```
import pandas as pd #data preprocessing
import numpy as np #linear algebra
import matplotlib.pyplot as plt #matplotlib work like MATLAB
import seaborn as sns#uses Matplotlib underneath to plot graphs
import warnings#to ignore warnings
warnings.filterwarnings('ignore')


## Importing Data
df = pd.read_csv("/content/drive/MyDrive/hotel_bookings.csv")


df.head()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	s
0	Resort Hotel	0	342	2015	July	27	1	
1	Resort Hotel	0	737	2015	July	27	1	
2	Resort Hotel	0	7	2015	July	27	1	
3	Resort Hotel	0	13	2015	July	27	1	
4	Resort Hotel	0	14	2015	July	27	1	

5 rows × 32 columns

```
df.describe()
```

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_ni
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.00
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.92
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.99
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.00
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.00
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.00
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.00
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.00

```
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
```

```
df.duplicated().sum()
```

```
31994
```

```
df.dtypes
```

```
hotel                            object
is_canceled                       int64
lead_time                         int64
arrival_date_year                 int64
arrival_date_month               object
arrival_date_week_number          int64
arrival_date_day_of_month         int64
stays_in_weekend_nights           int64
stays_in_week_nights              int64
adults                            int64
children                        float64
babies                            int64
meal                             object
country                          object
```

```
market_segment                  object
distribution_channel            object
is_repeated_guest                int64
previous_cancellations           int64
previous_bookings_not_canceled   int64
reserved_room_type              object
assigned_room_type              object
booking_changes                  int64
deposit_type                    object
agent                          float64
company                        float64
days_in_waiting_list             int64
customer_type                   object
adr                            float64
required_car_parking_spaces      int64
total_of_special_requests        int64
reservation_status              object
reservation_status_date         object
dtype: object
```

```
df.dtypes.value_counts()
```

```
int64      16
object     12
float64     4
dtype: int64
```

```
import warnings
warnings.filterwarnings('ignore')
```

Defining Target and Independent Feature

```
Y = df[['is_canceled']]
X = df.drop(['is_canceled'], axis=1)
```

Get Cancellation Rate

```
Y.mean()
```

```
is_canceled    0.370416
dtype: float64
```

Split Featres into Numerical and Categorical

```
num = X.select_dtypes(include='number')
char = X.select_dtypes(include='object')


# check for how many unique values each column has
def unique_levels(x):
    x = x.value_counts().count()
    return x

df_value_counts = pd.DataFrame(num.apply(lambda X: unique_levels(X)))


df_value_counts.columns = ['feature_levels']
df_value_counts.sort_values(by = 'feature_levels', ascending=False)
```

	feature_levels
adr	8879
lead_time	479
company	352
agent	333
days_in_waiting_list	128
previous_bookings_not_canceled	73
arrival_date_week_number	53
stays_in_week_nights	35
arrival_date_day_of_month	31
booking_changes	21
stays_in_weekend_nights	17
previous_cancellations	15

```
slice1 = df_value_counts.loc[df_value_counts['feature_levels']<=20]
cat_list = slice1.index
cat = num.loc[:, cat_list]
cat.dtypes
```

```
arrival_date_year               int64
stays_in_weekend_nights         int64
adults                          int64
children                      float64
babies                          int64
is_repeated_guest               int64
previous_cancellations          int64
required_car_parking_spaces     int64
total_of_special_requests       int64
dtype: object
```

```
slice2 = df_value_counts.loc[df_value_counts['feature_levels']>20]
num_list = slice2.index
num = num.loc[:, num_list]
num.dtypes
```

```
lead_time                         int64
arrival_date_week_number          int64
arrival_date_day_of_month         int64
stays_in_week_nights              int64
previous_bookings_not_canceled    int64
booking_changes                   int64
agent                           float64
company                         float64
days_in_waiting_list              int64
adr                             float64
dtype: object
```

```
char = pd.concat([char, cat], axis = 1, join = 'inner')
```

```
char.dtypes
```

```
hotel                        object
arrival_date_month           object
meal                         object
country                      object
market_segment               object
distribution_channel         object
reserved_room_type           object
assigned_room_type           object
deposit_type                 object
customer_type                object
reservation_status           object
reservation_status_date      object
arrival_date_year             int64
stays_in_weekend_nights       int64
adults                        int64
children                    float64
babies                        int64
```

```
is_repeated_guest               int64
previous_cancellations          int64
required_car_parking_spaces     int64
total_of_special_requests       int64
dtype: object
```

Outliers Analysis of Numerical Features

```
num.describe(percentiles=[0.01,0.05,0.10,0.25,0.50,0.75,0.85,0.88,0.90,0.99])
```

```
def outlier_cap(x):
    x = x.clip(lower = x.quantile(0.01))
    x = x.clip(upper = x.quantile(0.99))
    return (x)
```

```
num = num.apply(lambda x: outlier_cap(x))
```

```
num.describe(percentiles=[0.01,0.05,0.10,0.25,0.50,0.75,0.85,0.88,0.90,0.99])
```

	lead_time	arrival_date_week_number	arrival_date_day_of_month	stays_in_we
mean	103.409180	27.173943	15.798241	

Missing Values Analysis

```
num.isnull().mean()
```

```
lead_time                       0.000000
arrival_date_week_number        0.000000
arrival_date_day_of_month       0.000000
stays_in_week_nights            0.000000
previous_bookings_not_canceled  0.000000
booking_changes                 0.000000
agent                           0.136862
company                         0.943069
days_in_waiting_list            0.000000
adr                             0.000000
dtype: float64
```

```
char.isnull().sum()
```

```
hotel                         0
arrival_date_month            0
meal                          0
country                     488
market_segment                0
distribution_channel          0
reserved_room_type            0
assigned_room_type            0
deposit_type                  0
customer_type                 0
reservation_status            0
reservation_status_date       0
arrival_date_year             0
stays_in_weekend_nights       0
adults                        0
children                      4
babies                        0
is_repeated_guest             0
```

```
previous_cancellations          0
required_car_parking_spaces     0
total_of_special_requests       0
dtype: int64
```

Dropping Variables that have >25% missing values

```
num = num.loc[:, num.isnull().mean() <= 0.25 ]
```

```
num.isnull().mean()
```

```
lead_time                       0.000000
arrival_date_week_number        0.000000
arrival_date_day_of_month       0.000000
stays_in_week_nights            0.000000
previous_bookings_not_canceled  0.000000
booking_changes                 0.000000
agent                           0.136862
days_in_waiting_list            0.000000
adr                             0.000000
dtype: float64
```

Imputation of Missing Values

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
num_1 = pd.DataFrame(imputer.fit_transform(num), index = num.index, columns=num.columns)
```

```
num_1.isnull().mean()
```

```
lead_time                       0.0
arrival_date_week_number        0.0
arrival_date_day_of_month       0.0
stays_in_week_nights            0.0
```

```
previous_bookings_not_canceled    0.0
booking_changes                   0.0
agent                             0.0
days_in_waiting_list              0.0
adr                               0.0
dtype: float64
```

```
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
char_1 = pd.DataFrame(imputer.fit_transform(char), index = char.index, columns=char.columns)
```

```
char_1.isnull().sum()
```

```
hotel                           0
arrival_date_month              0
meal                            0
country                         0
market_segment                  0
distribution_channel            0
reserved_room_type              0
assigned_room_type              0
deposit_type                    0
customer_type                   0
reservation_status              0
reservation_status_date         0
arrival_date_year               0
stays_in_weekend_nights         0
adults                          0
children                        0
babies                          0
is_repeated_guest               0
previous_cancellations          0
required_car_parking_spaces     0
total_of_special_requests       0
dtype: int64
```

Feature Selection - Numerical Features

Part 1: Remove Features with Zero Variance

```
from sklearn.feature_selection import VarianceThreshold
varselector = VarianceThreshold(threshold = 0)
varselector.fit_transform(num_1)

#get column indices to keep and create new df with those columns only
cols = varselector.get_support(indices=True)
num_2 = num_1.iloc[:,cols]
```

```
num_2.iloc[0]
```

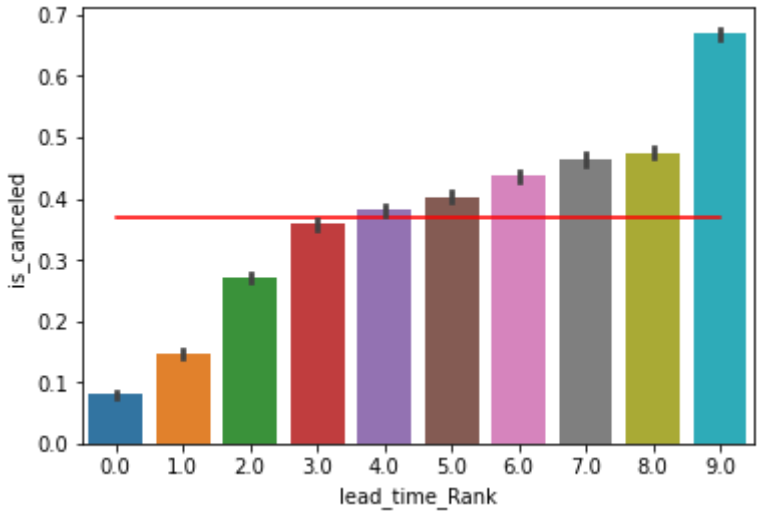
```
lead_time                        342.000000
arrival_date_week_number          27.000000
arrival_date_day_of_month          1.000000
stays_in_week_nights               0.000000
previous_bookings_not_canceled     0.000000
booking_changes                    3.000000
agent                             86.008598
days_in_waiting_list               0.000000
adr                                0.000000
Name: 0, dtype: float64
```

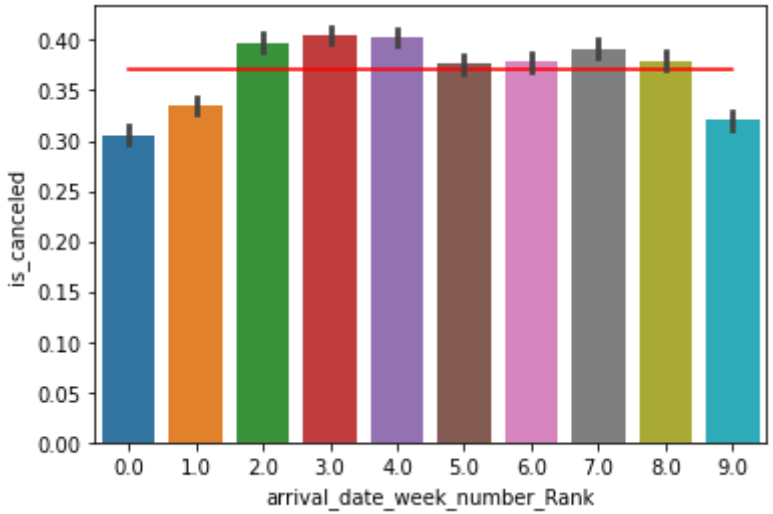
Part 2: Bivariate Analysis (Feature Discretization)

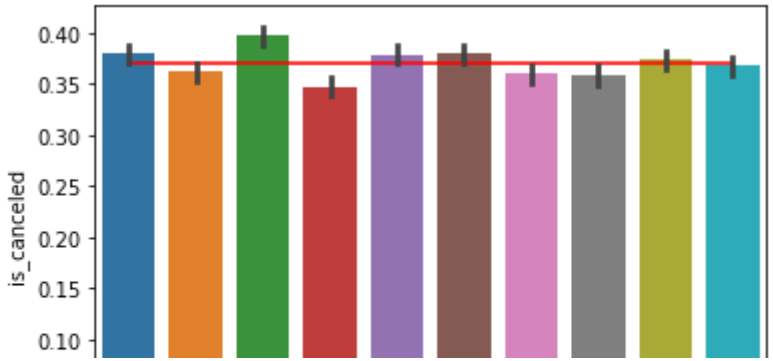
```
from sklearn.preprocessing import KBinsDiscretizer
discrete = KBinsDiscretizer(n_bins = 10, encode = 'ordinal', strategy = 'quantile')
num_binned = pd.DataFrame(discrete.fit_transform(num_2), index=num_2.index, columns = num_2.columns).add_suffix('_Rank')
num_binned.tail()
```

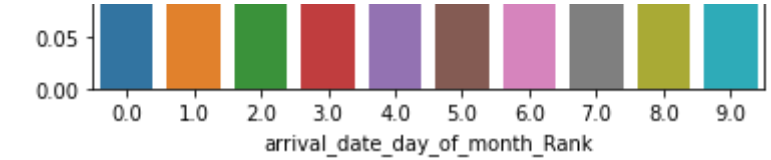
	lead_time_Rank	arrival_date_week_number_Rank	arrival_date_day_of_month_Rank
119385	2.0	6.0	9.0
119386	6.0	6.0	9.0
119387	3.0	6.0	9.0

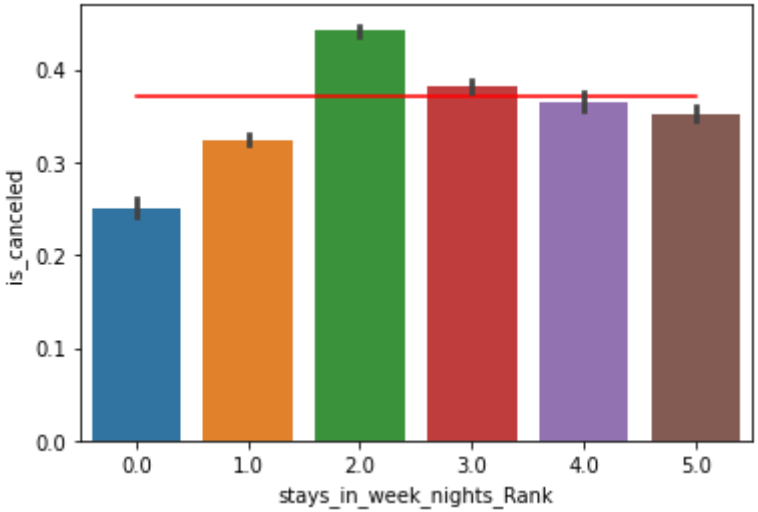
```
# Check if the features show a slope at all
#If they do, then do you see some deciles below the population average and some higher than population average?
#If that is the case then the slope will be strong
#Conclusion: A strong slope is indicative of the features' ability to discriminate the event from non event
#            making it a good predictor

X_bin_combined = pd.concat([Y,num_binned], axis = 1, join = 'inner')

from numpy import mean
for col in num_binned.columns:
    plt.figure()
    sns.lineplot(x=col, y = X_bin_combined['is_canceled'].mean(), data=X_bin_combined, color='red')
    sns.barplot(x=col, y='is_canceled', data=X_bin_combined, estimator = mean)
plt.show()
```

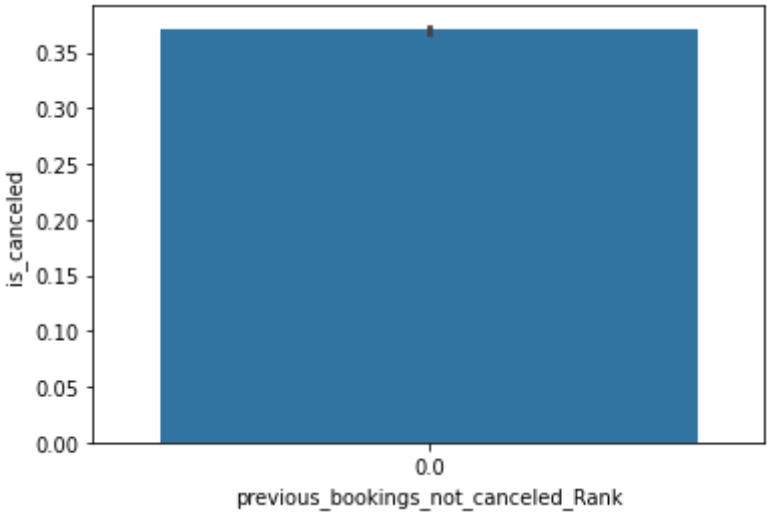

0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
is_canceled
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
lead_time_Rank

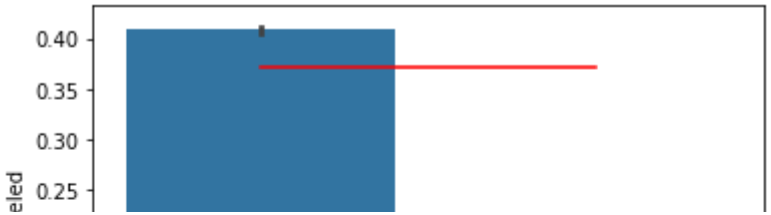

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
is_canceled
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
arrival_date_week_number_Rank


0.40
0.35
0.30
0.25
0.20
0.15
0.10
is_canceled

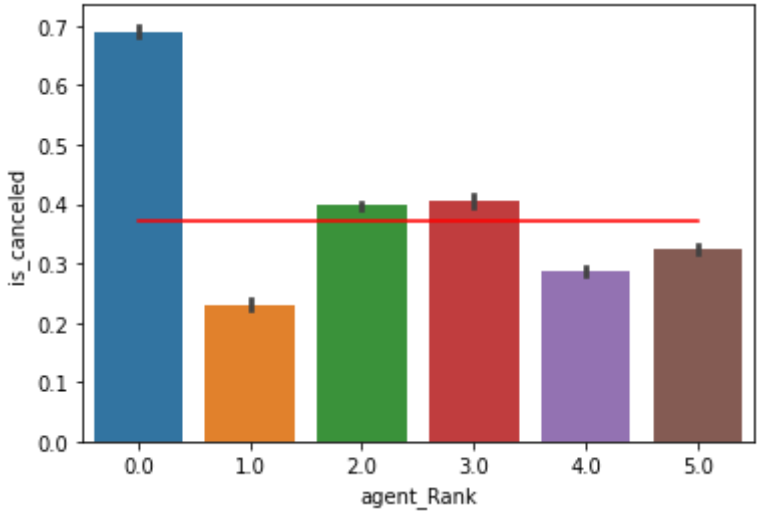

0.05
0.00
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
arrival_date_day_of_month_Rank

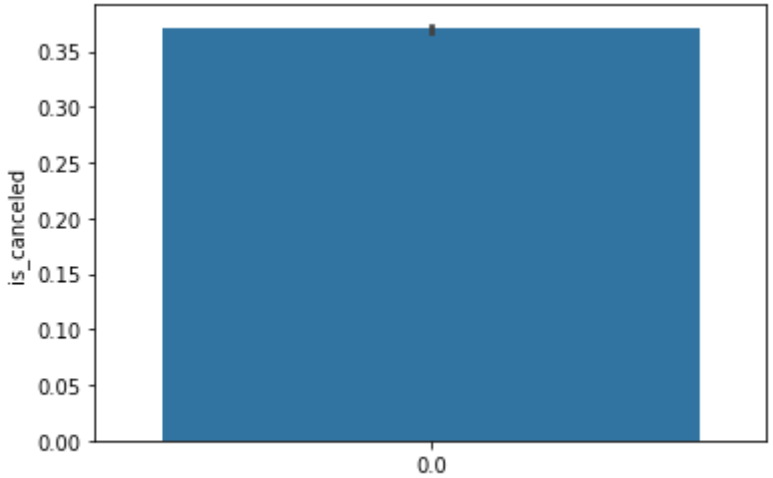

0.4
0.3
0.2
0.1
0.0
is_canceled
0.0
1.0
2.0
3.0
4.0
5.0
stays_in_week_nights_Rank


0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
is_canceled
0.0
previous_bookings_not_canceled_Rank


0.40
0.35
0.30
0.25
eled


is_canc
0.20
0.15
0.10
0.05
0.00
0.0
1.0
booking_changes_Rank


is_canceled
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
1.0
2.0
3.0
4.0
5.0
agent_Rank


is_canceled
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0

```
num.columns
```

```
      Index(['lead_time', 'arrival_date_week_number', 'arrival_date_day_of_month',
             'stays_in_week_nights', 'previous_bookings_not_canceled',
             'booking_changes', 'agent', 'days_in_waiting_list', 'adr'],
            dtype='object')
```

```
# Check the descriptive stats for the following feature
# previous_bookings_not_canceled
# days_in_waiting_list
# booking_changes

num_2['day_wait_ind'] = np.where(num_2['days_in_waiting_list']>0, 1, 0)
num_2['previous_bookings_not_canceled_ind'] = np.where(num_2['previous_bookings_not_canceled']>0, 1, 0)
num_2['booking_changes_ind'] = np.where(num_2['booking_changes']<0, 1, 0)


num_varlist = ['arrival_date_day_of_month', 'arrival_date_week_number', 'days_in_waiting_list', 'previous_bookings_not_canceled',
               'booking_changes']

num_2 = num_2.drop(num_varlist, axis=1)


num_2.dtypes
```

```
     lead_time                            float64
     stays_in_week_nights                 float64
     agent                                float64
     adr                                  float64
     day_wait_ind                           int64
     previous_bookings_not_canceled_ind     int64
     booking_changes_ind                    int64
     dtype: object
```

Part 3 - Select K Best

```
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2, k=4)
selector.fit_transform(num_2, Y)
```

```
#get columns to create new df with them only
cols = selector.get_support(indices=True)
select_features_df_num = num_2.iloc[:,cols]
```

```
select_features_df_num.iloc[0]
```

```
lead_time                            342.000000
agent                                 86.008598
adr                                    0.000000
previous_bookings_not_canceled_ind     0.000000
Name: 0, dtype: float64
```

Feature Selection - Categorical Features

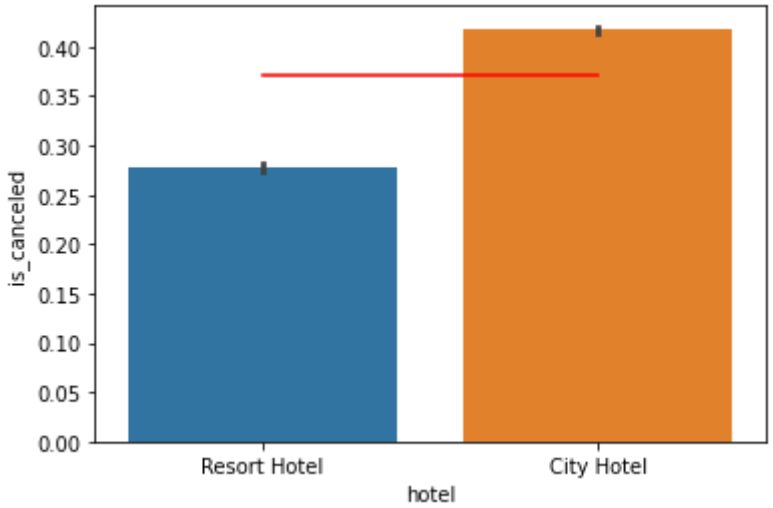
```
#checking for level of each values
char_1.nunique()
```

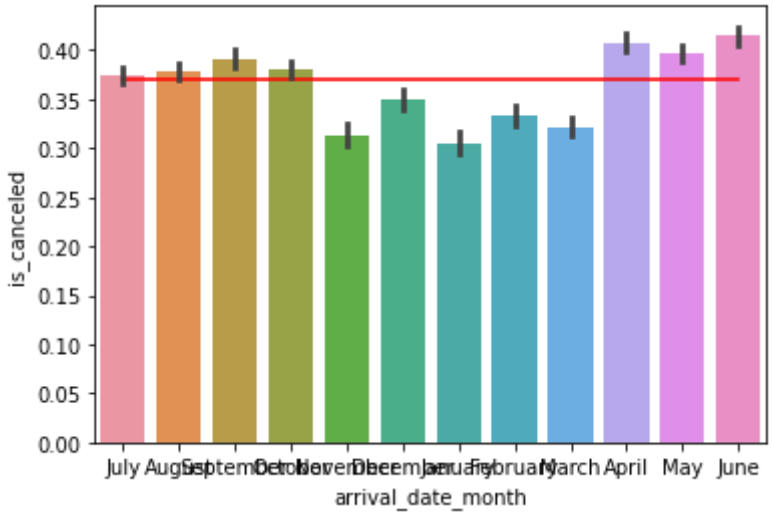
```
hotel                          2
arrival_date_month            12
meal                           5
country                      177
market_segment                 8
distribution_channel           5
reserved_room_type            10
assigned_room_type            12
deposit_type                   3
customer_type                  4
reservation_status             3
reservation_status_date      926
arrival_date_year              3
stays_in_weekend_nights       17
adults                        14
children                       5
babies                         5
is_repeated_guest              2
previous_cancellations        15
```

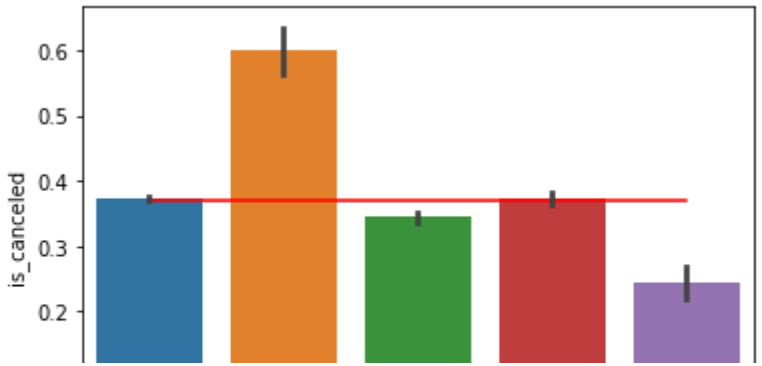
```
required_car_parking_spaces      5
total_of_special_requests        6
dtype: int64
```

Part 1 - Bi Variate Analysis

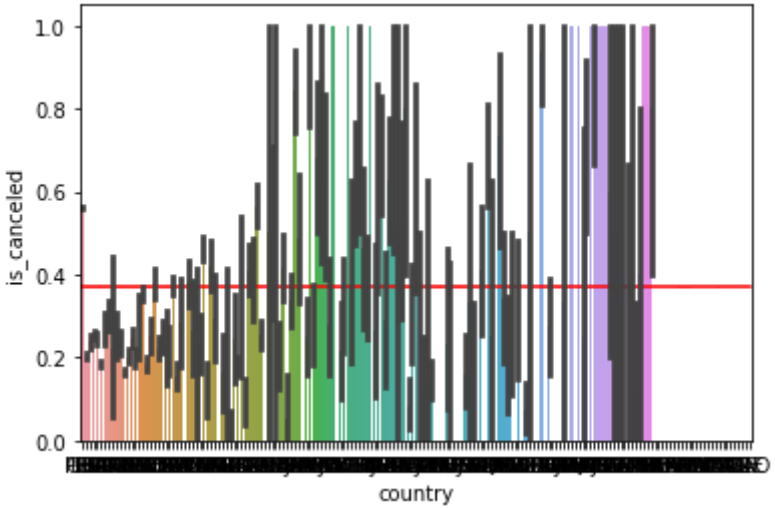
```
X_char_merged = pd.concat([Y,char_1], axis=1, join='inner')

for col in char.columns:
    plt.figure()
    sns.lineplot(x=col, y = X_char_merged['is_canceled'].mean(), data=X_char_merged, color='red')
    sns.barplot(x=col, y='is_canceled', data = X_char_merged, estimator=mean)
plt.show()
```

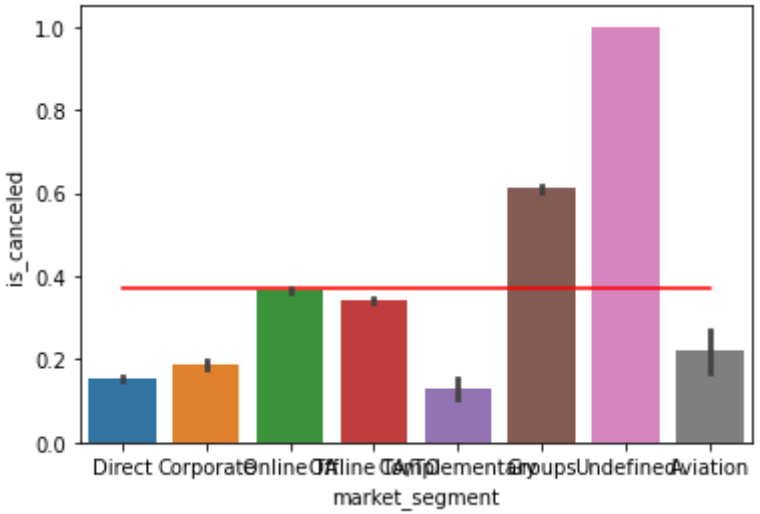

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
is_canceled
Resort Hotel
City Hotel
hotel


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
is_canceled
July
April
May
June
arrival_date_month

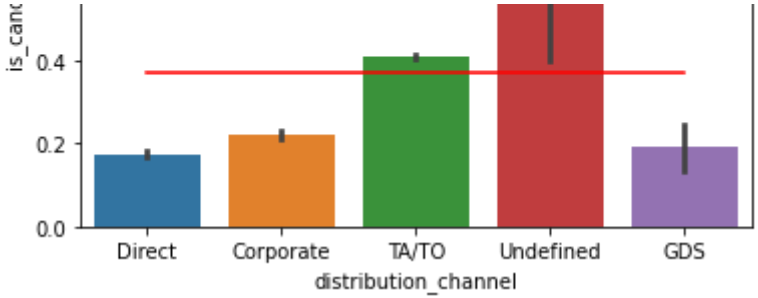

0.6
0.5
0.4
0.3
0.2
is_canceled

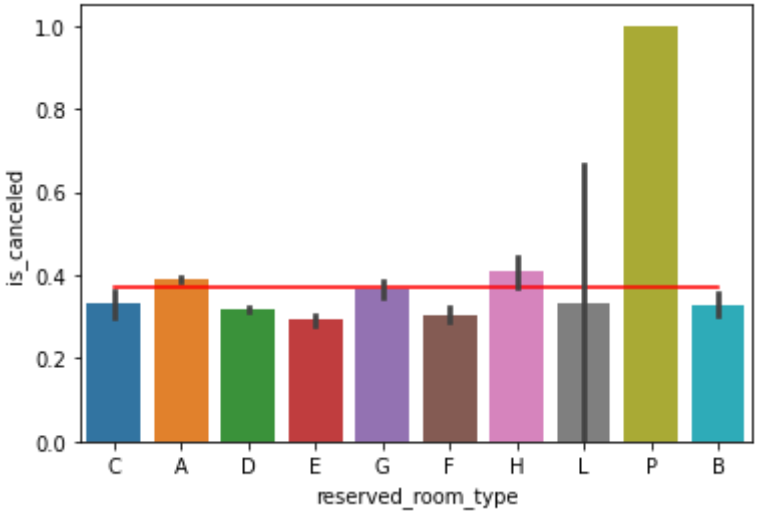

0.1
0.0
BB
FB
HB
SC
Undefined
meal

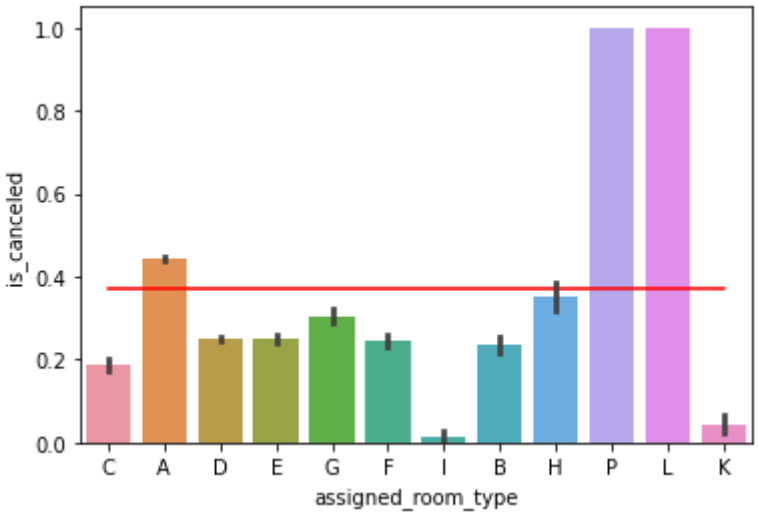

1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
country


1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
Direct
Corporate
Undefined
Aviation
market_segment

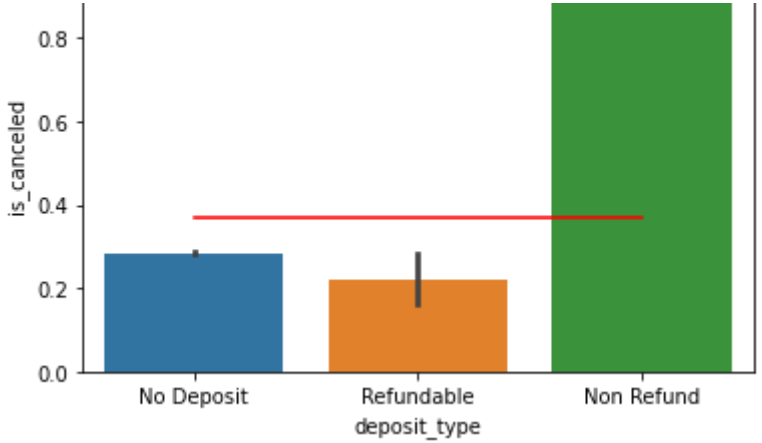

1.0
0.8
0.6

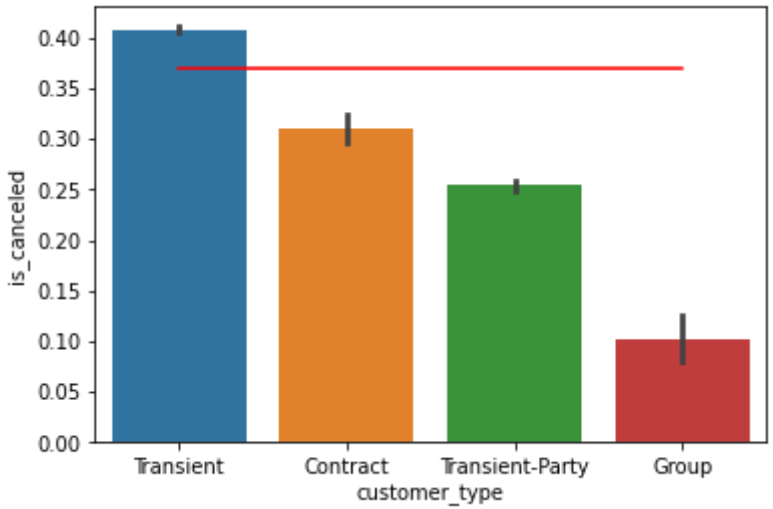

is_can
0.4
0.2
0.0
Direct
Corporate
TA/TO
Undefined
GDS
distribution_channel

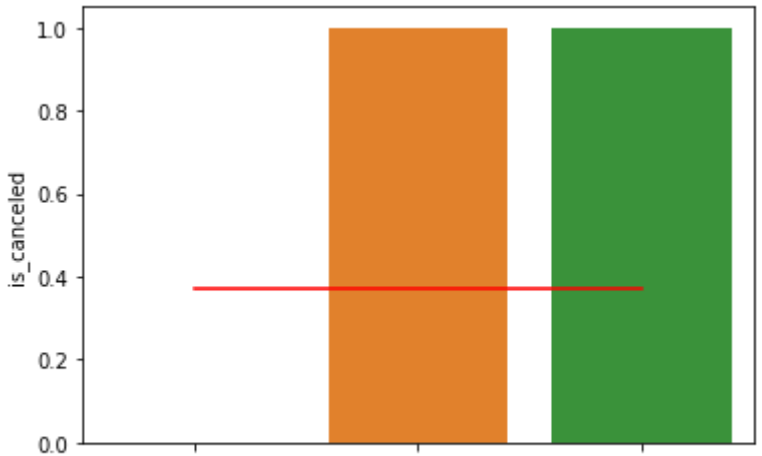

1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
C
A
D
E
G
F
H
L
P
B
reserved_room_type

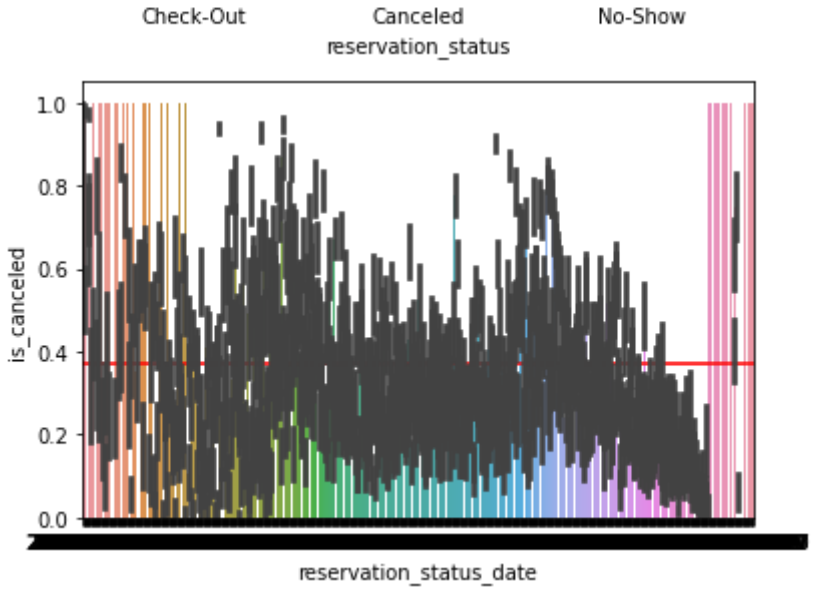

1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
C
A
D
E
G
F
I
B
H
P
L
K
assigned_room_type

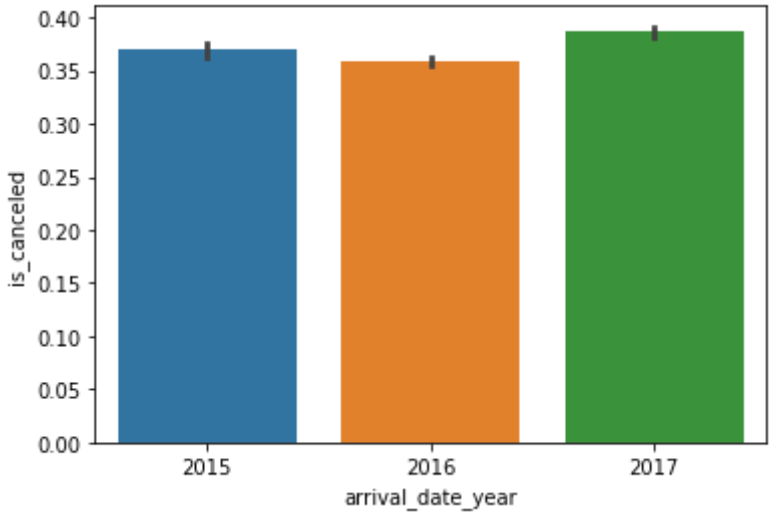

1.0

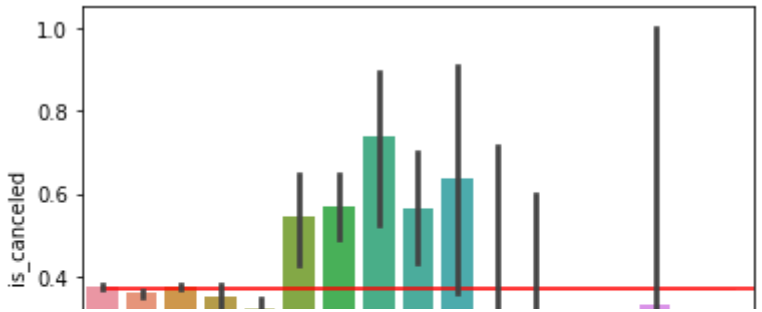

0.8
0.6
0.4
0.2
0.0
is_canceled
No Deposit
Refundable
Non Refund
deposit_type

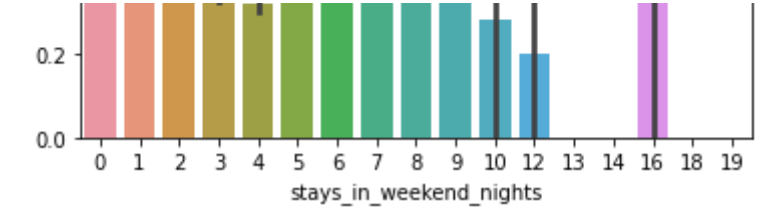

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
is_canceled
Transient
Contract
Transient-Party
Group
customer_type

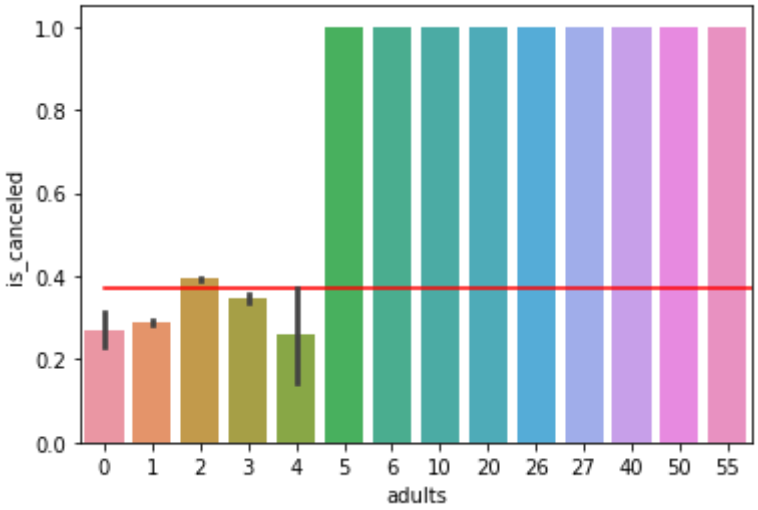

1.0
0.8
0.6
0.4
0.2
0.0
is_canceled

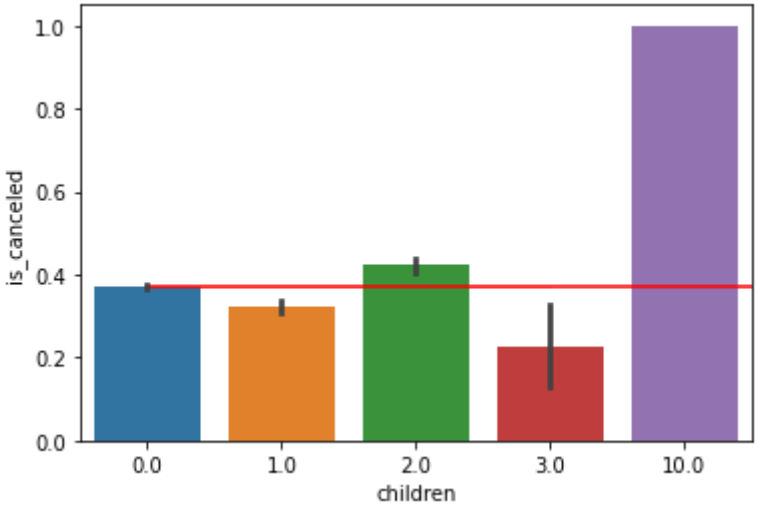

Check-Out
Canceled
No-Show
reservation_status
is_canceled
reservation_status_date


is_canceled
2015
2016
2017
arrival_date_year

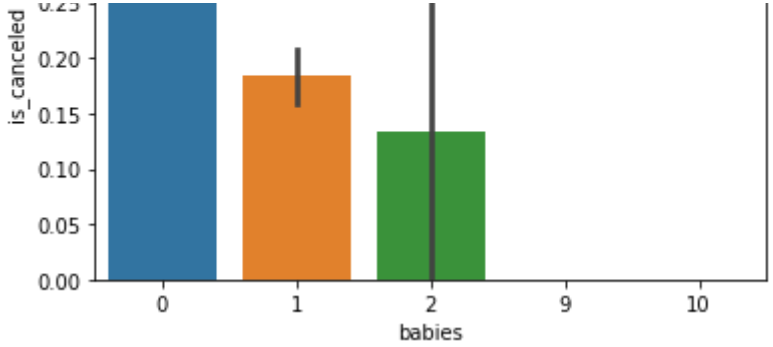

is_canceled

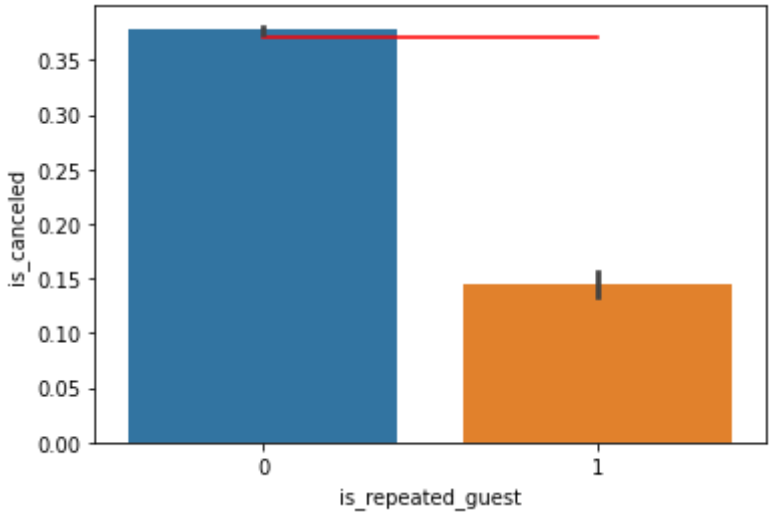

0.2
0.0
0 1 2 3 4 5 6 7 8 9 10 12 13 14 16 18 19
stays_in_weekend_nights

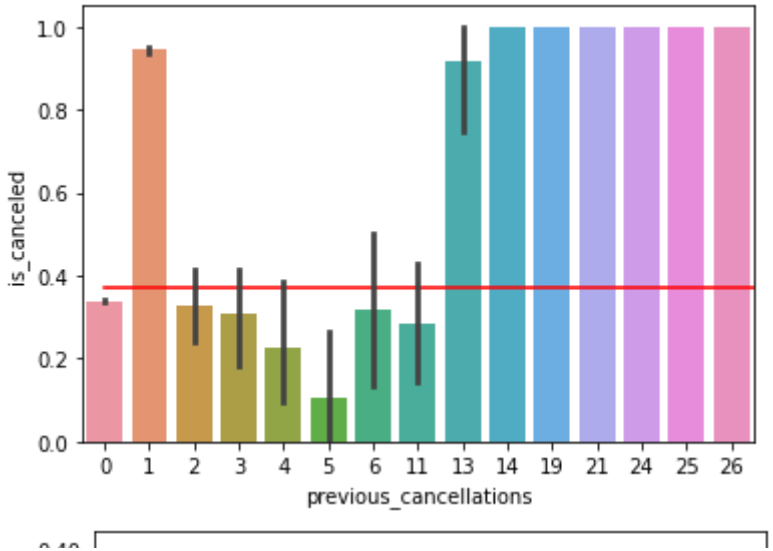

1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
0 1 2 3 4 5 6 10 20 26 27 40 50 55
adults


1.0
0.8
0.6
0.4
0.2
0.0
is_canceled
0.0 1.0 2.0 3.0 10.0
children


0.35
0.30


is_canceled
0.20
0.15
0.10
0.05
0.00
0
1
2
9
10
babies


is_canceled
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
is_repeated_guest


is_canceled
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
11
13
14
19
21
24
25
26
previous_cancellations


```
charlist = ['arrival_date_month', 'country', 'assigned_room_type', 'reservation_status',
            'reservation_status_date', 'arrival_date_year']
char_1 = char_1.drop(charlist, axis=1)
```


```
char_1.dtypes
```

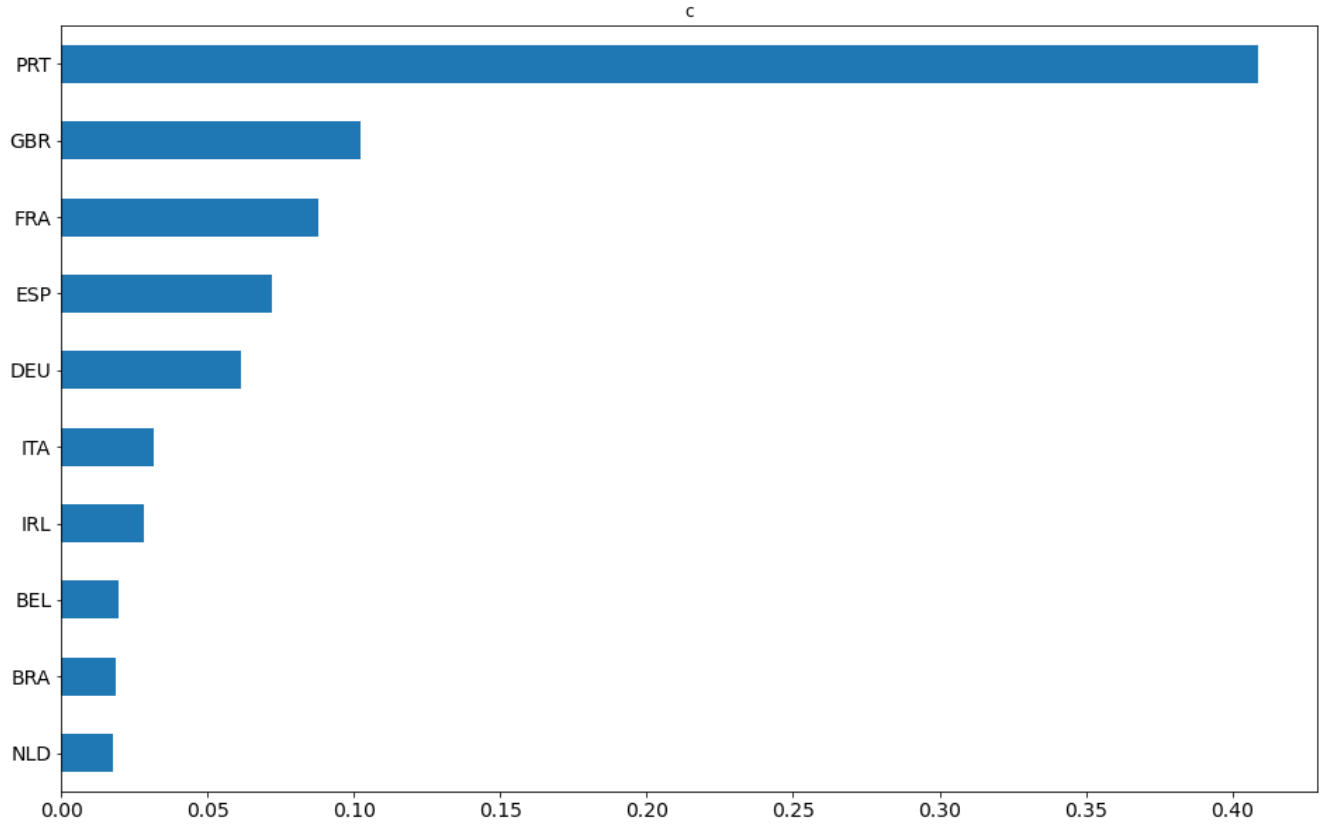
```
hotel                           object
meal                            object
market_segment                  object
distribution_channel            object
reserved_room_type              object
deposit_type                    object
customer_type                   object
stays_in_weekend_nights         object
adults                          object
children                        object
babies                          object
is_repeated_guest               object
previous_cancellations          object
required_car_parking_spaces     object
total_of_special_requests       object
dtype: object
```

```
# create dummy features with n-1 variables
X_char_dum = pd.get_dummies(char_1, drop_first=True)
X_char_dum.shape
```

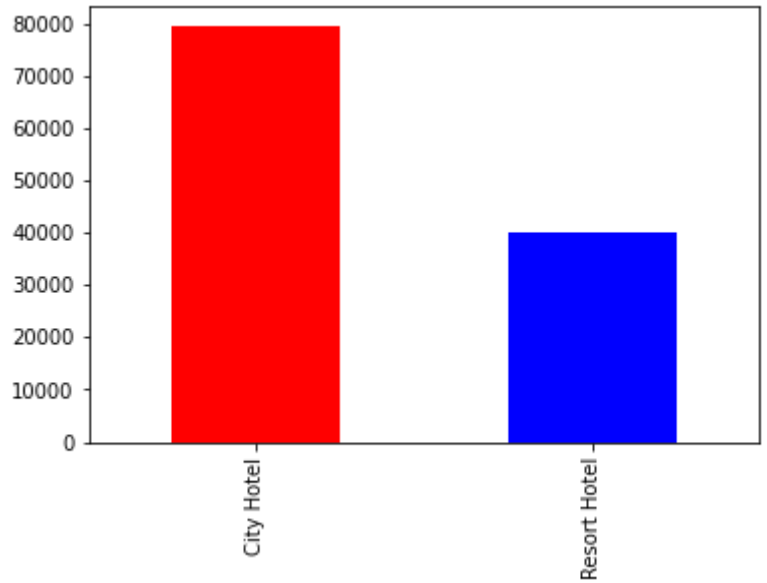
```
(119390, 91)
```

```
# Country univariant analysis only for top 10 countries
fig, axes = plt.subplots(1, 1, figsize=(16, 10))
```

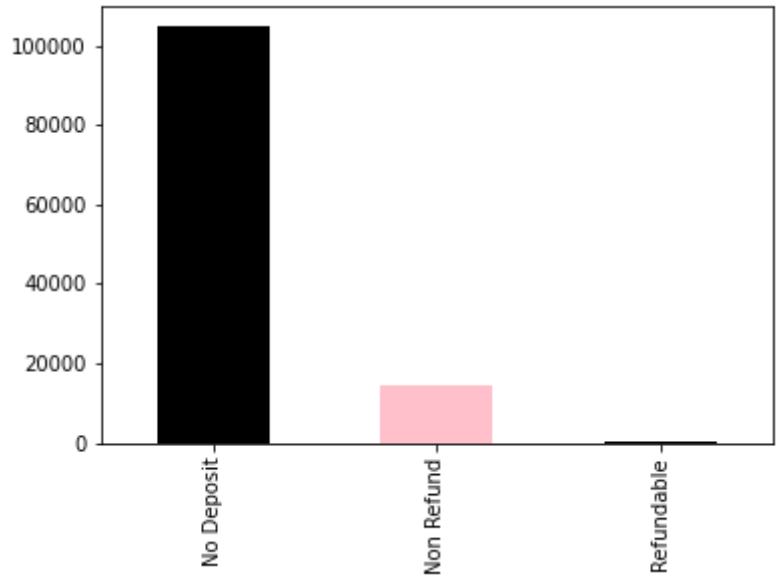
```
_ = df['country'].value_counts(normalize=True).head(10).sort_values(ascending=True).plot(kind = 'barh', title='c', fontsize=14)
```



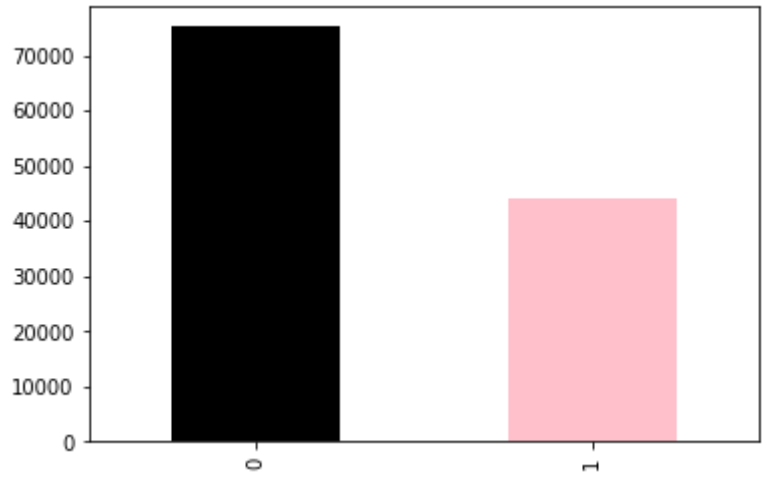

```
df["hotel"].value_counts().plot(kind="bar", color=["red", "blue"]);
```



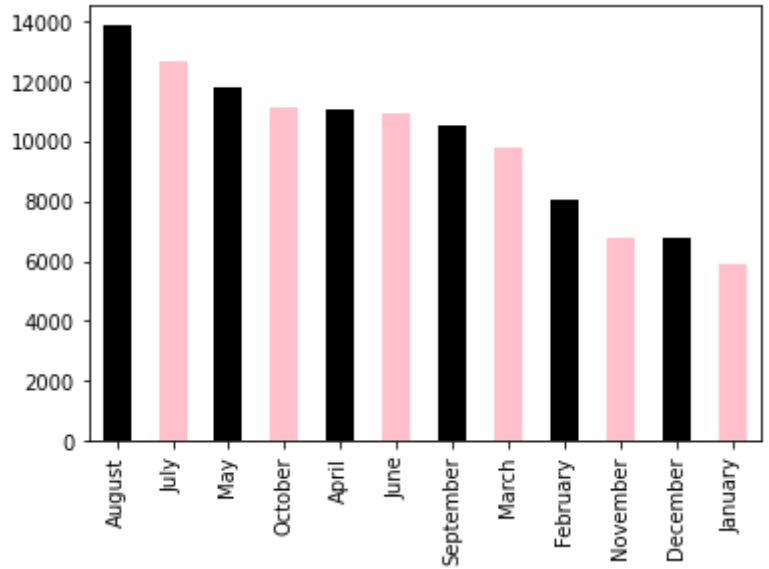

```
df["deposit_type"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```



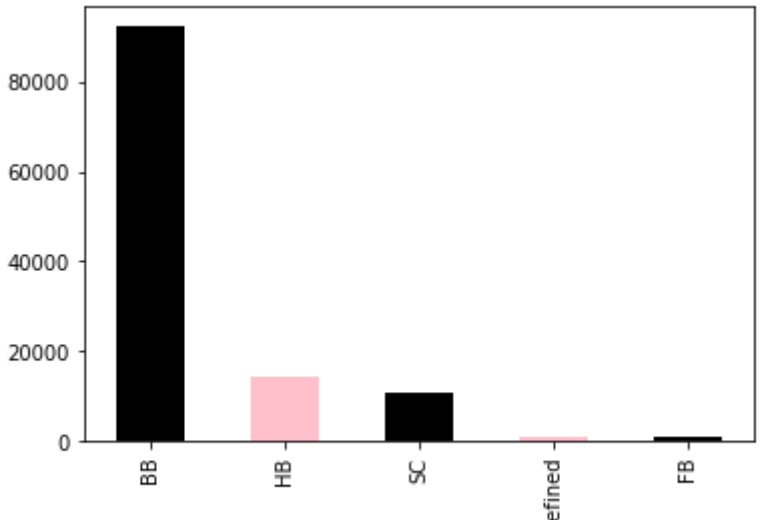

```
df["is_canceled"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```



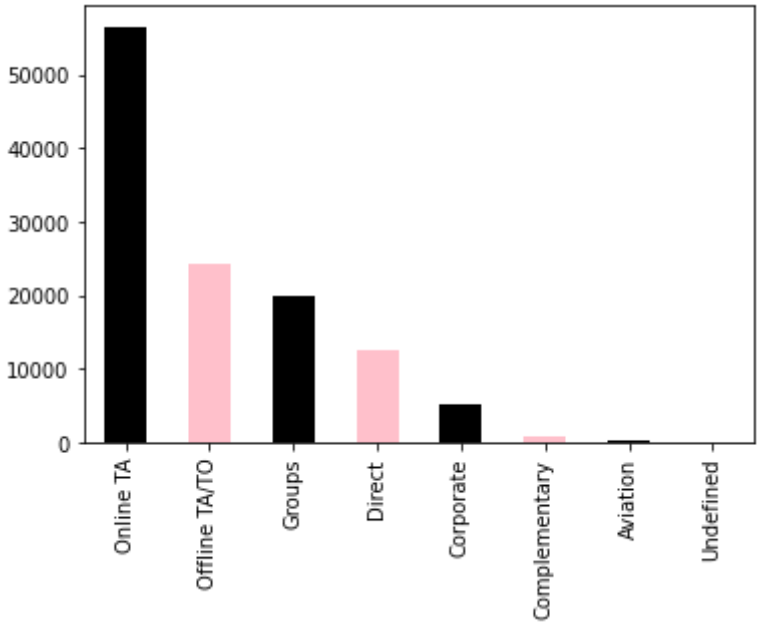

```
df["arrival_date_month"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```



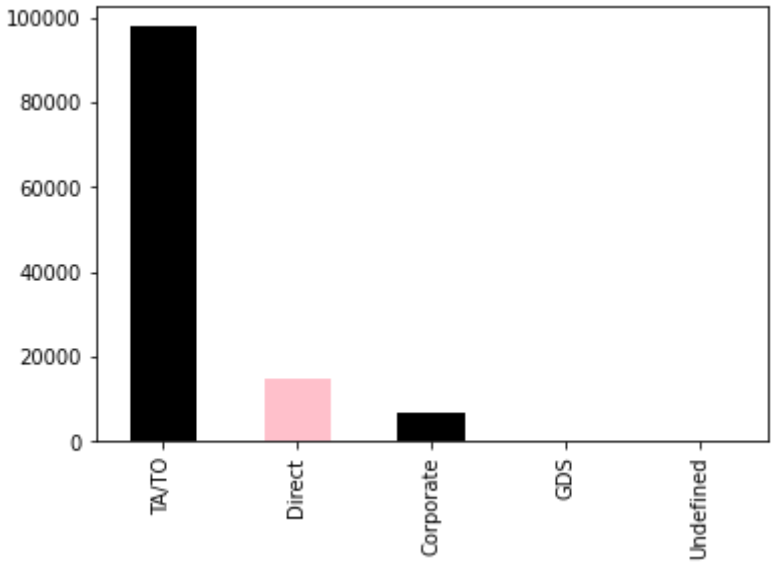

```
df["meal"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```



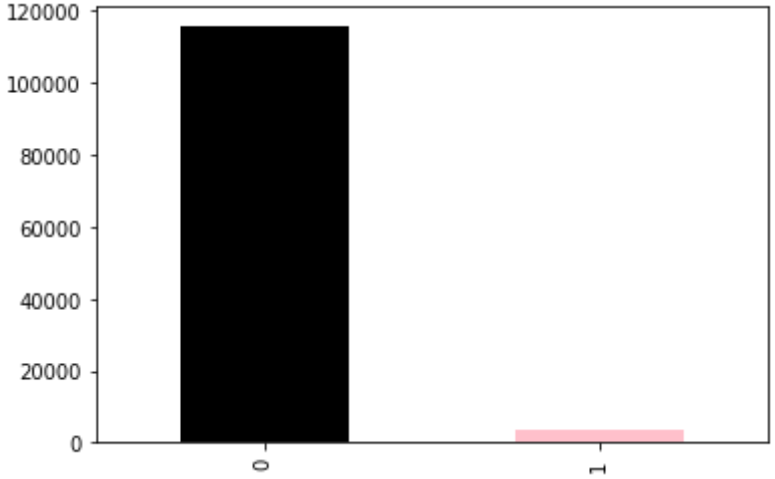

```
df["market_segment"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```




```
df["distribution_channel"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```

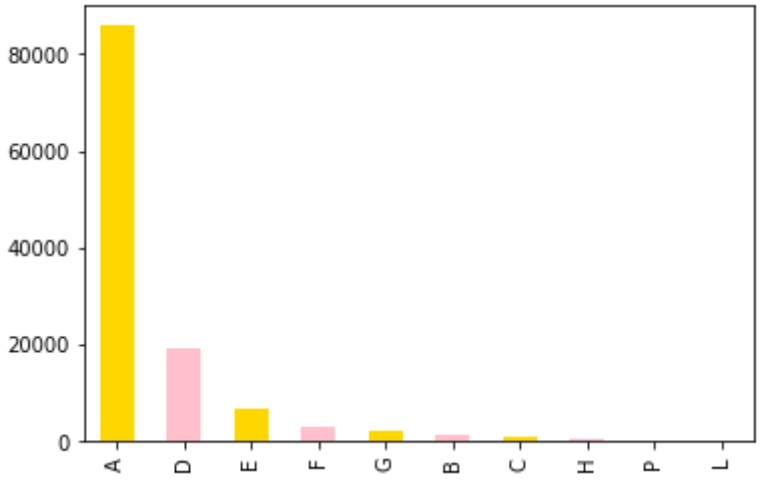



```
df["is_repeated_guest"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```



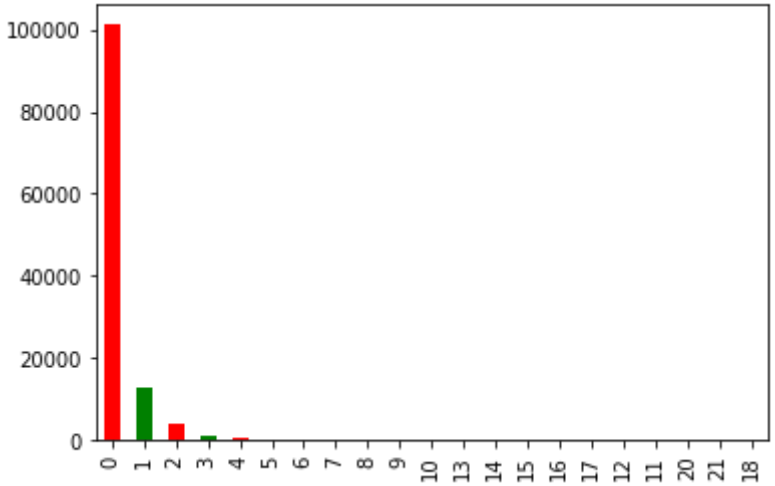

```
df["previous_cancellations"].value_counts().plot(kind="bar", color=["Yellow", "Pink"]);
```


```
df["reserved_room_type"].value_counts().plot(kind="bar", color=["Gold", "Pink"]);
```



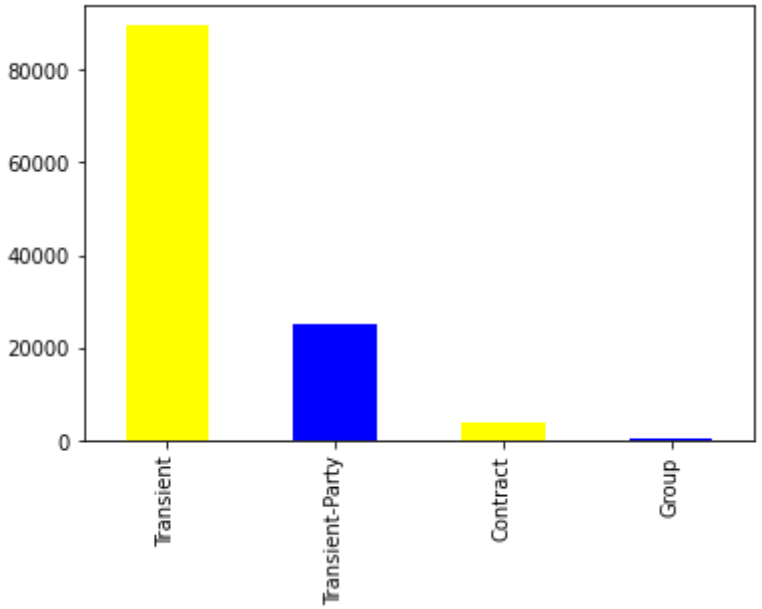

```
df["assigned_room_type"].value_counts().plot(kind="bar", color=["White", "Black"]);
```


```
df["booking_changes"].value_counts().plot(kind="bar", color=["Red", "Green"]);
```



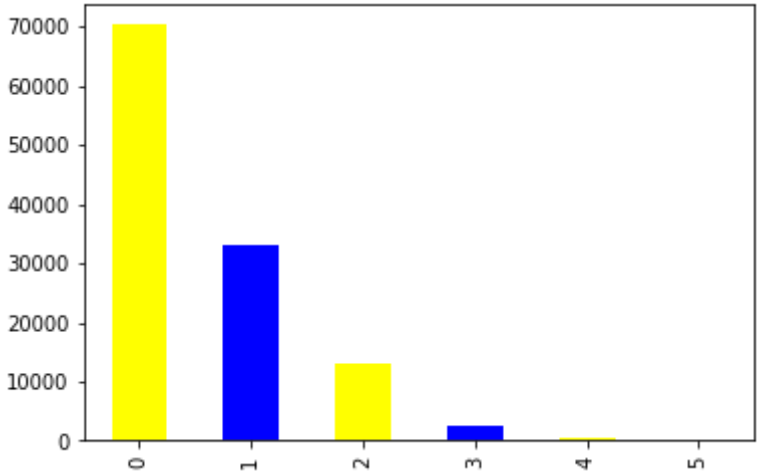

```
df["deposit_type"].value_counts().plot(kind="bar", color=["Yellow", "Blue"]);
```


```
df["customer_type"].value_counts().plot(kind="bar", color=["Yellow", "Blue"]);
```




```
df["required_car_parking_spaces"].value_counts().plot(kind="bar", color=["Yellow", "Blue"]);
```

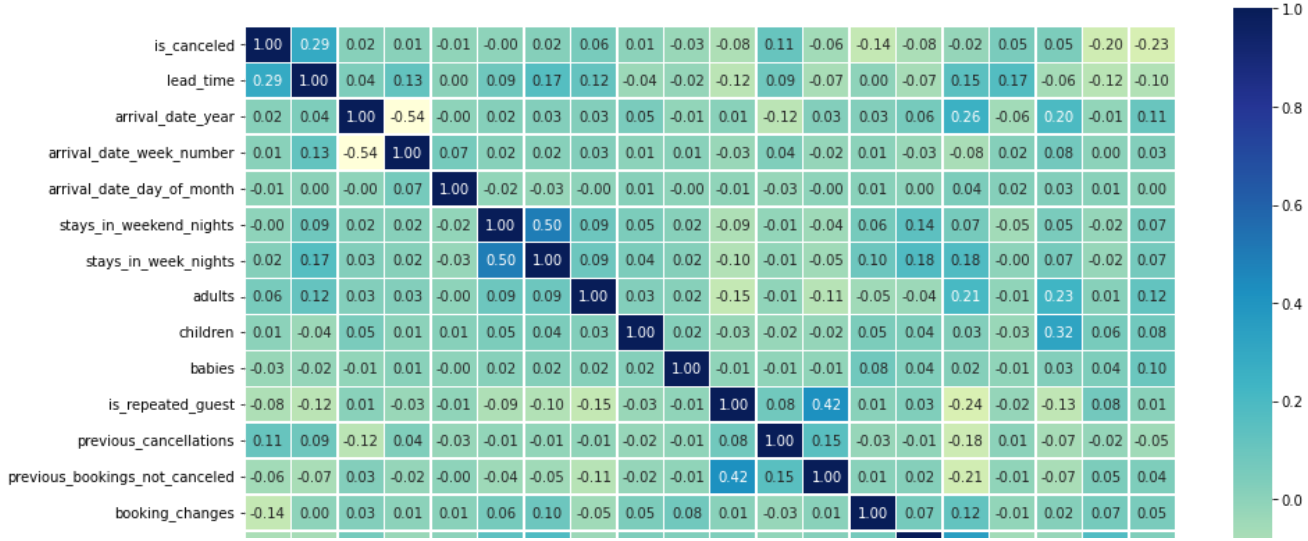

```
df["total_of_special_requests"].value_counts().plot(kind="bar", color=["Yellow", "Blue"]);
```



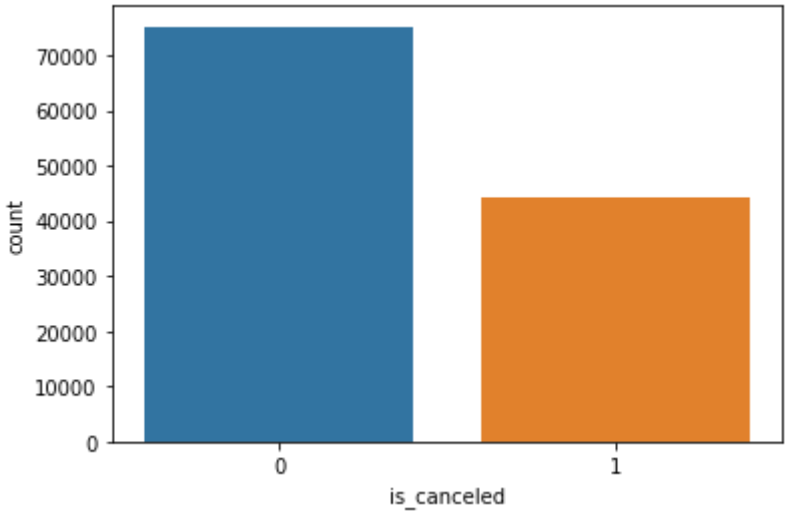

```
df["reservation_status"].value_counts().plot(kind="bar", color=["Yellow", "Blue"]);
```


```
corr_matrix = df.corr()
fig, ax =plt.subplots(figsize=(15,10))
ax = sns.heatmap(corr_matrix,
                 annot =True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom+0.5,top-0.5)
```

```
(20.5, -0.5)
```

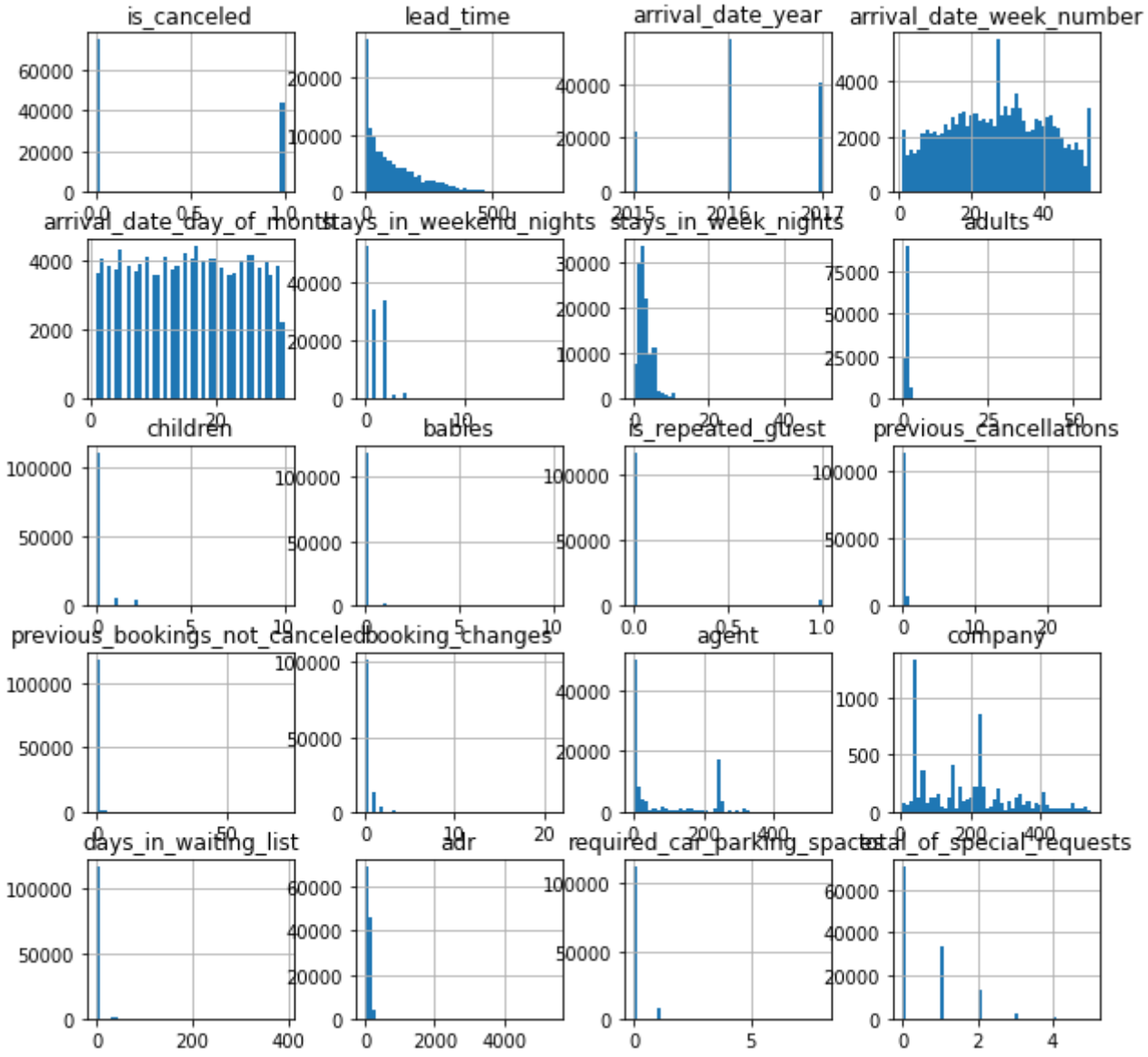



```
sns.countplot(df['is_canceled'])
plt.show()
```



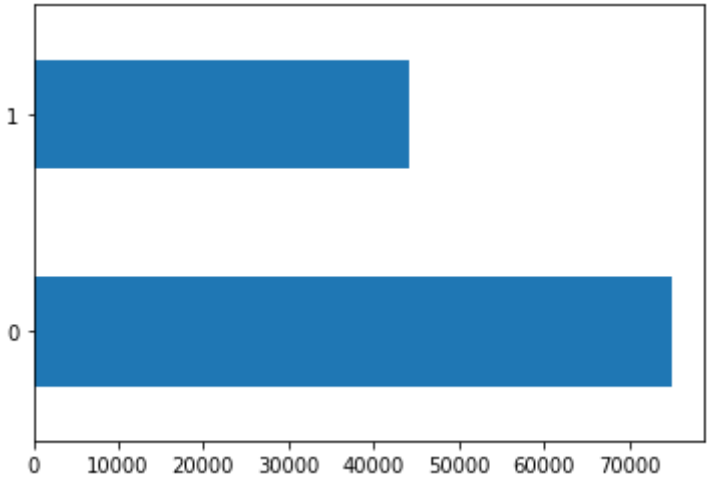

```
#df.plot(kind ='box',subplots = True, layout =(30,30),sharex = False)
```

```
#df.plot(kind ='density',subplots = True, layout =(30,30),sharex = False)

df.hist(figsize=(10,10),bins=50)
plt.show()
```



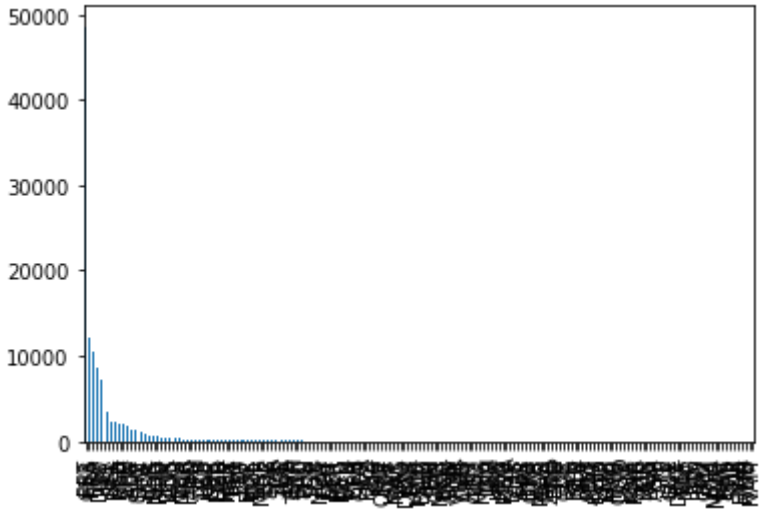
```
df.is_canceled.value_counts().plot.barh()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e66883e10>
```



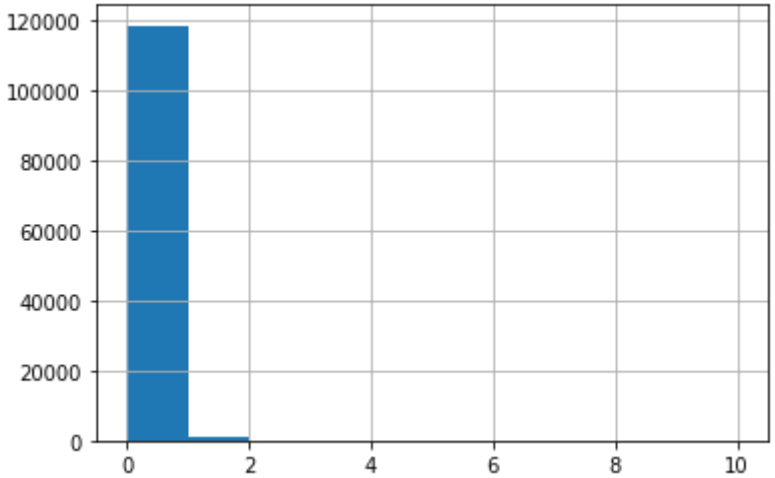

```
df.country.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e6753e7d0>
```



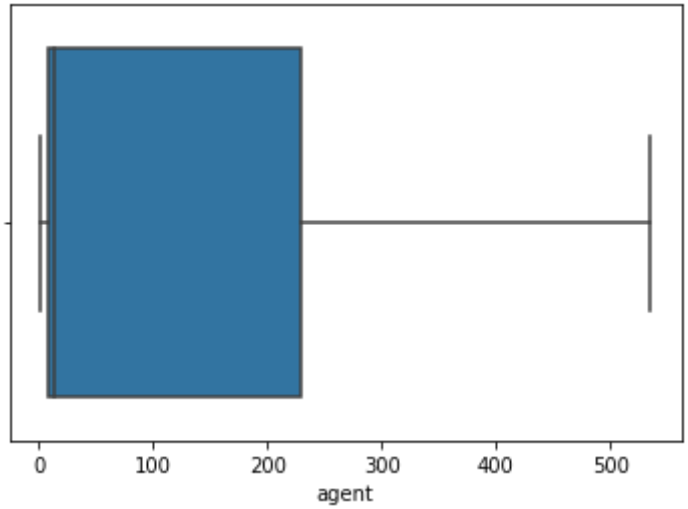
```
df.babies.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e69aa7790>
```



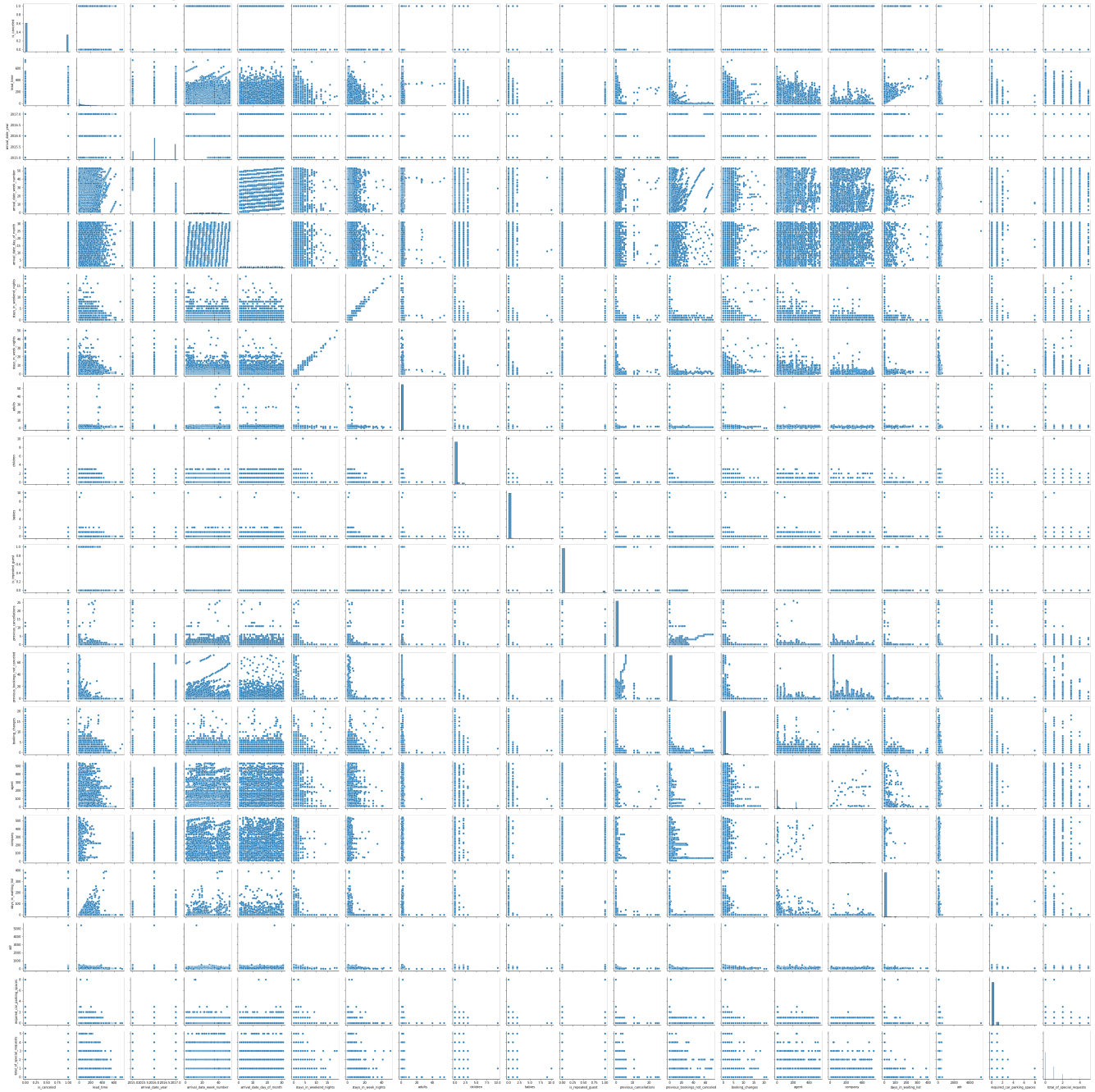

```
sns.boxplot(x="agent",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e66d781d0>
```




```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f1e673e79d0>
```



Part 2 - Select K Best

```
selector = SelectKBest(chi2, k=30)
selector.fit_transform(X_char_dum, Y)

#get columns to create new df with them only
cols = selector.get_support(indices=True)
select_features_df_char = X_char_dum.iloc[:,cols]
```

Creating the Master Feature Set for Model Development

```
X_all = pd.concat([select_features_df_char, select_features_df_num], axis=1, join='inner')
```

Train Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_all, Y, test_size=0.3, random_state=20)
```

```
print("Shape of Training Data",X_train.shape)
print("Shape of Testing Data",X_test.shape)
print("Attrition Rate in Training Data",y_train.mean())
print("Attrition Rate in Testing Data",y_test.mean())
```

```
Shape of Training Data (83573, 34)
Shape of Testing Data (35817, 34)
Attrition Rate in Training Data is_canceled    0.370443
dtype: float64
Attrition Rate in Testing Data is_canceled    0.370355
dtype: float64
```

Building Models

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion = 'gini', random_state = 20)
```

```
from sklearn.model_selection import GridSearchCV
param_dist = {'max_depth': [3,5,6,7], 'min_samples_split': [140,280,420,560,700]}
tree_grid = GridSearchCV(dtree, param_grid = param_dist, cv = 10, n_jobs = -1 )
tree_grid.fit(X_train, y_train)
print('Best Parameters using Grid Search: \n', tree_grid.best_params_)
```

```
Best Parameters using Grid Search:
 {'max_depth': 7, 'min_samples_split': 140}
```

```
dtree = DecisionTreeClassifier(criterion = 'gini', random_state = 0, max_depth = 7, min_samples_split=140)
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=7, min_samples_split=140, random_state=0)
```

```
#building a random forest model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(criterion = 'gini', random_state = 0, max_depth = 7, min_samples_split=140)
rf.fit(X_train,y_train)
```

```
RandomForestClassifier(max_depth=7, min_samples_split=140, random_state=0)
```

```
# Model Evaluation
y_pred_tree = dtree.predict(X_test)
y_pred_rf = rf.predict(X_test)
```

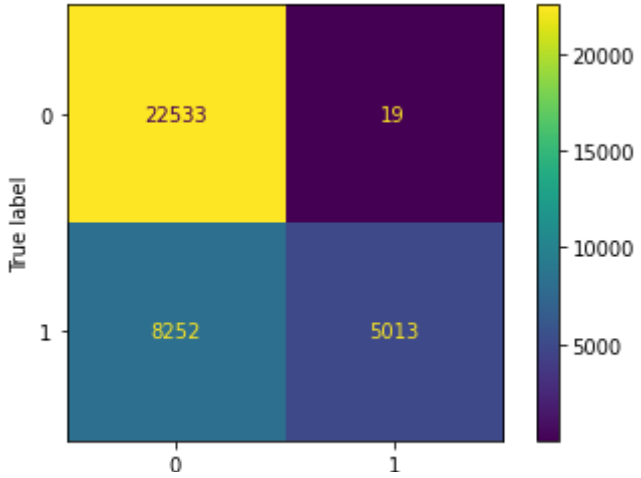
```
# metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

```
print('Decision Tree Metrics ')
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_tree))
print("Precision",metrics.precision_score(y_test,y_pred_tree))
print("Recall",metrics.recall_score(y_test,y_pred_tree))
print("f1_score",metrics.f1_score(y_test,y_pred_tree))
```

```
Decision Tree Metrics
Accuracy: 0.769076137029902
Precision 0.9962241653418124
Recall 0.3779117979645684
f1_score 0.5479586817511067
```

```
metrics.plot_confusion_matrix(dtree, X_test, y_test)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f1e66494290>
```

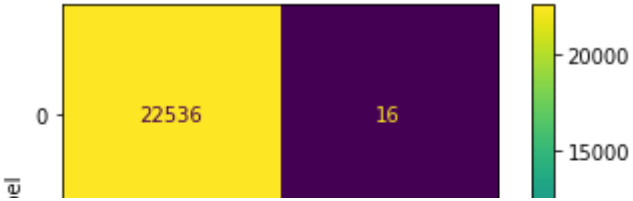



```
print('Random Forest Metrics')
print('Accuracy:', metrics.accuracy_score(y_test, y_pred_rf))
print('Precision:', metrics.precision_score(y_test, y_pred_rf))
print('Recall:', metrics.recall_score(y_test, y_pred_rf))
print('f1_score:', metrics.f1_score(y_test, y_pred_rf))
```

```
Random Forest Metrics
Accuracy: 0.7692994946533769
Precision: 0.9968216130313866
Recall: 0.37828872973991706
f1_score: 0.5484452702333461
```

```
metrics.plot_confusion_matrix(rf, X_test,y_test)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f1e5d4c17d0>
```

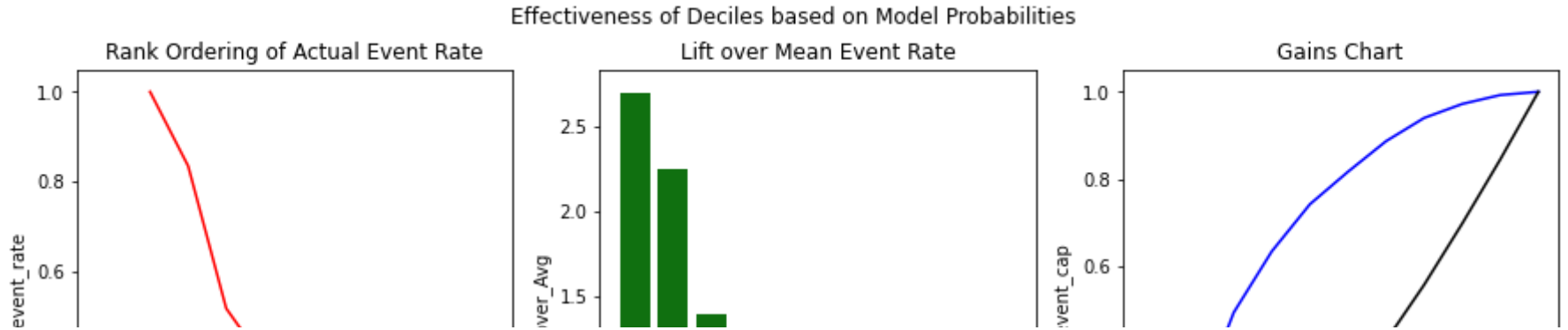


Probability Decile Analysis

```
y_pred_prob = rf.predict_proba(X_all)[:,1] # here we choose col 1 cause it has prob of attrition data col 0 is not attrition
df['y_pred_p'] = pd.DataFrame(y_pred_prob)
df['P_Rank_rf'] = pd.qcut(df['y_pred_p'].rank(method = 'first').values,10,duplicates='drop').codes+1
rank_df_actuals = df.groupby('P_Rank_rf')['is_canceled'].agg(['count', 'mean'])
rank_df_actuals = pd.DataFrame(rank_df_actuals)
rank_df_actuals.rename(columns={'mean': 'Actual_event_rate'}, inplace=True)

sorted_rank_df=rank_df_actuals.sort_values(by='P_Rank_rf',ascending=False)
sorted_rank_df['N_events']=rank_df_actuals['count']*rank_df_actuals['Actual_event_rate']
sorted_rank_df['cum_events']=sorted_rank_df['N_events'].cumsum()
sorted_rank_df['event_cap']=sorted_rank_df['N_events']/max(sorted_rank_df['N_events'].cumsum())
sorted_rank_df['cum_event_cap']=sorted_rank_df['event_cap'].cumsum()

sorted_rank_df['N_non_events']=sorted_rank_df['count']-sorted_rank_df['N_events']
sorted_rank_df['cum_non_events']=sorted_rank_df['N_non_events'].cumsum()
sorted_rank_df['non_event_cap']=sorted_rank_df['N_non_events']/max(sorted_rank_df['N_non_events'].cumsum())
sorted_rank_df['cum_non_event_cap']=sorted_rank_df['non_event_cap'].cumsum()

sorted_rank_df['KS']=round((sorted_rank_df['cum_event_cap']-sorted_rank_df['cum_non_event_cap']),4)

sorted_reindexed=sorted_rank_df.reset_index()
sorted_reindexed['Decile']=sorted_reindexed.index+1
sorted_reindexed['Lift_over_Avg']=sorted_reindexed['Actual_event_rate']/(max(sorted_reindexed['N_events'].cumsum())/max(sorted_reinde
sorted_reindexed
```

	P_Rank_rf	count	Actual_event_rate	N_events	cum_events	event_cap	cum_event_cap	N_non_events	cum_non_events	non_event
0	10	11939	1.000000	11939.0	11939.0	0.269967	0.269967	0.0	0.0	0.000
1	9	11939	0.834827	9967.0	21906.0	0.225375	0.495342	1972.0	1972.0	0.026
2	8	11939	0.518050	6185.0	28091.0	0.139856	0.635198	5754.0	7726.0	0.076
3	7	11939	0.398610	4759.0	32850.0	0.107611	0.742809	7180.0	14906.0	0.095
4	6	11939	0.275400	3288.0	36138.0	0.074349	0.817158	8651.0	23557.0	0.115
5	5	11939	0.259067	3093.0	39231.0	0.069939	0.887098	8846.0	32403.0	0.117
6	4	11939	0.196750	2349.0	41580.0	0.053116	0.940213	9590.0	41993.0	0.127
7	3	11939	0.118435	1414.0	42994.0	0.031974	0.972187	10525.0	52518.0	0.140
8	2	11939	0.075718	904.0	43898.0	0.020441	0.992628	11035.0	63553.0	0.146

```
fig, axes = plt.subplots(1, 3, sharex = True, figsize = (15,5))
fig.suptitle('Effectiveness of Deciles based on Model Probabilities')
axes[0].set_title('Rank Ordering of Actual Event Rate')
axes[1].set_title('Lift over Mean Event Rate')
axes[2].set_title('Gains Chart')
sns.lineplot(ax=axes[0], x = 'Decile', y = 'Actual_event_rate', data = sorted_reindexed, color = 'red')
sns.barplot(ax=axes[1], x = 'Decile', y = 'Lift_over_Avg', data = sorted_reindexed, color = 'green')
sns.lineplot(ax=axes[2], x = 'Decile', y = 'cum_event_cap', data = sorted_reindexed, color = 'blue')
sns.lineplot(ax=axes[2], x = 'Decile', y = 'cum_non_event_cap', data = sorted_reindexed, color = 'black')
plt.show()
```




```
df['Predicted_cancel_Rank'] = np.where(df['P_Rank_rf']<8, 'Bottom7', 'Top3')
df['Predicted_cancel_Rank'].value_counts()
```

```
Bottom7    83573
Top3       35817
Name: Predicted_cancel_Rank, dtype: int64
```

```
df_top3 = df.loc[df['Predicted_cancel_Rank']=='Top3', :]
```

```
df_top3.shape
```

```
(35817, 35)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

Colab paid products - Cancel contracts here