School of Information Studies
Syracuse University

Hotel Booking Cancellation Prediction

IST 718: Big Data Analytics
SPRING 2020
Prof. Humayun Khan

Submitted By:
Arsheen Barnagarwala
Karan Puran Ashar
Dhwani Rekhang Gandhi
Preethi Ramesh

Introduction

The rate of cancellation for bookings in the hospitality industry is quite high in the competitive market offering no deposit bookings. Once the booking has been cancelled, there is almost nothing to be done at the end of hotel. This kind of setting creates discomfort and monetary losses for many hotels and creates a demand to take prior precautions for high number of cancellations. Therefore, predicting bookings that can be cancelled and preventing these cancellations will create value for hospitality industry [1].

In this project, we will try to explain how future cancelled hotel bookings can be predicted in advance with the help of machine learning methods. Also, what measures and steps can be implemented for reducing their impact on industry's revenue.

Research Problem Statement

In this proposed project, we will be predicting whether a booking made in a hotel can be cancelled in future or not. For this, we have developed models that will identify and flag bookings with high cancellation probability by understanding the trends and features associated with it. Thus, hotels can act on these booking to contain the associated revenue losses.

This kind of prediction will help in producing better net demand forecasts, improve overbooking/cancellation policies, and have more assertive pricing and inventory allocation strategies. This analysis can also be used by hotels to identify their loyal customers who never cancel their booking and can provide loyalty benefits.

Data and it's source

For this project, we are using "Hotel Booking Demand" from Kaggle. This data set contains booking information for a city hotel and a resort hotel and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces, among other things. All personally identifying information has been removed from the data.

The data is originally from the article 'Hotel Booking Demand Datasets', written by Nuno Antonio, Ana Almeida, and Luis Nunes for Data in Brief, Volume 22, February 2019.

Link: https://www.kaggle.com/jessemostipak/hotel-booking-demand

Load Data:

In [2]:
```
hotel_bookings=pd.read_csv("hotel_bookings.csv")

# Printing first 10 records of hotel_bookings
hotel_bookings.head(10)
```

Out[2]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month
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	Resort Hotel	0	14	2015	July	27	1
6	Resort Hotel	0	0	2015	July	27	1

In [20]:
```
# Number of rows:
print(hotel_bookings['hotel'].count())

# Number of Columns:
print(len(hotel_bookings.columns))
```

```
119210
37
```

Initial Strategy

We are doing this project on Spark as this would help us to get more in-depth knowledge of Big Data in industry. For this project, we will be using supervised learning with ensemble technique for training and testing against the dataset. We will classify each booking using different classifiers such as Decision Tree, Random Forest and Gradient Boosted Tree which are ensemble learning models, we propose to attain the best accuracy model for predicting whether a hotel booking can be cancelled in future or not.

Data cleaning & Data Preparation

1. **Missing Values**:
   - The children and country columns had missing values.

```
In [4]: # Number of Null Values in data:
        hotel_bookings.isna().sum()

Out[4]: hotel                           0
        is_canceled                     0
        lead_time                       0
        arrival_date_year               0
        arrival_date_month              0
        arrival_date_week_number        0
        arrival_date_day_of_month       0
        stays_in_weekend_nights         0
        stays_in_week_nights            0
        adults                          0
        children                        4
        babies                          0
        meal                            0
        country                       488
        market_segment                  0
        distribution_channel            0
        is_repeated_guest               0
        previous_cancellations          0
        previous_bookings_not_canceled  0
        reserved_room_type              0
        assigned_room_type              0
        booking_changes                 0
        deposit_type                    0
```

   - The missing ‘children’ values were filled with ‘0’ since the average count of children in the entire dataset was very close to ‘0’
   - The missing ‘country’ values were filled with a variable called ‘unknown’

```
In [5]: # Replacing null values in 'children' column with '0' because average of the same column is nearly '0'
        hotel_bookings['children'].fillna(0,inplace=True)

        # Replacing null values in 'country' column with 'Unknown' text
        hotel_bookings['country'].fillna('Unknown',inplace=True)
```

   - Some rows contain entries with 0 adults, 0 children and 0 babies, these records were dropped as there were no guests.

```
In [9]: # Number of rows with children, adults, babies = 0 , in short rows with no guests
        hotel_bookings[(hotel_bookings['children']==0) & (hotel_bookings['adults']==0) & (hotel_bookings['babies']==0)]['hotel'].count()

Out[9]: 180

In [10]: # Removing rows with children, adults, babies = 0 , in short rows with no guests
         hotel_bookings=hotel_bookings.drop(hotel_bookings[(hotel_bookings['children']==0) & (hotel_bookings['adults']==0) & (hotel_booki
```

2. **Outlier Detection**:
   - We determined the lower and upper points of each column
   - If there is point between the lower point and the upper point or equal, we do not apply any filtering else, we have removed observations larger and lower than the upper point and lower point of the observations from the dataset.

```
Lower point: -195.5     upper point: 376.5      lead_time
Lower point: -3.0       upper point: 5.0        stays_in_weekend_nights
Lower point: -2.0       upper point: 6.0        stays_in_week_nights
Lower point: 2.0        upper point: 2.0        adults
Lower point: 0.0        upper point: 0.0        children
Lower point: 0.0        upper point: 0.0        babies
Lower point: -11.5      upper point: 208.5      adr
```

3. **Managing Data Types**:
   - Changed datatypes to category to reduce the memory used by the data-frame

In [6]:
```
# Converting data types to category to reduce data frame size

hotel_bookings['hotel']=hotel_bookings['hotel'].astype('category')
hotel_bookings['is_canceled']=hotel_bookings['is_canceled'].astype('category')
hotel_bookings['meal']=hotel_bookings['meal'].astype('category')
hotel_bookings['country']=hotel_bookings['country'].astype('category')
hotel_bookings['market_segment']=hotel_bookings['market_segment'].astype('category')
hotel_bookings['distribution_channel']=hotel_bookings['distribution_channel'].astype('category')
hotel_bookings['is_repeated_guest']=hotel_bookings['is_repeated_guest'].astype('category')
hotel_bookings['reserved_room_type']=hotel_bookings['reserved_room_type'].astype('category')
hotel_bookings['deposit_type']=hotel_bookings['deposit_type'].astype('category')
hotel_bookings['customer_type']=hotel_bookings['customer_type'].astype('category')
hotel_bookings['required_car_parking_spaces']=hotel_bookings['required_car_parking_spaces'].astype('category')
hotel_bookings['total_of_special_requests']=hotel_bookings['total_of_special_requests'].astype('category')
hotel_bookings['reservation_status']=hotel_bookings['reservation_status'].astype('category')
hotel_bookings['reservation_status_date']=hotel_bookings['reservation_status_date'].astype('datetime64[ns]')
hotel_bookings['children']=hotel_bookings['children'].astype('int64')
hotel_bookings['stays_in_weekend_nights']=hotel_bookings['stays_in_weekend_nights'].astype('category')
hotel_bookings['stays_in_week_nights']=hotel_bookings['stays_in_week_nights'].astype('category')
hotel_bookings['assigned_room_type']=hotel_bookings['assigned_room_type'].astype('category')
hotel_bookings['arrival_date_month']=hotel_bookings['arrival_date_month'].astype('category')
```

4. **Combining Columns**:
   - Converted Day, Month, Year into a single arrival date

In [12]:
```
# Converting Day, Month, Year into a single arrival date for ease of analysis

month_number={'January':1,
              'February':2,
              'March':3,
              'April':4,
              'May':5,
              'June':6,
              'July':7,
              'August':8,
              'September':9,
              'October':10,
              'November':11,
              'December':12}

hotel_bookings['arrival_date_month_number']=hotel_bookings['arrival_date_month'].apply(lambda x:month_number[x])
hotel_bookings['arrival_date_month_number']=hotel_bookings['arrival_date_month_number'].astype('category')

hotel_bookings['date_of_arrival']=pd.to_datetime(hotel_bookings['arrival_date_year'].astype(str)+hotel_bookings['arrival_date_mo
```

5. **Introducing New Columns:**
   - Created a column ‘total_night_stays’ for calculating the number of nights stayed for each booking including weekends and weekdays.

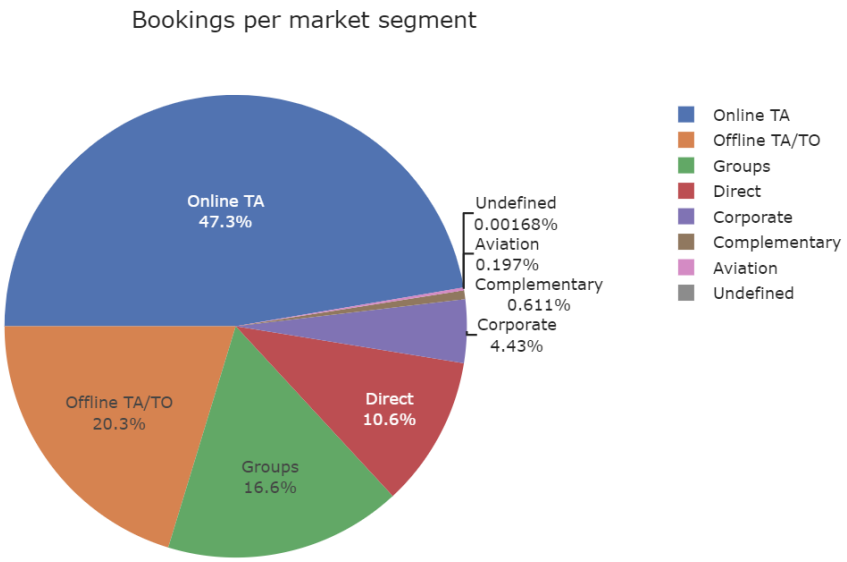
In [16]:
```
# Creating a column for 'total_nigt_stays' for number of night stays
hotel_bookings['total_night_stays']=hotel_bookings['stays_in_week_nights'].astype(int)+hotel_bookings['stays_in_weekend_nights']
```

- Creating a new column 'One_day_stay' for all those bookings which have check-in and check-out status but '0 (zero)' night stays.

```
In [17]: # Creating a categorical column for 'One_day_stay' signifying 'Yes' or 'No'
hotel_bookings.loc[(hotel_bookings['reservation_status']=='Check-Out')&(hotel_bookings['total_night_stays']==0),'One_day_stay']=
hotel_bookings['One_day_stay']=hotel_bookings['One_day_stay'].astype('category')
```

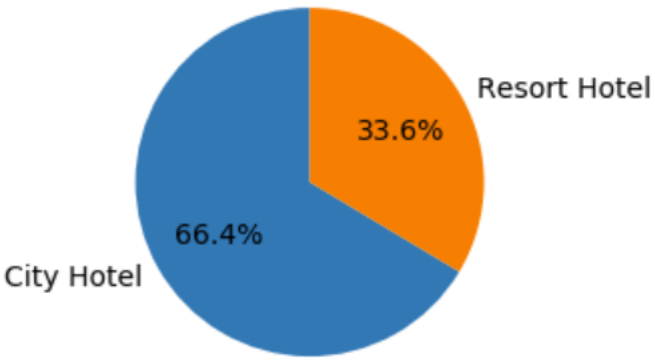
Data Exploration

To start with data exploration, we analysed the total number of bookings by different segments.



Inference: The above graph shows that most bookings in both hotels are made through online booking segment, which constitutes to around 47% (approx.)

Now, let see the distribution of booking by hotels



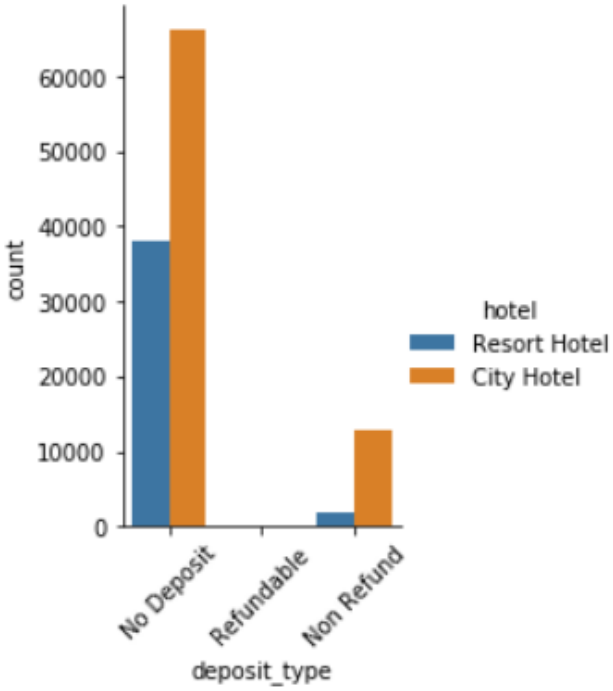
Inference: The above graph shows that maximum percentage of bookings are made by City hotel, which constitutes around 66% (approx.)

We also calculated the percentage of bookings cancelled and by each hotel:

```
Total bookings canceled: 44,199 (37 %)
Resort hotel bookings canceled: 11,120 (28 %)
City hotel bookings canceled: 33,079 (42 %)
```

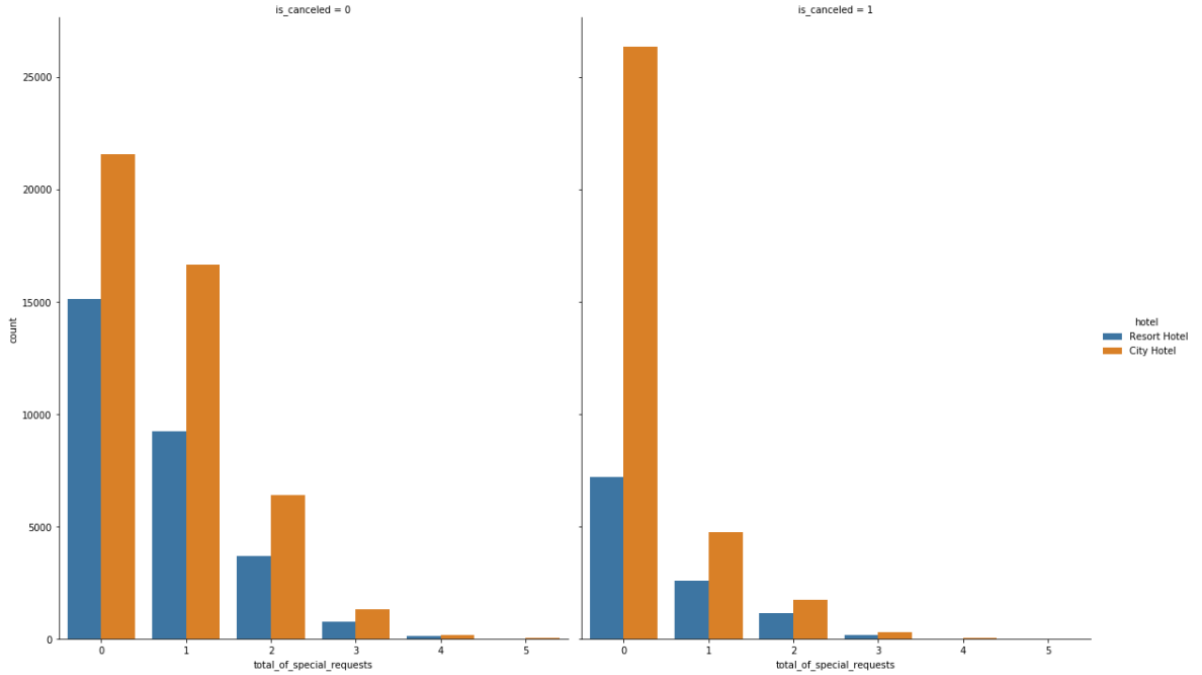
Inference: The above data shows that City hotel experiences maximum percentage of cancellations among both the hotels.

Let see how the deposit ('deposit_type') made before booking affects the booking cancellation ('is_canceled').



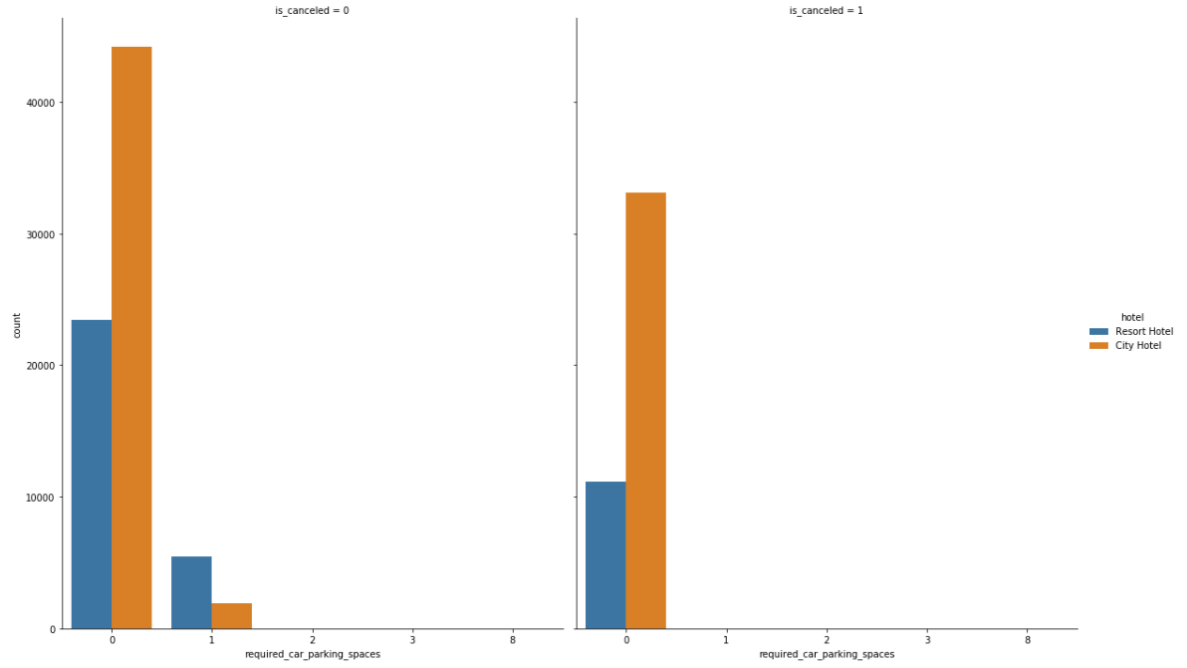
Inference: The above graph shows that maximum number of bookings are made with no deposit, that's why they are getting cancelled.

Now, let us analyse how special request made by guests affects the booking cancellation.



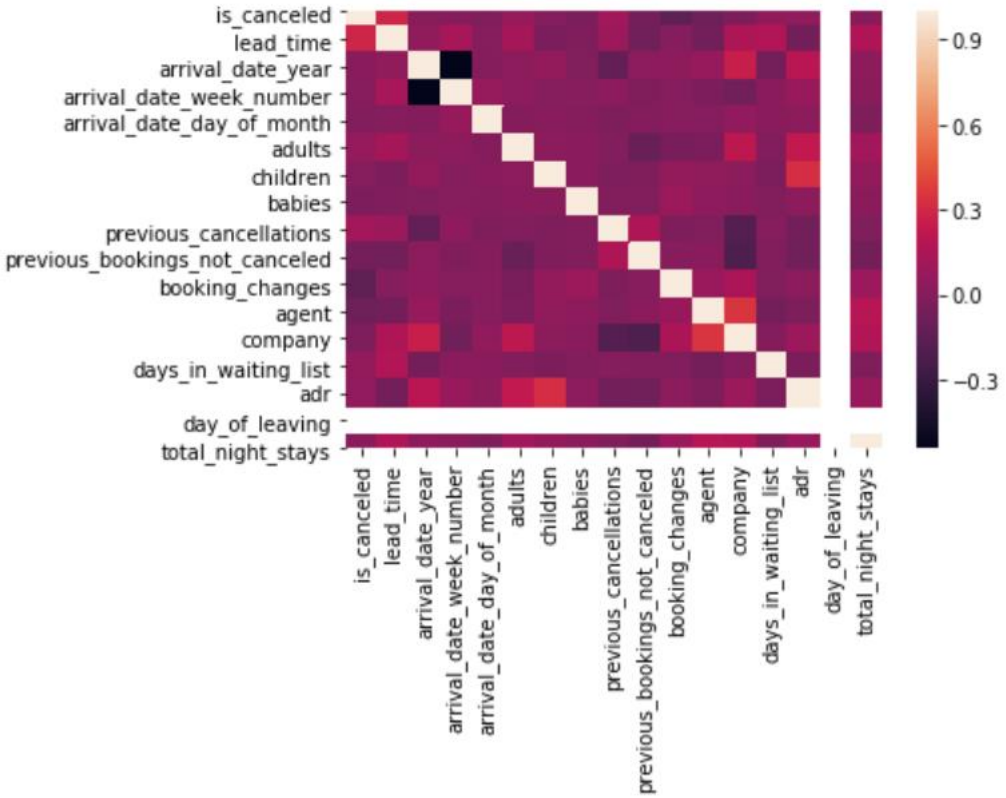
Inference: The above graph shows that the bookings which had no special request are more likely to cancel compared to the bookings which have special requests.

To analyse the above inference further we plotted a graph for special request for car parking space and analysed how it affects the booking cancellation.



Inference: The above graph shows that people who have requested for car parking space never cancelled the booking whereas the people who have not requested parking space mostly cancelled the booking.

In the last of data-exploration we tried to plot the correlation heat map of all the attributes to find the best features for the further analysis.



We did not get a clear picture of the correlation form the above heatmap. So we tried to employ another method for feature engineering which is Cramer's V test.

Feature Selection

Our target variable 'is_canceled' is a categorical variable. Majority of the features in our dataset are categorical variables. To find relation between two categorical variables we conducted the Cramer's V test. The input of the Cramer's V test is the statistic component of the chi square test. Below is the formula for the Cramer's V test:

$$\phi_c = \sqrt{\frac{\chi^2}{N(k-1)}}$$

- Where-
- χ^2 - is the Pearson chi-square statistic component
- N - is the sample size involved in the test
- k - is the lesser number of categories of either variable

```
In [26]: cramers={}
```

```
In [27]: indexer = StringIndexer(inputCol="adults", outputCol="adultsIndex")
assembler=VectorAssembler(inputCols=['adultsIndex'],outputCol='adultsVector')
pipeline = Pipeline(stages=[indexer,assembler])
df1=pipeline.fit(df1).transform(df1)
```

```
In [28]: r=ChiSquareTest.test(df1,'adultsVector','is_canceled').head()
print("pValues: " + str(r.pValues))
print("degreesOfFreedom: " + str(r.degreesOfFreedom))
print("statistics: " + str(r.statistics))
cramers['adults']=float(np.sqrt(r.statistics/df1.count()))
```

The above code shows how the Cramer's V score was calculated. The score for each feature was stored in a dictionary called 'cramers'. The top features in the dictionary were selected for modelling.

```
[191]: cramers

[191]: {'deposit_type': 0.48153033481822316,
 'agent': 0.3859997080414959,
 'country': 0.3619829323351092,
 'previous_cancellation': 0.27871297605202555,
 'market_segment': 0.2670060369757948,
 'total_of_special_requests': 0.26558603528304653,
 'required_car_parking_spaces': 0.19760625952288097,
 'booking_changes': 0.1873933838724716,
 'distribution_channel': 0.1771672495133925,
 'days_in_waiting_list': 0.16316172060506876,
 'total_night_stays': 0.15124257573873204,
 'company': 0.14073809279963861,
 'customer_type': 0.13661731757126358,
 'stays_in_week_nights': 0.11910310330907341,
 'arrival_date_week_number': 0.09707392724299048,
 'adults': 0.08616012240949061,
 'is_repeated_guest': 0.08374544952596244,
 'reserved_room_type': 0.07276874876259916,
 'arrival_date_month': 0.06988568846127591,
 'meal': 0.05058364957307752,
 'babies': 0.03444196988114078,
 'stays_in_weekend_nights': 0.03072712475479656,
 'children': 0.028777570267179396}
```

Prediction Models & Pipeline

After feature engineering, we concluded that the most correlated columns with our target variable are all categorical columns. So, we made a pipeline by defining three different stages:

- StringIndexer: Converts categorical values to categorical indices.
- OneHotEncoder: It maps categorical indices to a column of binary vector, with at most a single one-value per row.
- VectorAssembler: It combines raw features and features generated by different feature transformers into single feature vector.

In [92]:
```
# Defining all categorical columns
categoricalColumns = ["deposit_type","agent","country","previous_cancellations" ,"market_segment", "total_of_special_requests"]

# Defining stages of pipeline for categorical columns
stages = []
for categoricalCol in categoricalColumns:
    stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol + "Index")
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()],
                            outputCols=[categoricalCol + "classEncoder"])
    stages += [stringIndexer, encoder]

# Indexing target variable column 'is_canceled'
label_stringIndx = StringIndexer(inputCol="is_canceled", outputCol="label")
stages += [label_stringIndx]

# Vectorizing all the categorical columns
VectorAssembler=VectorAssembler(inputCols=['deposit_typeclassEncoder','agentclassEncoder',
                                           'countryclassEncoder','previous_cancellationsclassEncoder',
                                           'market_segmentclassEncoder','total_of_special_requestsclassEncoder'],
                                outputCol='features')
stages+=[VectorAssembler]
```

Then, fitting the pipeline to the cleaned data-frame 'Hotel_Bookings'

In [93]:
```
# Fitting all the above defined stages to the dataframe
partial_Pipeline = Pipeline().setStages(stages)
pipeline_Model = partial_Pipeline.fit(Hotel_Bookings)
prepped_DF = pipeline_Model.transform(Hotel_Bookings)
```

As the number of cancelled and uncancelled booking in dataset were not equally divided. In short, data was biased.

In [18]:
```
# Checking number of cancelled and uncancelled bookings
# Value indicating if the booking was canceled (1) or not (0)
hotel_bookings['is_canceled'].value_counts()
```

Out[18]:
```
0    75011
1    44199
Name: is_canceled, dtype: int64
```

So, for unbiased prediction, we divided the dataset into train and test with equal number of cancelled and uncancelled instances.

In [94]:
```
# Defining train and test datasets
# The below code divides dataset into equal propotion of target variable for unbiased testing and training of models

zeros = prepped_DF.filter(prepped_DF["is_canceled"]==0)
ones = prepped_DF.filter(prepped_DF["is_canceled"]==1)
# split datasets into training and testing
train0, test0 = zeros.randomSplit([0.8,0.2], seed=1234)
train1, test1 = ones.randomSplit([0.8,0.2], seed=1234)
# stack datasets back together
train = train0.union(train1)
test = test0.union(test1)
```

Then, after splitting the data, we implemented three models:

Model 1: GBT CLASSIFIER

MODEL 1: GBT Classifier

In [96]:
```
# Training the model
gbClassifier = GBTClassifier()
trainedModel_gbt = gbClassifier.fit(train)
```

In [98]:
```
# Testing the model
predictions_gbt = trainedModel_gbt.transform(test)
predictions_gbt
```

Model 2: DECISION TREE

MODEL 2: Decision Tree Classifier

In [106]:
```
# Training the model
dtclassifier=DecisionTreeClassifier()
trainModel_dt = dtclassifier.fit(train)
```

In [107]:
```
# Testing the model
predictions_dt = trainModel_dt.transform(test)
predictions_dt
```

Model 3: RANDOM FOREST

MODEL 3: Random Forest Classifier

In [114]:
```
rfclassifier=RandomForestClassifier()
trainedModel_rf = rfclassifier.fit(train)
```

In [117]:
```
# Testing the model
predictions_rf= trainedModel_rf.transform(test)
predictions_rf
```

Model Comparison

Our main objective in this project is to attain the best accuracy model for predicting whether a hotel booking can be cancelled in future or not. To reduce bias and classifying categorical variables at its best, we have employed Ensemble technique classifiers which are similar yet conceptually different. Below is the comparison of the output of the above-mentioned classifiers:

```
In [97]: Model_Comparison
```

Out[97]:

	S.No.	ModelName	Accuracy	Precision	Recall	ROC Curve	PR curve
0	1	GBT	0.8036	0.7982	0.9213	0.8837	0.8428
1	2	Decision Tree	0.7912	0.7768	0.9383	0.7225	0.6967
2	3	Random Forest	0.7637	0.7285	0.9963	0.8112	0.7697

As we can see that, GBT Classifier gives the best Accuracy and Precision of 80.36% and 79.12% respectively out of all the three models but the Recall is 88.37% which is least of all. Yet, we are choosing GBT as the Best Model for this problem because we believe that the major evaluation metrics for this kind of problem is Precision rather than Accuracy. If there is less precision in classifying the bookings with high probability of cancellation, the hotels can do overbooking and as a result they won't be able to accommodate all customers together. The inability of hotel to serve their customers properly can also result in deterioration of hotel's reputation in the market. That' why we believe GBT is the best model for predicting hotel booking cancellation as it is giving the best precision value.

Recommendations & Insights

- Collect mandatory Deposits before a week of arrival to avoid short-term booking cancellations
- Use the "Flight booking Strategy "
- Add more customization opportunities for the customers
- Set Discounted or Advance Purchase Rates
- Add Deals for Direct and Corporate Bookings
- Send Reminder mails to the customers before arrival period and ask for their possible ETA

NOTE: Though you would see deep learning part in our presentation, but we have not added that part to our report. As it was an approach made just to have an experience and we are not confident enough to add it to our final report.

References:

1. Zeytinci, E. (2019, December 30). Predicting Hotel Cancellations with Machine Learning. Retrieved from https://towardsdatascience.com/predicting-hotel-cancellations-with-machine-learning-fa669f93e794

2. Antonio, N., Almeida, A. D., & Nunes, L. (2019). Hotel booking demand datasets. Data in Brief, 22, 41–49. doi: 10.1016/j.dib.2018.11.126

3. Mostipak, J. (2020, February 13). Hotel booking demand. Retrieved from https://www.kaggle.com/jessemostipak/hotel-booking-demand

4. Marcuswingen. (2020, March 6). EDA of bookings and ML to predict cancelations. Retrieved from https://www.kaggle.com/marcuswingen/eda-of-bookings-and-ml-to-predict-cancelations