31	2	5	2	0.0	0	0	0	0	
119388	109	35	31	2	5	2	0.0	0	0	0	0	8
119389	205	35	29	2	7	2	0.0	0	0	0	0	

119210 rows × 16 columns

In [45]:
```
num_df.var()
```

Out[45]:
```
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 [46]:
```
# 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 [47]:
```
num_df.var()
```

Out[47]:
```
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 [48]:
```
num_df['adr'] = num_df['adr'].fillna(value = num_df['adr'].mean())
```

In [49]:
```
num_df.head()
```

Out[49]:

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

In [50]:
```
X = pd.concat([cat_df, num_df], axis = 1)
y = df['is_canceled']
```

In [51]:
```
X.shape, y.shape
```

Out[51]:
```
((119210, 26), (119210,))
```

In [52]:
```
# 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 [53]:
```
X_train.head()
```

Out[53]:

	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	st
51581	1	0	2	2	2	0	0	2	3	24	4.234107	3.091042	3.091042	2	
88927	1	0	3	2	1	0	0	2	5	15	0.000000	3.044522	2.639057	0	
78388	1	0	1	1	1	0	0	0	10	15	1.386294	3.761200	2.564949	1	
117245	1	0	2	2	1	0	0	3	8	1	5.501258	3.433987	3.401197	2	
6930	0	0	2	2	1	0	0	2	3	26	4.905275	3.401197	2.564949	0	

In [54]:
```
X_test.head()
```

Out[54]:

	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	st
92508	1	3	2	2	1	0	0	2	7	8	3.135494	3.367296	1.945910	0	
62810	1	0	5	2	1	3	0	2	11	25	4.094345	1.609438	3.178054	1	
10799	0	0	1	1	1	0	0	3	4	3	1.945910	2.708050	1.791759	0	
111056	1	3	2	2	1	0	0	3	5	6	3.465736	2.944439	1.386294	0	
8939	0	2	2	2	6	0	0	2	6	20	4.882802	3.784190	3.091042	0	

In [55]:
```
y_train.head(), y_test.head()
```

Out[55]:
```
(51581     1
 88927     0
 78388     0
 117245    0
 6930      1
 Name: is_canceled, dtype: int64,
 92508     0
 62810     1
 10799     1
 111056    0
 8939      1
 Name: is_canceled, dtype: int64)
```

Model Building

Logistic Regression

In [56]:
```
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 : 0.8141375164275928
Confusion Matrix :
[[21328  1300]
 [ 5347  7788]]
Classification Report :
              precision    recall  f1-score   support

           0       0.80      0.94      0.87     22628
           1       0.86      0.59      0.70     13135

    accuracy                           0.81     35763
   macro avg       0.83      0.77      0.78     35763
weighted avg       0.82      0.81      0.80     35763
```

KNN

In [57]:
```
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 : 0.8907530128904174
Confusion Matrix :
[[21782   846]
 [ 3061 10074]]
Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.96      0.92     22628
           1       0.92      0.77      0.84     13135

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

Decision Tree Classifier

In [58]:
```
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 : 0.943489080893661
Confusion Matrix :
[[21636   992]
 [ 1029 12106]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.96      0.96     22628
           1       0.92      0.92      0.92     13135

    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 [59]:
```
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 : 0.9546179011827867
Confusion Matrix :
[[22447   181]
 [ 1442 11693]]
Classification Report :
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     22628
           1       0.98      0.89      0.94     13135

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

Ada Boost Classifier

In [60]:
```
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 : 0.9432374241534547
Confusion Matrix :
[[21646   982]
 [ 1048 12087]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      0.96      0.96     22628
           1       0.92      0.92      0.92     13135

    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 [61]:
```
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 Ada Boost Classifier is : {acc_gb}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9136537762491961
Confusion Matrix :
[[22411   217]
 [ 2871 10264]]
Classification Report :
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     22628
           1       0.98      0.78      0.87     13135

    accuracy                           0.91     35763
   macro avg       0.93      0.89      0.90     35763
weighted avg       0.92      0.91      0.91     35763
```

XgBoost Classifier

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

```
[04:24:46] WARNING: ../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Expli
citly set eval_metric if you'd like to restore the old behavior.
Accuracy Score of Ada Boost Classifier is : 0.9835863881665409
Confusion Matrix :
[[22618    10]
 [  577 12558]]
Classification Report :
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     22628
           1       1.00      0.96      0.98     13135

    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 [63]:
```
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: 0.4648867       total: 97.1ms   remaining: 9.61s
1:      learn: 0.4058078       total: 133ms    remaining: 6.52s
2:      learn: 0.3875110       total: 169ms    remaining: 5.48s
3:      learn: 0.3535874       total: 206ms    remaining: 4.93s
4:      learn: 0.3236745       total: 240ms    remaining: 4.57s
5:      learn: 0.2684702       total: 275ms    remaining: 4.3s
6:      learn: 0.2457782       total: 309ms    remaining: 4.1s
7:      learn: 0.2191933       total: 345ms    remaining: 3.96s
8:      learn: 0.2085785       total: 381ms    remaining: 3.85s
9:      learn: 0.1869480       total: 418ms    remaining: 3.76s
10:     learn: 0.1789103       total: 452ms    remaining: 3.66s
11:     learn: 0.1657092       total: 485ms    remaining: 3.56s
12:     learn: 0.1533197       total: 522ms    remaining: 3.5s
13:     learn: 0.1456774       total: 557ms    remaining: 3.42s
14:     learn: 0.1369556       total: 590ms    remaining: 3.34s
15:     learn: 0.1333089       total: 624ms    remaining: 3.27s
16:     learn: 0.1258505       total: 658ms    remaining: 3.21s
17:     learn: 0.1183625       total: 693ms    remaining: 3.16s
18:     learn: 0.1140672       total: 728ms    remaining: 3.1s
19:     learn: 0.1123283       total: 760ms    remaining: 3.04s
20:     learn: 0.1077371       total: 792ms    remaining: 2.98s
21:     learn: 0.1023068       total: 828ms    remaining: 2.94s
22:     learn: 0.1010914       total: 861ms    remaining: 2.88s
23:     learn: 0.0976685       total: 895ms    remaining: 2.83s
24:     learn: 0.0934428       total: 930ms    remaining: 2.79s
25:     learn: 0.0909552       total: 964ms    remaining: 2.74s
26:     learn: 0.0885469       total: 997ms    remaining: 2.69s
27:     learn: 0.0831459       total: 1.03s    remaining: 2.66s
28:     learn: 0.0803908       total: 1.07s    remaining: 2.62s
29:     learn: 0.0777907       total: 1.1s     remaining: 2.57s
30:     learn: 0.0749345       total: 1.14s    remaining: 2.53s
31:     learn: 0.0721676       total: 1.17s    remaining: 2.49s
32:     learn: 0.0700127       total: 1.21s    remaining: 2.45s
33:     learn: 0.0690773       total: 1.24s    remaining: 2.41s
34:     learn: 0.0671976       total: 1.3s     remaining: 2.42s
35:     learn: 0.0645444       total: 1.38s    remaining: 2.45s
36:     learn: 0.0630320       total: 1.47s    remaining: 2.51s
37:     learn: 0.0620520       total: 1.56s    remaining: 2.55s
38:     learn: 0.0612635       total: 1.65s    remaining: 2.58s
39:     learn: 0.0607282       total: 1.74s    remaining: 2.6s
40:     learn: 0.0589359       total: 1.82s    remaining: 2.62s
41:     learn: 0.0581673       total: 1.87s    remaining: 2.59s
42:     learn: 0.0548969       total: 1.96s    remaining: 2.6s
43:     learn: 0.0538749       total: 2.01s    remaining: 2.56s
44:     learn: 0.0513108       total: 2.07s    remaining: 2.53s
45:     learn: 0.0501760       total: 2.15s    remaining: 2.53s
46:     learn: 0.0485658       total: 2.24s    remaining: 2.52s
47:     learn: 0.0472781       total: 2.33s    remaining: 2.52s
48:     learn: 0.0451023       total: 2.38s    remaining: 2.48s
49:     learn: 0.0436204       total: 2.48s    remaining: 2.48s
50:     learn: 0.0433404       total: 2.56s    remaining: 2.46s
51:     learn: 0.0417831       total: 2.67s    remaining: 2.46s
52:     learn: 0.0412041       total: 2.76s    remaining: 2.45s
53:     learn: 0.0399382       total: 2.85s    remaining: 2.43s
54:     learn: 0.0391923       total: 2.94s    remaining: 2.41s
55:     learn: 0.0384424       total: 3.04s    remaining: 2.39s
56:     learn: 0.0371314       total: 3.13s    remaining: 2.36s
57:     learn: 0.0366425       total: 3.16s    remaining: 2.29s
58:     learn: 0.0356611       total: 3.19s    remaining: 2.22s
59:     learn: 0.0343144       total: 3.23s    remaining: 2.15s
60:     learn: 0.0340940       total: 3.26s    remaining: 2.08s
61:     learn: 0.0333094       total: 3.29s    remaining: 2.02s
62:     learn: 0.0320758       total: 3.33s    remaining: 1.95s
63:     learn: 0.0314483       total: 3.36s    remaining: 1.89s
64:     learn: 0.0309426       total: 3.39s    remaining: 1.83s
65:     learn: 0.0306604       total: 3.43s    remaining: 1.76s
66:     learn: 0.0296151       total: 3.46s    remaining: 1.71s
```

```
67:     learn: 0.0293775        total: 3.49s    remaining: 1.64s
68:     learn: 0.0291442        total: 3.52s    remaining: 1.58s
69:     learn: 0.0287316        total: 3.56s    remaining: 1.52s
70:     learn: 0.0282689        total: 3.59s    remaining: 1.47s
71:     learn: 0.0282646        total: 3.63s    remaining: 1.41s
72:     learn: 0.0268145        total: 3.66s    remaining: 1.35s
73:     learn: 0.0267197        total: 3.7s     remaining: 1.3s
74:     learn: 0.0260803        total: 3.73s    remaining: 1.24s
75:     learn: 0.0251093        total: 3.77s    remaining: 1.19s
76:     learn: 0.0248968        total: 3.8s     remaining: 1.14s
77:     learn: 0.0241763        total: 3.84s    remaining: 1.08s
78:     learn: 0.0231455        total: 3.87s    remaining: 1.03s
79:     learn: 0.0228714        total: 3.91s    remaining: 977ms
80:     learn: 0.0218651        total: 3.94s    remaining: 925ms
81:     learn: 0.0212382        total: 3.98s    remaining: 873ms
82:     learn: 0.0212352        total: 4.01s    remaining: 822ms
83:     learn: 0.0210997        total: 4.04s    remaining: 770ms
84:     learn: 0.0206427        total: 4.08s    remaining: 720ms
85:     learn: 0.0199398        total: 4.12s    remaining: 670ms
86:     learn: 0.0192763        total: 4.15s    remaining: 620ms
87:     learn: 0.0192078        total: 4.18s    remaining: 570ms
88:     learn: 0.0187708        total: 4.22s    remaining: 521ms
89:     learn: 0.0185294        total: 4.25s    remaining: 472ms
90:     learn: 0.0183999        total: 4.29s    remaining: 424ms
91:     learn: 0.0178609        total: 4.32s    remaining: 376ms
92:     learn: 0.0176215        total: 4.36s    remaining: 328ms
93:     learn: 0.0170791        total: 4.39s    remaining: 280ms
94:     learn: 0.0169462        total: 4.42s    remaining: 233ms
95:     learn: 0.0161446        total: 4.46s    remaining: 186ms
96:     learn: 0.0158362        total: 4.49s    remaining: 139ms
97:     learn: 0.0158016        total: 4.52s    remaining: 92.2ms
98:     learn: 0.0155392        total: 4.55s    remaining: 46ms
99:     learn: 0.0151124        total: 4.58s    remaining: 0us
```

In [64]:
```
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 : 0.9954701786762855
Confusion Matrix :
[[22612    16]
 [  146 12989]]
Classification Report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     22628
           1       1.00      0.99      0.99     13135

    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 [65]:
```
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 Ada Boost Classifier is : {acc_etc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9539468165422363
Confusion Matrix :
[[22423   205]
 [ 1442 11693]]
Classification Report :
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     22628
           1       0.98      0.89      0.93     13135

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

LGBM Classifier

In [66]:
```
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 Ada Boost Classifier is : {acc_lgbm}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{clf_report}")
```

```
Accuracy Score of Ada Boost Classifier is : 0.9474037412968711
Confusion Matrix :
[[21636   992]
 [  889 12246]]
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     22628
           1       0.93      0.93      0.93     13135

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

Voting Classifier

In [67]:
```
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: 0.4648867       total: 35.8ms   remaining: 3.54s
1:      learn: 0.4058078       total: 69.2ms   remaining: 3.39s
2:      learn: 0.3875110       total: 102ms    remaining: 3.31s
3:      learn: 0.3535874       total: 136ms    remaining: 3.26s
4:      learn: 0.3236745       total: 171ms    remaining: 3.25s
5:      learn: 0.2684702       total: 207ms    remaining: 3.24s
6:      learn: 0.2457782       total: 241ms    remaining: 3.2s
7:      learn: 0.2191933       total: 275ms    remaining: 3.17s
8:      learn: 0.2085785       total: 311ms    remaining: 3.14s
9:      learn: 0.1869480       total: 348ms    remaining: 3.13s
10:     learn: 0.1789103       total: 382ms    remaining: 3.09s
11:     learn: 0.1657092       total: 419ms    remaining: 3.07s
12:     learn: 0.1533197       total: 457ms    remaining: 3.06s
13:     learn: 0.1456774       total: 494ms    remaining: 3.03s
14:     learn: 0.1369556       total: 526ms    remaining: 2.98s
15:     learn: 0.1333089       total: 561ms    remaining: 2.94s
16:     learn: 0.1258505       total: 597ms    remaining: 2.92s
17:     learn: 0.1183625       total: 635ms    remaining: 2.89s
18:     learn: 0.1140672       total: 670ms    remaining: 2.86s
19:     learn: 0.1123283       total: 705ms    remaining: 2.82s
20:     learn: 0.1077371       total: 737ms    remaining: 2.77s
21:     learn: 0.1023068       total: 770ms    remaining: 2.73s
22:     learn: 0.1010914       total: 802ms    remaining: 2.68s
23:     learn: 0.0976685       total: 882ms    remaining: 2.79s
24:     learn: 0.0934428       total: 918ms    remaining: 2.75s
25:     learn: 0.0909552       total: 952ms    remaining: 2.71s
26:     learn: 0.0885469       total: 985ms    remaining: 2.66s
27:     learn: 0.0831459       total: 1.02s    remaining: 2.62s
28:     learn: 0.0803908       total: 1.05s    remaining: 2.58s
29:     learn: 0.0777907       total: 1.09s    remaining: 2.56s
30:     learn: 0.0749345       total: 1.17s    remaining: 2.61s
31:     learn: 0.0721676       total: 1.22s    remaining: 2.59s
32:     learn: 0.0700127       total: 1.25s    remaining: 2.54s
33:     learn: 0.0690773       total: 1.28s    remaining: 2.49s
34:     learn: 0.0671976       total: 1.32s    remaining: 2.45s
35:     learn: 0.0645444       total: 1.35s    remaining: 2.41s
36:     learn: 0.0630320       total: 1.39s    remaining: 2.36s
37:     learn: 0.0620520       total: 1.42s    remaining: 2.32s
38:     learn: 0.0612635       total: 1.46s    remaining: 2.28s
39:     learn: 0.0607282       total: 1.49s    remaining: 2.23s
40:     learn: 0.0589359       total: 1.53s    remaining: 2.2s
41:     learn: 0.0581673       total: 1.56s    remaining: 2.16s
42:     learn: 0.0548969       total: 1.6s     remaining: 2.12s
43:     learn: 0.0538749       total: 1.63s    remaining: 2.08s
44:     learn: 0.0513108       total: 1.67s    remaining: 2.04s
45:     learn: 0.0501760       total: 1.7s     remaining: 2s
46:     learn: 0.0485658       total: 1.73s    remaining: 1.95s
47:     learn: 0.0472781       total: 1.77s    remaining: 1.91s
48:     learn: 0.0451023       total: 1.8s     remaining: 1.87s
49:     learn: 0.0436204       total: 1.83s    remaining: 1.83s
50:     learn: 0.0433404       total: 1.87s    remaining: 1.79s
51:     learn: 0.0417831       total: 1.9s     remaining: 1.76s
52:     learn: 0.0412041       total: 1.94s    remaining: 1.72s
53:     learn: 0.0399382       total: 1.97s    remaining: 1.68s
54:     learn: 0.0391923       total: 2.01s    remaining: 1.64s
55:     learn: 0.0384424       total: 2.04s    remaining: 1.6s
56:     learn: 0.0371314       total: 2.07s    remaining: 1.56s
57:     learn: 0.0366425       total: 2.1s     remaining: 1.52s
58:     learn: 0.0356611       total: 2.14s    remaining: 1.49s
59:     learn: 0.0343144       total: 2.17s    remaining: 1.45s
60:     learn: 0.0340940       total: 2.2s     remaining: 1.41s
61:     learn: 0.0333094       total: 2.23s    remaining: 1.37s
62:     learn: 0.0320758       total: 2.27s    remaining: 1.33s
63:     learn: 0.0314483       total: 2.3s     remaining: 1.29s
64:     learn: 0.0309426       total: 2.33s    remaining: 1.25s
65:     learn: 0.0306604       total: 2.37s    remaining: 1.22s
66:     learn: 0.0296151       total: 2.4s     remaining: 1.18s
```

```
67:     learn: 0.0293775        total: 2.43s    remaining: 1.14s
68:     learn: 0.0291442        total: 2.46s    remaining: 1.11s
69:     learn: 0.0287316        total: 2.5s     remaining: 1.07s
70:     learn: 0.0282689        total: 2.53s    remaining: 1.03s
71:     learn: 0.0282646        total: 2.56s    remaining: 998ms
72:     learn: 0.0268145        total: 2.6s     remaining: 963ms
73:     learn: 0.0267197        total: 2.63s    remaining: 926ms
74:     learn: 0.0260803        total: 2.67s    remaining: 889ms
75:     learn: 0.0251093        total: 2.7s     remaining: 853ms
76:     learn: 0.0248968        total: 2.73s    remaining: 816ms
77:     learn: 0.0241763        total: 2.77s    remaining: 781ms
78:     learn: 0.0231455        total: 2.8s     remaining: 744ms
79:     learn: 0.0228714        total: 2.84s    remaining: 709ms
80:     learn: 0.0218651        total: 2.87s    remaining: 673ms
81:     learn: 0.0212382        total: 2.9s     remaining: 638ms
82:     learn: 0.0212352        total: 2.93s    remaining: 601ms
83:     learn: 0.0210997        total: 2.97s    remaining: 565ms
84:     learn: 0.0206427        total: 3s       remaining: 529ms
85:     learn: 0.0199398        total: 3.03s    remaining: 494ms
86:     learn: 0.0192763        total: 3.07s    remaining: 458ms
87:     learn: 0.0192078        total: 3.1s     remaining: 423ms
88:     learn: 0.0187708        total: 3.13s    remaining: 387ms
89:     learn: 0.0185294        total: 3.17s    remaining: 352ms
90:     learn: 0.0183999        total: 3.2s     remaining: 316ms
91:     learn: 0.0178609        total: 3.23s    remaining: 281ms
92:     learn: 0.0176215        total: 3.27s    remaining: 246ms
93:     learn: 0.0170791        total: 3.3s     remaining: 211ms
94:     learn: 0.0169462        total: 3.33s    remaining: 176ms
95:     learn: 0.0161446        total: 3.37s    remaining: 140ms
96:     learn: 0.0158362        total: 3.4s     remaining: 105ms
97:     learn: 0.0158016        total: 3.43s    remaining: 69.9ms
98:     learn: 0.0155392        total: 3.46s    remaining: 34.9ms
99:     learn: 0.0151124        total: 3.49s    remaining: 0us
[04:25:32] WARNING: ../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Expli
citly set eval_metric if you'd like to restore the old behavior.
```

Out[67]:

```
VotingClassifier(estimators=[('Gradient Boosting Classifier',
                              GradientBoostingClassifier()),
                             ('Cat Boost Classifier',
                              <catboost.core.CatBoostClassifier object at 0x79620ce1df90>),
                             ('XGboost',
                              XGBClassifier(base_score=0.5, booster='gbtree',
                                            colsample_bylevel=1,
                                            colsample_bynode=1,
                                            colsample_bytree=1, gamma=0,
                                            gpu_id=-1, importance_type='gain',
                                            interaction_constraints='...
                                            tree_method='exact',
                                            validate_parameters=1,
                                            verbosity=None)),
                             ('Decision Tree', DecisionTreeClassifier()),
                             ('Extra Tree', ExtraTreesClassifier()),
                             ('Light Gradient',
                              LGBMClassifier(learning_rate=1)),
                             ('Random Forest', RandomForestClassifier()),
                             ('Ada Boost',
                              AdaBoostClassifier(base_estimator=DecisionTreeClassifier())),
                             ('Logistic', LogisticRegression()),
                             ('Knn', KNeighborsClassifier())])
```

In [68]:

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

```
Accuracy Score of Ada Boost Classifier is : 0.9641808573106283
Confusion Matrix :
[[22613    15]
 [ 1266 11869]]
Classification Report :
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     22628
           1       1.00      0.90      0.95     13135

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

ANN

In [69]:
```
from tensorflow.keras.utils import to_categorical

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

In [70]:
```
# 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 [71]:
```
import keras
from keras.layers import Dense
from keras.models import Sequential

model  = Sequential()
model.add(Dense(100, activation = 'relu', input_shape = (26, )))
model.add(Dense(100, activation = 'relu'))
model.add(Dense(2, activation = 'sigmoid'))
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model_history = model.fit(X_train, y_train, validation_data = (X_test, y_test),
                          epochs = 100)
```

```
Epoch 1/100
2608/2608 [==============================] - 9s 3ms/step - loss: 0.4204 - accuracy: 0.8115 - val_loss: 0.2111 - val_accuracy: 0.9232
Epoch 2/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.1896 - accuracy: 0.9346 - val_loss: 0.1453 - val_accuracy: 0.9460
Epoch 3/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.1306 - accuracy: 0.9579 - val_loss: 0.1159 - val_accuracy: 0.9599
Epoch 4/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.1006 - accuracy: 0.9685 - val_loss: 0.1209 - val_accuracy: 0.9600
Epoch 5/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0912 - accuracy: 0.9716 - val_loss: 0.0778 - val_accuracy: 0.9750
Epoch 6/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0780 - accuracy: 0.9759 - val_loss: 0.0616 - val_accuracy: 0.9819
Epoch 7/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0712 - accuracy: 0.9777 - val_loss: 0.0722 - val_accuracy: 0.9806
Epoch 8/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0649 - accuracy: 0.9800 - val_loss: 0.0588 - val_accuracy: 0.9831
Epoch 9/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0624 - accuracy: 0.9805 - val_loss: 0.0681 - val_accuracy: 0.9787
Epoch 10/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0603 - accuracy: 0.9810 - val_loss: 0.0483 - val_accuracy: 0.9874
Epoch 11/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0531 - accuracy: 0.9831 - val_loss: 0.0674 - val_accuracy: 0.9793
Epoch 12/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0510 - accuracy: 0.9841 - val_loss: 0.0685 - val_accuracy: 0.9795
Epoch 13/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0509 - accuracy: 0.9836 - val_loss: 0.0356 - val_accuracy: 0.9887
Epoch 14/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0466 - accuracy: 0.9857 - val_loss: 0.0518 - val_accuracy: 0.9841
Epoch 15/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0476 - accuracy: 0.9844 - val_loss: 0.0391 - val_accuracy: 0.9882
Epoch 16/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0426 - accuracy: 0.9867 - val_loss: 0.0608 - val_accuracy: 0.9811
Epoch 17/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0488 - accuracy: 0.9854 - val_loss: 0.0564 - val_accuracy: 0.9838
Epoch 18/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0398 - accuracy: 0.9875 - val_loss: 0.0680 - val_accuracy: 0.9794
Epoch 19/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0452 - accuracy: 0.9862 - val_loss: 0.0450 - val_accuracy: 0.9892
Epoch 20/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0447 - accuracy: 0.9858 - val_loss: 0.0341 - val_accuracy: 0.9887
Epoch 21/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0372 - accuracy: 0.9890 - val_loss: 0.0893 - val_accuracy: 0.9781
Epoch 22/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0377 - accuracy: 0.9883 - val_loss: 0.0420 - val_accuracy: 0.9866
Epoch 23/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0403 - accuracy: 0.9873 - val_loss: 0.0938 - val_accuracy: 0.9655
Epoch 24/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0387 - accuracy: 0.9869 - val_loss: 0.0388 - val_accuracy: 0.9899
Epoch 25/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0342 - accuracy: 0.9892 - val_loss: 0.0343 - val_accuracy: 0.9905
Epoch 26/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0336 - accuracy: 0.9893 - val_loss: 0.0557 - val_accuracy: 0.9813
Epoch 27/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0343 - accuracy: 0.9898 - val_loss: 0.0510 - val_accuracy: 0.9861
Epoch 28/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0362 - accuracy: 0.9880 - val_loss: 0.0375 - val_accuracy: 0.9878
Epoch 29/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0326 - accuracy: 0.9897 - val_loss: 0.0549 - val_accuracy: 0.9850
Epoch 30/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0374 - accuracy: 0.9881 - val_loss: 0.0292 - val_accuracy: 0.9913
Epoch 31/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0313 - accuracy: 0.9899 - val_loss: 0.0341 - val_accuracy: 0.9904
Epoch 32/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0343 - accuracy: 0.9893 - val_loss: 0.0316 - val_accuracy: 0.9900
Epoch 33/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0311 - accuracy: 0.9899 - val_loss: 0.0335 - val_accuracy: 0.9897
Epoch 34/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0326 - accuracy: 0.9898 - val_loss: 0.0539 - val_accuracy: 0.9858
```

```
Epoch 35/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0327 - accuracy: 0.9893 - val_loss: 0.0334 - val_accuracy: 0.9909
Epoch 36/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0303 - accuracy: 0.9905 - val_loss: 0.0430 - val_accuracy: 0.9878
Epoch 37/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0301 - accuracy: 0.9909 - val_loss: 0.0291 - val_accuracy: 0.9917
Epoch 38/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0307 - accuracy: 0.9902 - val_loss: 0.0342 - val_accuracy: 0.9917
Epoch 39/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0287 - accuracy: 0.9905 - val_loss: 0.0408 - val_accuracy: 0.9886
Epoch 40/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0302 - accuracy: 0.9901 - val_loss: 0.0383 - val_accuracy: 0.9902
Epoch 41/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0264 - accuracy: 0.9915 - val_loss: 0.0429 - val_accuracy: 0.9871
Epoch 42/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0286 - accuracy: 0.9909 - val_loss: 0.0444 - val_accuracy: 0.9875
Epoch 43/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0318 - accuracy: 0.9905 - val_loss: 0.0266 - val_accuracy: 0.9920
Epoch 44/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0254 - accuracy: 0.9915 - val_loss: 0.0437 - val_accuracy: 0.9881
Epoch 45/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0241 - accuracy: 0.9923 - val_loss: 0.0402 - val_accuracy: 0.9897
Epoch 46/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0263 - accuracy: 0.9920 - val_loss: 0.0441 - val_accuracy: 0.9865
Epoch 47/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0286 - accuracy: 0.9908 - val_loss: 0.0325 - val_accuracy: 0.9917
Epoch 48/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0242 - accuracy: 0.9922 - val_loss: 0.1266 - val_accuracy: 0.9636
Epoch 49/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0291 - accuracy: 0.9912 - val_loss: 0.0343 - val_accuracy: 0.9907
Epoch 50/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0263 - accuracy: 0.9920 - val_loss: 0.0354 - val_accuracy: 0.9898
Epoch 51/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0252 - accuracy: 0.9920 - val_loss: 0.0287 - val_accuracy: 0.9912
Epoch 52/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0251 - accuracy: 0.9919 - val_loss: 0.0414 - val_accuracy: 0.9892
Epoch 53/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0231 - accuracy: 0.9930 - val_loss: 0.0336 - val_accuracy: 0.9902
Epoch 54/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0245 - accuracy: 0.9923 - val_loss: 0.0282 - val_accuracy: 0.9914
Epoch 55/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0235 - accuracy: 0.9928 - val_loss: 0.0353 - val_accuracy: 0.9899
Epoch 56/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0250 - accuracy: 0.9920 - val_loss: 0.0397 - val_accuracy: 0.9897
Epoch 57/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0245 - accuracy: 0.9920 - val_loss: 0.0298 - val_accuracy: 0.9923
Epoch 58/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0210 - accuracy: 0.9931 - val_loss: 0.0286 - val_accuracy: 0.9917
Epoch 59/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0214 - accuracy: 0.9931 - val_loss: 0.0353 - val_accuracy: 0.9909
Epoch 60/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0215 - accuracy: 0.9929 - val_loss: 0.0325 - val_accuracy: 0.9913
Epoch 61/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0233 - accuracy: 0.9928 - val_loss: 0.0330 - val_accuracy: 0.9907
Epoch 62/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0235 - accuracy: 0.9926 - val_loss: 0.0382 - val_accuracy: 0.9895
Epoch 63/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0212 - accuracy: 0.9929 - val_loss: 0.0382 - val_accuracy: 0.9901
Epoch 64/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0217 - accuracy: 0.9930 - val_loss: 0.0450 - val_accuracy: 0.9854
Epoch 65/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0250 - accuracy: 0.9925 - val_loss: 0.0457 - val_accuracy: 0.9883
Epoch 66/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0223 - accuracy: 0.9927 - val_loss: 0.0336 - val_accuracy: 0.9912
Epoch 67/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0211 - accuracy: 0.9933 - val_loss: 0.0392 - val_accuracy: 0.9901
Epoch 68/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0236 - accuracy: 0.9920 - val_loss: 0.0444 - val_accuracy: 0.9890
```

```
Epoch 69/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0224 - accuracy: 0.9930 - val_loss: 0.0267 - val_accuracy: 0.9926
Epoch 70/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0215 - accuracy: 0.9930 - val_loss: 0.0328 - val_accuracy: 0.9911
Epoch 71/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0227 - accuracy: 0.9927 - val_loss: 0.0597 - val_accuracy: 0.9857
Epoch 72/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0207 - accuracy: 0.9930 - val_loss: 0.0474 - val_accuracy: 0.9863
Epoch 73/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0219 - accuracy: 0.9929 - val_loss: 0.0257 - val_accuracy: 0.9937
Epoch 74/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0180 - accuracy: 0.9944 - val_loss: 0.0334 - val_accuracy: 0.9917
Epoch 75/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0217 - accuracy: 0.9933 - val_loss: 0.0278 - val_accuracy: 0.9921
Epoch 76/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0196 - accuracy: 0.9936 - val_loss: 0.0413 - val_accuracy: 0.9900
Epoch 77/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0206 - accuracy: 0.9936 - val_loss: 0.0437 - val_accuracy: 0.9874
Epoch 78/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0209 - accuracy: 0.9932 - val_loss: 0.0360 - val_accuracy: 0.9902
Epoch 79/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0190 - accuracy: 0.9939 - val_loss: 0.0427 - val_accuracy: 0.9897
Epoch 80/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0198 - accuracy: 0.9935 - val_loss: 0.0300 - val_accuracy: 0.9922
Epoch 81/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0184 - accuracy: 0.9939 - val_loss: 0.0283 - val_accuracy: 0.9930
Epoch 82/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0173 - accuracy: 0.9941 - val_loss: 0.0398 - val_accuracy: 0.9902
Epoch 83/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0196 - accuracy: 0.9939 - val_loss: 0.0335 - val_accuracy: 0.9920
Epoch 84/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0188 - accuracy: 0.9938 - val_loss: 0.0273 - val_accuracy: 0.9924
Epoch 85/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0177 - accuracy: 0.9942 - val_loss: 0.0424 - val_accuracy: 0.9893
Epoch 86/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0217 - accuracy: 0.9930 - val_loss: 0.0291 - val_accuracy: 0.9931
Epoch 87/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0201 - accuracy: 0.9938 - val_loss: 0.0427 - val_accuracy: 0.9897
Epoch 88/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0167 - accuracy: 0.9949 - val_loss: 0.0318 - val_accuracy: 0.9918
Epoch 89/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0204 - accuracy: 0.9932 - val_loss: 0.0254 - val_accuracy: 0.9935
Epoch 90/100
2608/2608 [==============================] - 7s 3ms/step - loss: 0.0182 - accuracy: 0.9940 - val_loss: 0.0294 - val_accuracy: 0.9923
Epoch 91/100
2608/2608 [==============================] - 9s 3ms/step - loss: 0.0179 - accuracy: 0.9940 - val_loss: 0.0363 - val_accuracy: 0.9903
Epoch 92/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0204 - accuracy: 0.9931 - val_loss: 0.0445 - val_accuracy: 0.9888
Epoch 93/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0169 - accuracy: 0.9945 - val_loss: 0.0332 - val_accuracy: 0.9908
Epoch 94/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0155 - accuracy: 0.9950 - val_loss: 0.0539 - val_accuracy: 0.9882
Epoch 95/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0198 - accuracy: 0.9940 - val_loss: 0.0292 - val_accuracy: 0.9930
Epoch 96/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0138 - accuracy: 0.9953 - val_loss: 0.0281 - val_accuracy: 0.9929
Epoch 97/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0189 - accuracy: 0.9941 - val_loss: 0.0315 - val_accuracy: 0.9920
Epoch 98/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0152 - accuracy: 0.9953 - val_loss: 0.0329 - val_accuracy: 0.9918
Epoch 99/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0176 - accuracy: 0.9939 - val_loss: 0.0363 - val_accuracy: 0.9897
Epoch 100/100
2608/2608 [==============================] - 8s 3ms/step - loss: 0.0161 - accuracy: 0.9949 - val_loss: 0.0300 - val_accuracy: 0.9924
```

In [72]:

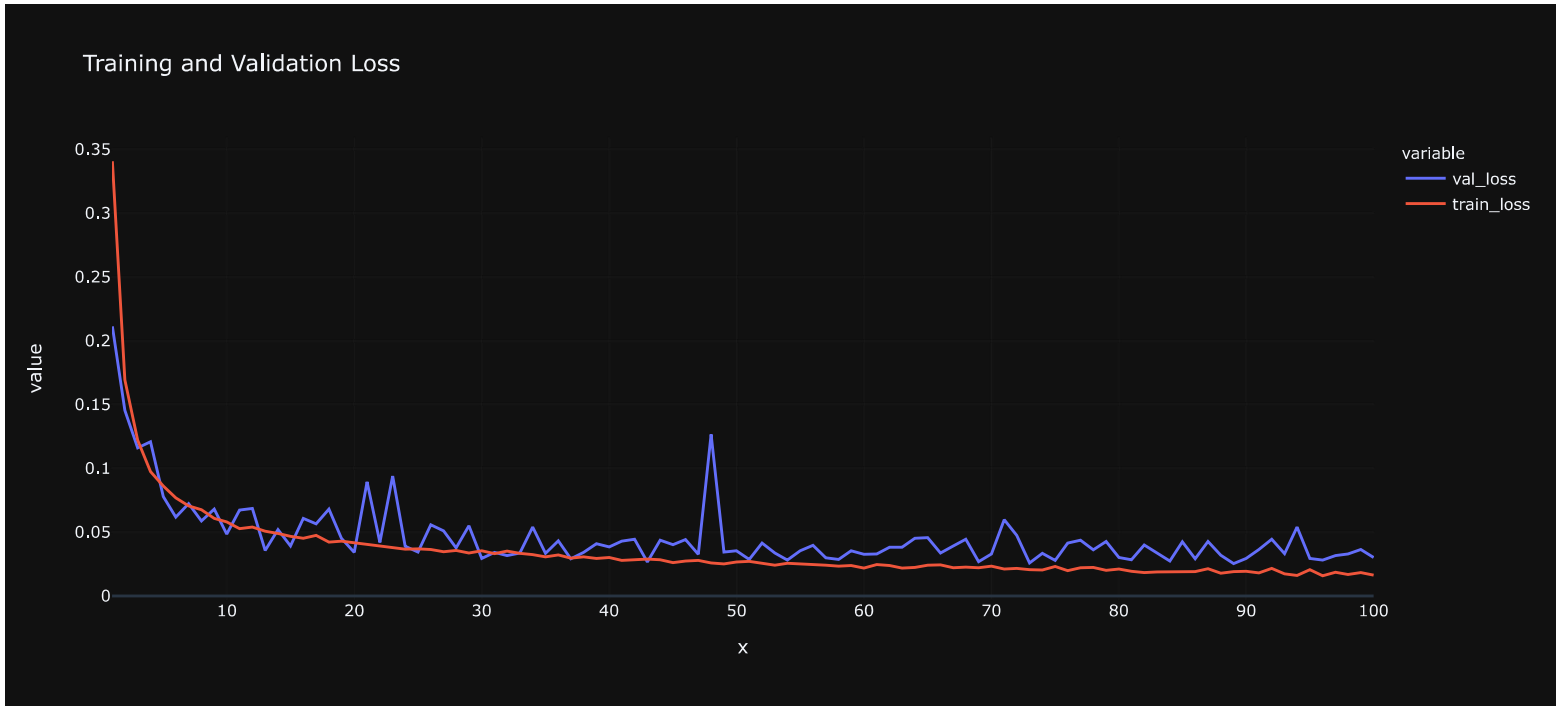
```
plt.figure(figsize = (12, 6))

train_loss = model_history.history['loss']
```

```
val_loss = model_history.history['val_loss']
epoch = range(1, 101)

loss = pd.DataFrame({'train_loss' : train_loss, 'val_loss' : val_loss})

px.line(data_frame = loss, x = epoch, y = ['val_loss', 'train_loss'], title = 'Training and Validation Loss',
        template = 'plotly_dark')
```



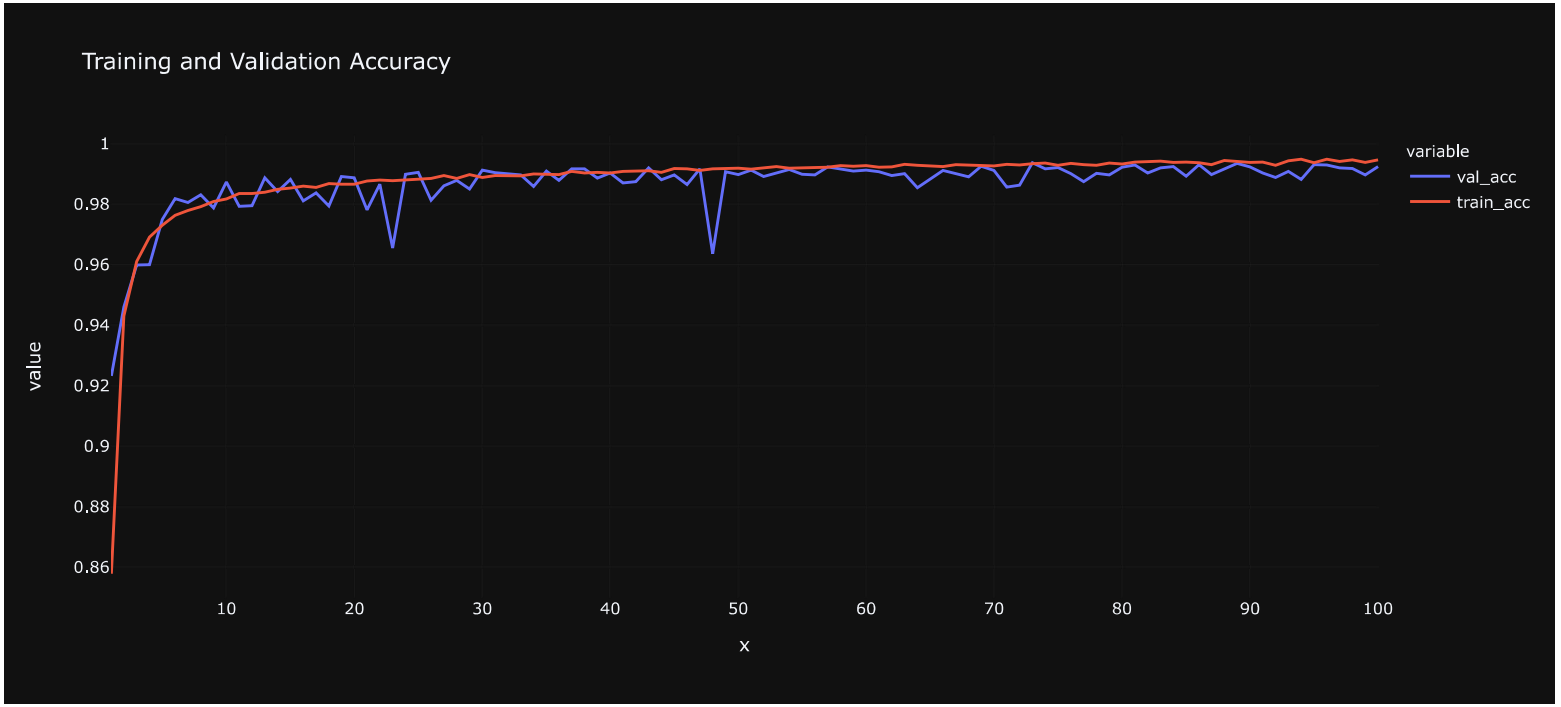

```
<Figure size 864x432 with 0 Axes>
```

In [73]:
```
plt.figure(figsize = (12, 6))

train_acc = model_history.history['accuracy']
val_acc = model_history.history['val_accuracy']
epoch = range(1, 101)


accuracy = pd.DataFrame({'train_acc' : train_acc, 'val_acc' : val_acc})

px.line(data_frame = accuracy, x = epoch, y = ['val_acc', 'train_acc'], title = 'Training and Validation Accuracy',
        template = 'plotly_dark')
```



<Figure size 864x432 with 0 Axes>

In [74]:
```
acc_ann = model.evaluate(X_test, y_test)[1]

print(f'Accuracy of model is {acc_ann}')
```

```
1118/1118 [==============================] - 2s 2ms/step - loss: 0.0300 - accuracy: 0.9924
Accuracy of model is 0.9923664331436157
```

Models Comparison

In [75]:
```
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'
                , 'ANN'],
    'Score' : [acc_lr, acc_knn, acc_dtc, acc_rd_clf, acc_ada, acc_gb, acc_xgb, acc_cat, acc_etc, acc_lgbm, acc_vtc, acc_ann]
})


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

Out[75]:

	Model	Score
7	Cat Boost	0.995470
11	ANN	0.992366
6	XgBoost	0.983586
10	Voting Classifier	0.964181
3	Random Forest Classifier	0.954618
8	Extra Trees Classifier	0.953947
9	LGBM	0.947404
2	Decision Tree Classifier	0.943489
4	Ada Boost Classifier	0.943237
5	Gradient Boosting Classifier	0.913654
1	KNN	0.890753
0	Logistic Regression	0.814138

In [76]:
```
px.bar(data_frame = models, x = 'Score', y = 'Model', color = 'Score', template = 'plotly_dark', title = 'Models Comparison')
```