Introduction

First, let's import libraries and the test and train data

In [ ]:
```
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#import data test and train to df
df_test = pd.read_csv('test_data.csv')
df_train = pd.read_csv('train_data.csv')
```

In [ ]:
```
#get shape of data
print(df_train.shape)
print(df_test.shape)
```

```
(29020, 19)
(7255, 19)
```

EDA

In [ ]:
```
#Let's look for missing values
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29020 entries, 0 to 29019
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   BookingID                29020 non-null  int64
 1   LeadTime                 29020 non-null  int64
 2   ArrivalYear              29020 non-null  int64
 3   ArrivalMonth             29020 non-null  int64
 4   ArrivalDate              29020 non-null  int64
 5   NumWeekendNights         29020 non-null  int64
 6   NumWeekNights            29020 non-null  int64
 7   MealPlan                 29020 non-null  object
 8   Parking                  29020 non-null  int64
 9   RoomType                 29020 non-null  object
 10  NumAdults                29020 non-null  int64
 11  NumChildren              29020 non-null  int64
 12  MarketSegment            29020 non-null  object
 13  RepeatedGuest            29020 non-null  int64
 14  NumPrevCancellations     29020 non-null  int64
 15  NumPreviousNonCancelled  29020 non-null  int64
 16  AvgRoomPrice             29020 non-null  float64
 17  SpecialRequests          29020 non-null  int64
 18  BookingStatus            29020 non-null  object
dtypes: float64(1), int64(14), object(4)
memory usage: 4.2+ MB
```

In [ ]:
```
#take a look at train set
df_train.head()
```

Out[ ]:

	BookingID	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeekendNights	Num
0	1	10	2018	3	31	0	
1	2	116	2018	2	28	2	
2	3	11	2018	7	25	1	
3	4	3	2017	9	12	0	
4	5	28	2018	3	7	1	

In [ ]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7255 entries, 0 to 7254
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   BookingID                7255 non-null   int64  
 1   LeadTime                 7255 non-null   int64  
 2   ArrivalYear              7255 non-null   int64  
 3   ArrivalMonth             7255 non-null   int64  
 4   ArrivalDate              7255 non-null   int64  
 5   NumWeekendNights         7255 non-null   int64  
 6   NumWeekNights            7255 non-null   int64  
 7   MealPlan                 7255 non-null   object 
 8   Parking                  7255 non-null   int64  
 9   RoomType                 7255 non-null   object 
 10  MarketSegment            7255 non-null   object 
 11  NumAdults                7255 non-null   int64  
 12  NumChildren              7255 non-null   int64  
 13  RepeatedGuest            7255 non-null   int64  
 14  NumPrevCancellations     7255 non-null   int64  
 15  NumPreviousNonCancelled  7255 non-null   int64  
 16  AvgRoomPrice             7255 non-null   float64
 17  SpecialRequests          7255 non-null   int64  
 18  BookingStatus            0 non-null      float64
dtypes: float64(2), int64(14), object(3)
memory usage: 1.1+ MB
```

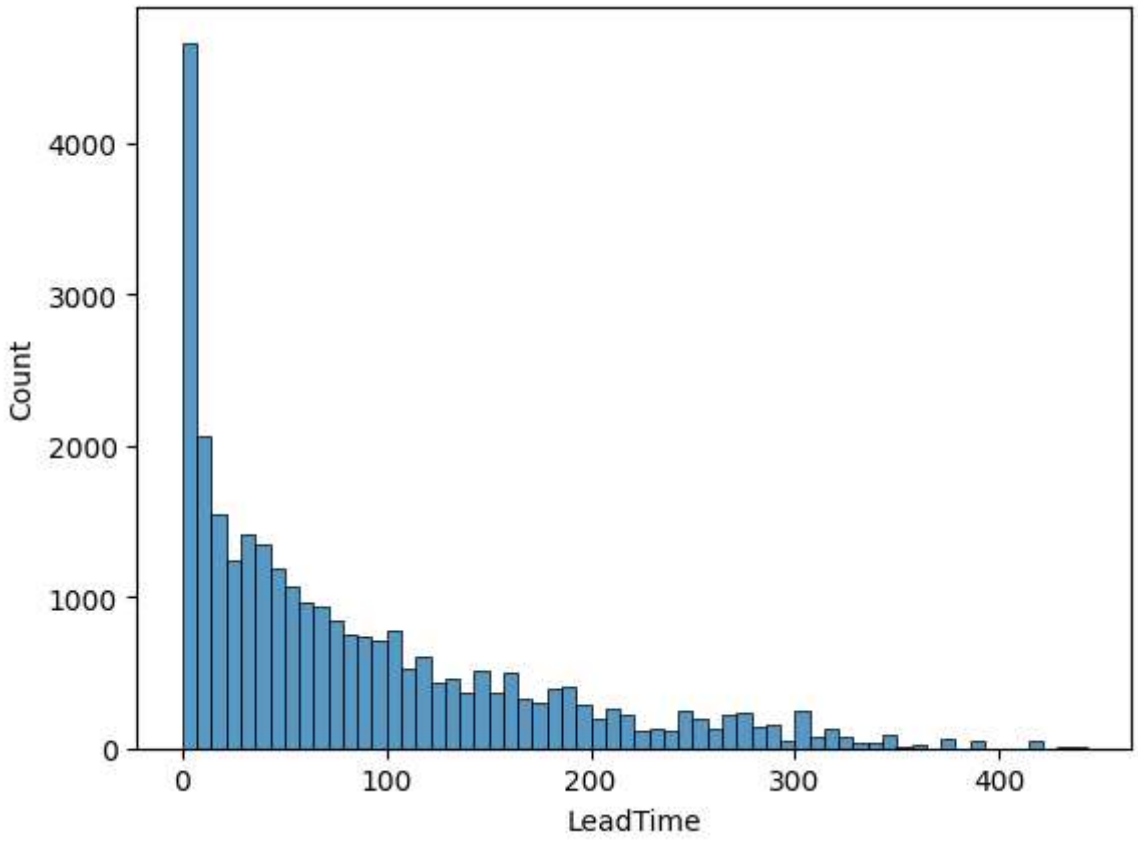
In [ ]:
```
#generate summary statistics
df_train.describe()
```

Out[ ]:

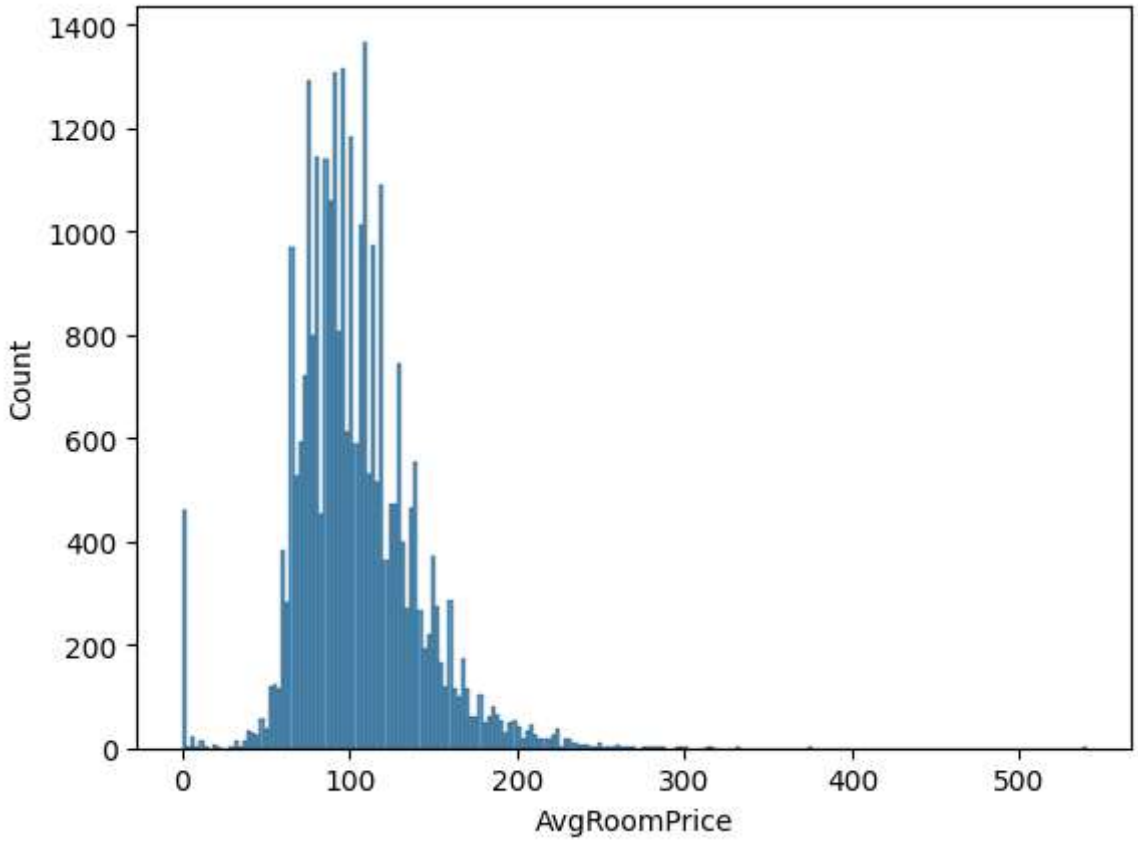
	BookingID	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeeken
count	29020.000000	29020.000000	29020.000000	29020.000000	29020.000000	2902
mean	14510.500000	85.276085	2017.819469	7.417333	15.589318	
std	8377.496742	85.886439	0.384635	3.064481	8.743740	
min	1.000000	0.000000	2017.000000	1.000000	1.000000	
25%	7255.750000	17.000000	2018.000000	5.000000	8.000000	
50%	14510.500000	57.000000	2018.000000	8.000000	16.000000	
75%	21765.250000	127.000000	2018.000000	10.000000	23.000000	
max	29020.000000	443.000000	2018.000000	12.000000	31.000000	

In [ ]:
```
#export the 'MarketSegment' column to a csv
df_train['MarketSegment'].to_csv('MarketSegment.csv', index=False)
```

In [ ]:
```
#histogram of lead time
sns.histplot(df_train['LeadTime'])
plt.show()
```

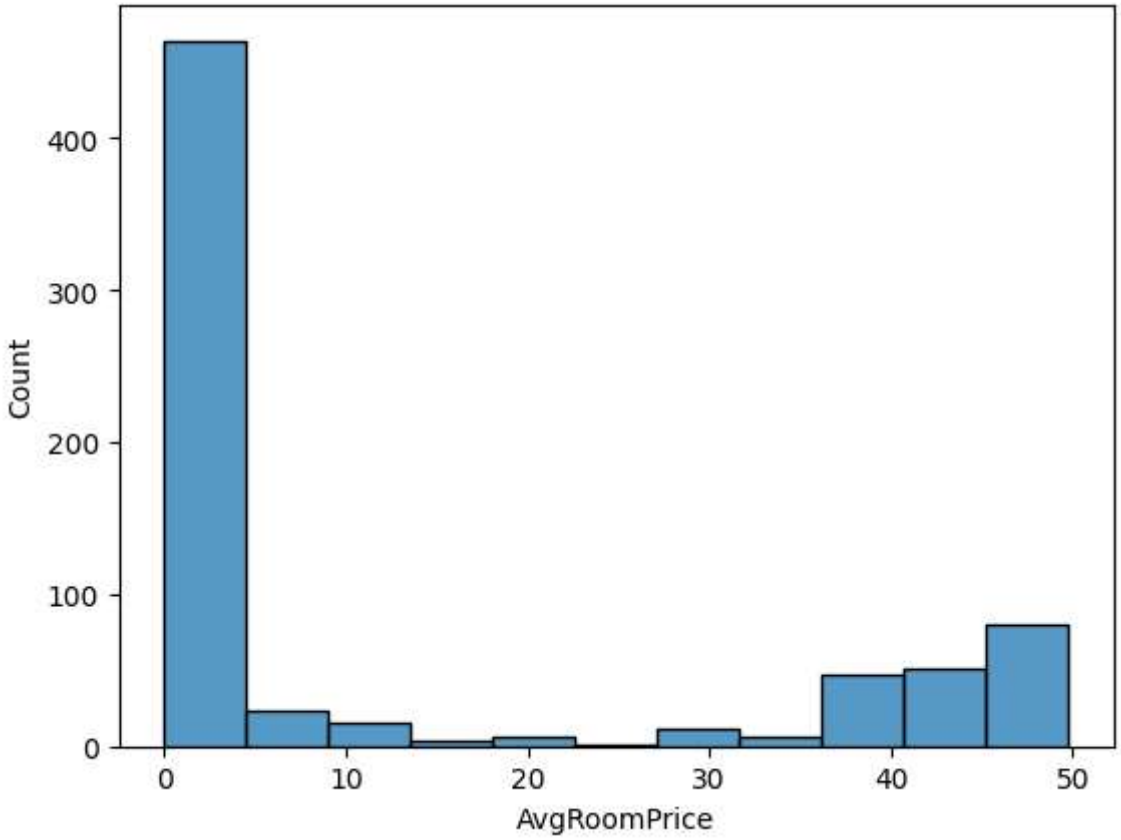


In [ ]:
```
#Let's make a histogram of AvgRoomPrice
sns.histplot(df_train['AvgRoomPrice'])
plt.show()
```

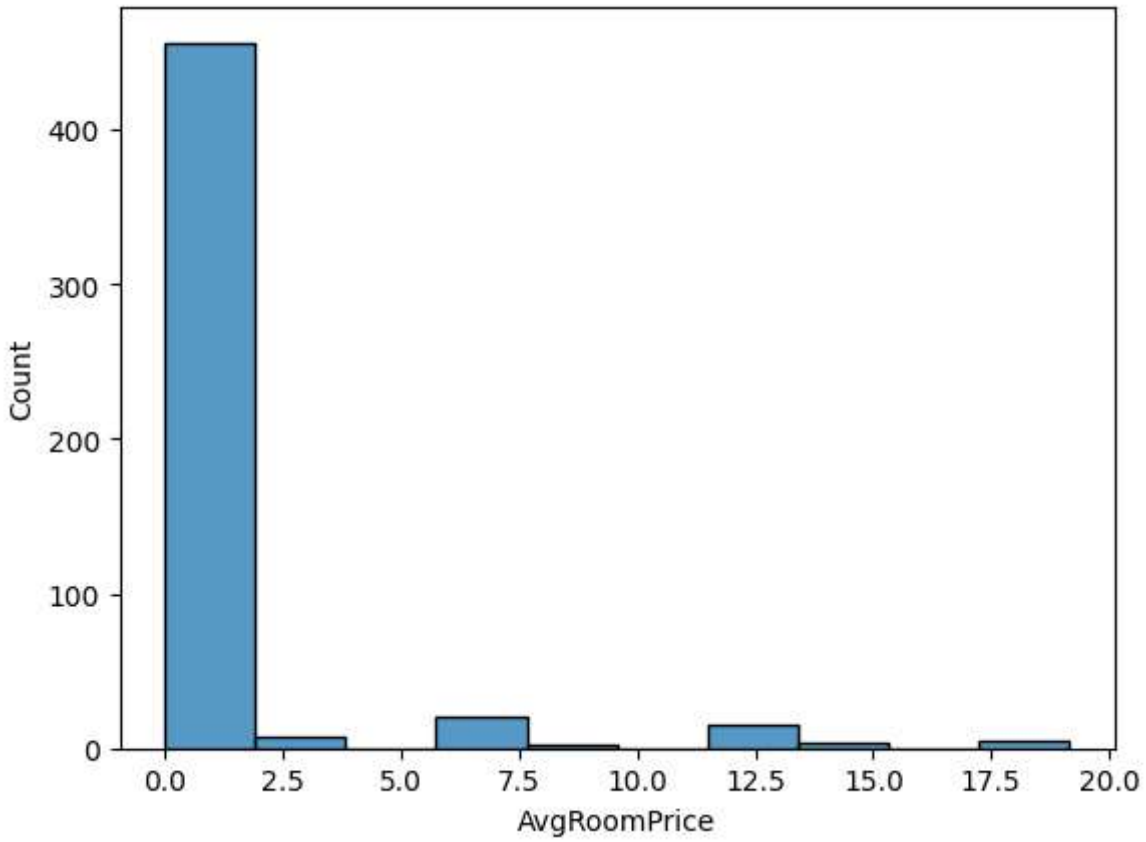


In [ ]:
```
#It looks like there are some prices that are 0. Lets explore this

price_df=df_train[df_train.AvgRoomPrice < 50]
sns.histplot(price_df['AvgRoomPrice'])
plt.show()
```



In [ ]:
```
price_df=df_train[df_train.AvgRoomPrice < 20]
sns.histplot(price_df['AvgRoomPrice'])
plt.show()
```



In [ ]:
```
price_df.head()
```

Out[ ]:

	BookingID	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeekendNights	N
7	8	1	2018	11	9	2	
24	25	4	2018	10	14	0	
95	96	0	2018	7	20	0	
238	239	8	2018	4	23	0	
294	295	5	2017	12	1	0	

In [ ]:
```
df_train.MarketSegment.unique()
```

Out[ ]:
```
array(['Corporate', 'Online', 'Offline', 'Complementary', 'Aviation'],
      dtype=object)
```

It makes sense for the complementary bookings to have a price of 0. However, booked online with a price of 0 does not make sense

In [ ]:
```
price_df=price_df[price_df.MarketSegment != 'Complementary']
price_df.head()
```

Out[ ]:

	BookingID	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeekendNights
95	96	0	2018	7	20	0
1053	1054	60	2017	9	21	1
1246	1247	33	2018	2	20	0
1377	1378	40	2018	1	14	0
1582	1583	147	2018	4	4	1

In [ ]:
```
price_df.shape
```

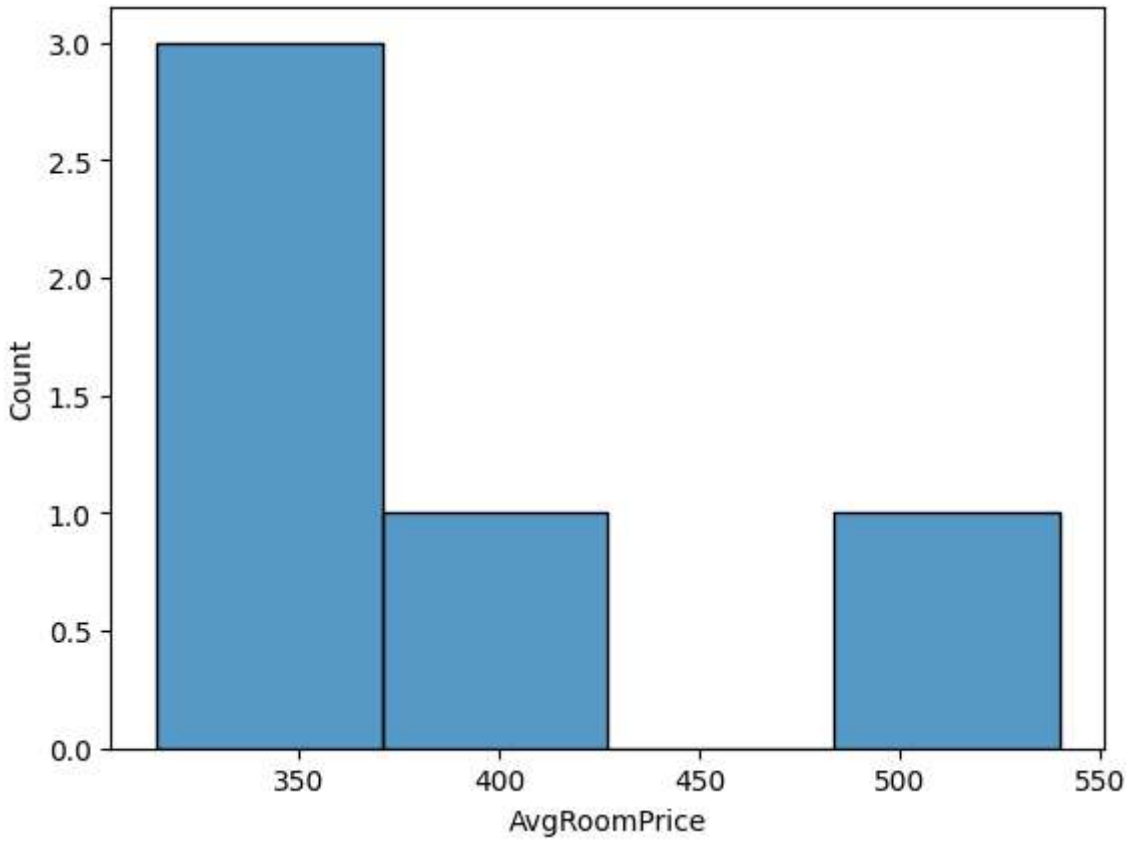
Out[ ]:
```
(209, 19)
```

This data is most likely incorrect and should be removed

In [ ]:
```
df_train=df_train.drop(df_train[(df_train.AvgRoomPrice == 0) & (df_train.MarketSegm
df_test=df_test.drop(df_test[(df_test.AvgRoomPrice == 0) & (df_test.MarketSegment !
```

Now we should also look at super high prices

In [ ]:
```
price_df=df_train[df_train.AvgRoomPrice > 300]
sns.histplot(price_df['AvgRoomPrice'])
plt.show()
```



In [ ]:
```
price_df.head()
```

Out[ ]:

	BookingID	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeekendNights
4229	4230	57	2018	12	30	1
5221	5222	28	2018	6	2	0
18364	18365	6	2018	8	13	0
23385	23386	21	2018	12	30	0
25237	25238	35	2018	3	25	0

These appear fine to keep in the dataset

Now lets look at some of the other columns

In [ ]:
```
for col in df_train.drop(columns={'BookingID','ArrivalDate','LeadTime'}).columns.to
    print(f"{col}: {df_train[col].unique()}")
```

```
ArrivalYear: [2018 2017]
ArrivalMonth: [ 3  2  7  9  6 11 12  4 10  1  8  5]
NumWeekendNights: [0 2 1 4 3 6 5 7]
NumWeekNights: [ 1  2  3  4  5  8  0  7  6 14  9 10 15 11 13 12 17 16]
MealPlan: ['Meal Plan 1' 'Meal Plan 2' 'Not Selected' 'Meal Plan 3']
Parking: [0 1]
RoomType: ['Room_Type 1' 'Room_Type 6' 'Room_Type 4' 'Room_Type 2' 'Room_Type 5'
 'Room_Type 7' 'Room_Type 3']
NumAdults: [1 2 3 0 4]
NumChildren: [ 0  1  2  3  9 10]
MarketSegment: ['Corporate' 'Online' 'Offline' 'Complementary' 'Aviation']
RepeatedGuest: [0 1]
NumPrevCancellations: [ 0  2  3  1 11  5 13  4  6]
NumPreviousNonCancelled: [ 0  7  2 24 17  1  3  5 28 12 10  9 48  4  6 51 32 16 43 1
4 13  8 56 30
 20 40 34 38 42 23 27 26 22 25 11 15 55 44 18 50 45 29 31 41 35 21 39 37
 49 58 19 54 53 57 52 33]
AvgRoomPrice: [ 95.    61.   129.75 ... 246.25  64.72  78.9 ]
SpecialRequests: [0 1 3 2 4 5]
BookingStatus: ['Canceled' 'Not_Canceled']
```

In [ ]:
```
for col in df_train.drop(columns={'BookingID','ArrivalDate','LeadTime'}).columns.to
    print(f"{col}: {df_train[col].value_counts(normalize=True)}")
```

```
ArrivalYear: 2018    0.821505
2017    0.178495
Name: ArrivalYear, dtype: float64
ArrivalMonth: 10    0.145921
9     0.128283
8     0.105690
6     0.088121
7     0.081884
11    0.081606
12    0.081399
4     0.074815
5     0.072077
3     0.064696
2     0.047682
1     0.027826
Name: ArrivalMonth, dtype: float64
NumWeekendNights: 0    0.462679
1    0.276249
2    0.251265
3    0.004643
4    0.003396
5    0.001144
6    0.000589
7    0.000035
Name: NumWeekendNights, dtype: float64
NumWeekNights: 2     0.315268
1     0.260032
3     0.217548
4     0.083235
0     0.064939
5     0.044355
6     0.005198
7     0.003361
10    0.001733
8     0.001698
9     0.000970
11    0.000520
12    0.000277
15    0.000277
14    0.000243
13    0.000173
17    0.000104
16    0.000069
Name: NumWeekNights, dtype: float64
MealPlan: Meal Plan 1     0.767586
Not Selected    0.142179
Meal Plan 2     0.090062
Meal Plan 3     0.000173
Name: MealPlan, dtype: float64
Parking: 0    0.969055
1    0.030945
Name: Parking, dtype: float64
RoomType: Room_Type 1    0.773650
Room_Type 4    0.168757
Room_Type 6    0.026474
Room_Type 2    0.019267
```

```
Room_Type 5    0.007381
Room_Type 7    0.004332
Room_Type 3    0.000139
Name: RoomType, dtype: float64
NumAdults: 2    0.722018
1    0.209093
3    0.064765
0    0.003639
4    0.000485
Name: NumAdults, dtype: float64
NumChildren: 0     0.926017
1     0.044598
2     0.028727
3     0.000554
9     0.000069
10    0.000035
Name: NumChildren, dtype: float64
MarketSegment: Online           0.638679
Offline          0.291496
Corporate        0.055305
Complementary    0.011054
Aviation         0.003465
Name: MarketSegment, dtype: float64
RepeatedGuest: 0    0.974357
1    0.025643
Name: RepeatedGuest, dtype: float64
NumPrevCancellations: 0     0.990471
1     0.005336
2     0.001490
3     0.001144
11    0.000728
5     0.000381
4     0.000277
13    0.000139
6     0.000035
Name: NumPrevCancellations, dtype: float64
NumPreviousNonCancelled: 0     0.977511
1     0.006099
2     0.003015
3     0.002287
5     0.001802
4     0.001698
6     0.001040
7     0.000693
8     0.000624
9     0.000589
10    0.000485
11    0.000381
12    0.000347
14    0.000277
15    0.000243
13    0.000208
17    0.000208
22    0.000208
20    0.000208
18    0.000139
```

```
19    0.000139
16    0.000139
21    0.000104
24    0.000104
26    0.000069
29    0.000069
48    0.000069
44    0.000069
28    0.000069
25    0.000069
32    0.000069
27    0.000069
23    0.000069
30    0.000069
53    0.000035
52    0.000035
58    0.000035
49    0.000035
54    0.000035
57    0.000035
37    0.000035
39    0.000035
35    0.000035
42    0.000035
41    0.000035
31    0.000035
45    0.000035
50    0.000035
55    0.000035
38    0.000035
34    0.000035
40    0.000035
56    0.000035
43    0.000035
51    0.000035
33    0.000035
Name: NumPreviousNonCancelled, dtype: float64
AvgRoomPrice: 65.00     0.024118
75.00     0.022628
90.00     0.019405
95.00     0.018643
115.00    0.018539
            ...
79.69     0.000035
90.86     0.000035
91.33     0.000035
155.13    0.000035
78.90     0.000035
Name: AvgRoomPrice, Length: 3522, dtype: float64
SpecialRequests: 0    0.544251
1    0.313916
2    0.120590
3    0.018990
4    0.002044
5    0.000208
Name: SpecialRequests, dtype: float64
```

```
BookingStatus: Not_Canceled    0.670247
Canceled        0.329753
Name: BookingStatus, dtype: float64
```

There does not appear to be anything weird with the columns and their value distributions

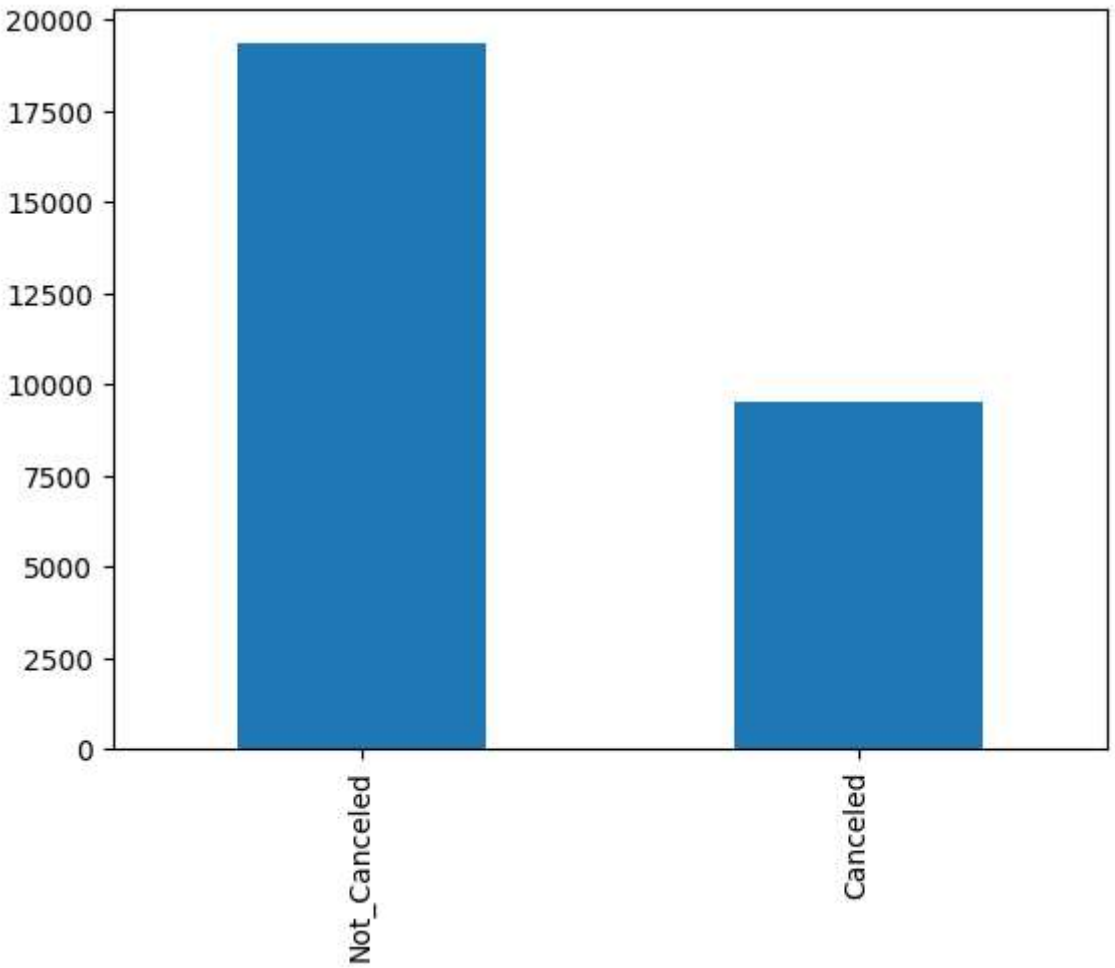
Now lets look at class imbalance

In [ ]:
```
df_train.BookingStatus.value_counts(normalize=True)
```

Out[ ]:
```
Not_Canceled    0.670247
Canceled        0.329753
Name: BookingStatus, dtype: float64
```

In [ ]:
```
df_train['BookingStatus'].value_counts().plot(kind='bar')
```

Out[ ]: <Axes: >



we have imbalanced classes. We will need to take this into account when training models

Data Pre Processing

We will want to convert all of the columns into some form of numeric representation

In [ ]:
```
#lets convert the booking status to a binary value
df_train['isCanceled'] = df_train['BookingStatus'].map({'Canceled': 1, 'Not_Cancele
```

In [ ]:
```
#let's make dummy variables for the room type
room_type_dummies = pd.get_dummies(df_train['RoomType'], prefix='RoomType')
#now for meal plan
meal_plan_dummies = pd.get_dummies(df_train['MealPlan'], prefix='MealPlan')

market_segment_dummies = pd.get_dummies(df_train['MarketSegment'], prefix='MarketSe
```

In [ ]:
```
#create a train df with the dummy variables, and the normalized values, and the bin
df_train = pd.concat([df_train, room_type_dummies, meal_plan_dummies,market_segment
```

In [ ]:
```
#let's look at the columns
df_train.columns
```

Out[ ]:
```
Index(['BookingID', 'LeadTime', 'ArrivalYear', 'ArrivalMonth', 'ArrivalDate',
       'NumWeekendNights', 'NumWeekNights', 'MealPlan', 'Parking', 'RoomType',
       'NumAdults', 'NumChildren', 'MarketSegment', 'RepeatedGuest',
       'NumPrevCancellations', 'NumPreviousNonCancelled', 'AvgRoomPrice',
       'SpecialRequests', 'BookingStatus', 'isCanceled',
       'RoomType_Room_Type 1', 'RoomType_Room_Type 2', 'RoomType_Room_Type 3',
       'RoomType_Room_Type 4', 'RoomType_Room_Type 5', 'RoomType_Room_Type 6',
       'RoomType_Room_Type 7', 'MealPlan_Meal Plan 1', 'MealPlan_Meal Plan 2',
       'MealPlan_Meal Plan 3', 'MealPlan_Not Selected',
       'MarketSegment_Aviation', 'MarketSegment_Complementary',
       'MarketSegment_Corporate', 'MarketSegment_Offline',
       'MarketSegment_Online'],
      dtype='object')
```

In [ ]:
```
#let's remove bookingID, Room Type vals, hasMealPlan, MealPlanVals, and BookingStat
df_train = df_train.drop(['BookingID','BookingStatus', 'RoomType', 'MealPlan','Mark
```

In [ ]:
```
#now lets create a preprocess pipeline to do the same for the testing dataset

def preprocess(df):
    df['isCanceled'] = df['BookingStatus'].map({'Canceled': 1, 'Not_Canceled': 0})
    #let's make dummy variables for the room type
    room_type_dummies = pd.get_dummies(df['RoomType'], prefix='RoomType')
#now for meal plan
    meal_plan_dummies = pd.get_dummies(df['MealPlan'], prefix='MealPlan')

    market_segment_dummies = pd.get_dummies(df['MarketSegment'], prefix='MarketSegm

    df = pd.concat([df, room_type_dummies, meal_plan_dummies,market_segment_dummies

    df = df.drop(['BookingID','BookingStatus', 'RoomType', 'MealPlan','MarketSegmen

    return df


df_test=preprocess(df_test)
```

Feature Engineering

In [ ]:
```
df_train.head()
```

Out[ ]:

	LeadTime	ArrivalYear	ArrivalMonth	ArrivalDate	NumWeekendNights	NumWeekNights
0	10	2018	3	31	0	1
1	116	2018	2	28	2	1
2	11	2018	7	25	1	2
3	3	2017	9	12	0	1
4	28	2018	3	7	1	3

5 rows × 31 columns

We can make new columns `number of nights stayed` and `total guests`

In [ ]:
```
def add_features(df):
    df['Total_Nights'] = df['NumWeekNights'] + df['NumWeekendNights']
    df['Total_Customers'] = df['NumChildren'] + df['NumAdults']
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Model Exploration

In [ ]:
```
# let's preform linear regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
#we already have the train df, so we will use that
#we will use the isCanceled column as the target
#we will use the rest of the columns as the features
X_train = df_model.drop('isCanceled', axis=1)
y_train = df_model['isCanceled']

#split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2

#instantiate the model
lm = LinearRegression()

#fit the model
lm.fit(X_train, y_train)

#predict
y_pred = lm.predict(X_test)

#evaluate accuracy of prediction, if the prediction is within 0.5 of the actual val
#round y_pred to 0 or 1
import numpy as np
y_pred_round = np.round(y_pred)
#compare y_pred_round to y_test

#calculate accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred_round)
```

Out[ ]:
```
0.8053066850447967
```

In [ ]:
```
#now let's preform pca on the data
from sklearn.decomposition import PCA

#instantiate the model
pca = PCA(n_components=2)

#fit the model
pca.fit(X_train)

#transform the data
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)

#instantiate the model
lm_pca = LinearRegression()

#fit the model
lm_pca.fit(X_train_pca, y_train)

#predict
y_pred_pca = lm_pca.predict(X_test_pca)

#round y_pred to 0 or 1

y_pred_round_pca = np.round(y_pred_pca)
```

```
#compare y_pred_round to y_test

#calculate accuracy

accuracy_score(y_test, y_pred_round_pca)
```

Out[ ]:
```
0.762749827705031
```

In [ ]:
```
df_train.columns
```

Out[ ]:
```
Index(['BookingID', 'LeadTime', 'ArrivalYear', 'ArrivalMonth', 'ArrivalDate',
       'NumWeekendNights', 'NumWeekNights', 'MealPlan', 'Parking', 'RoomType',
       'NumAdults', 'NumChildren', 'MarketSegment', 'RepeatedGuest',
       'NumPrevCancellations', 'NumPreviousNonCancelled', 'AvgRoomPrice',
       'SpecialRequests', 'BookingStatus', 'LeadTimeNormal',
       'AvgRoomPriceNormal', 'isCanceled', 'hasMealPlan', 'MealPlanVals',
       'RoomTypeVals'],
      dtype='object')
```

In [ ]:
```
#create a date column from the year and month and date columns
df_train['Date'] = df_train['ArrivalYear'].astype(str) + '-' + df_train['ArrivalMon
```

In [ ]:
```
print(df_train['Date'].head())
```

```
0    2018-3-31
1    2018-2-28
2    2018-7-25
3    2017-9-12
4     2018-3-7
Name: Date, dtype: object
```

PCA