



BIG DATA REPORT

Final Assignment



BY: OSAMA GHALEB ZUMARI

Summer Semester

Table Of Content

Problem Statement: (3)

About Data: (2)

Step By Step Coding Process

Installing Pyspark & Resources Calculator (5)

Reading Data (6)

Preparing Data & General Statistics (7)

Visualizations & Findings (10)

JSON (20)

Preprocessing (30)

Modelling (35)

Advantages & Challenged of Data Preparation & Sourcing: (38)

Applying Hadoop & Evaluating Between Hadoop and PySpark (40)

Big Data Life Cycle And Decisions: (45)

Hadoop & Pyspark, Hive & Spark: (49)

References (55)

Problem Statement:

There are many reservations happening in hotels, most of them are accomplished successfully, but the rest may get cancelled which cost hotels empty rooms without getting benefit from them. In another meaning, these cancelations cause underutilization for hotels rooms and waste other potentials customers and guests. The statement is how we might increase the revenue by understanding the factors that affect the reservation status whether in checking in or cancelling them to prepare for any circumstances. Also, how we might ensure reservations will occur properly. We will use big data tools to handle huge amounts of data in terms of finding patterns using descriptive analysis and predictive modelling for future forecasts.

About The Data:

We collected two data formats which are a CSV file and a JSON file:

Hotel Reservations Dataset as CSV: consists of 119390 rows and 32 columns and a size of 16.4 megabyte. It expresses attributes of guests about their reservations features and a target where it expresses did that customer cancel his/her reservations or checked out properly.

The columns:

Column.	Description	Values
Hotel	The hotel type.	Two string values whether resort hotel or city hotel.
Is cancelled	Reservation statue	Binary classification whether True or False
Lead time	Amount of waiting time.	Numerical Continuous values.
Arrival year date	Year of arrival time	Three values whether in 2015 / 2016 / 2017
Arrival month date	Month of arrival time	Month values from January to December
Arrival week number	The week number according to the year	Values from 1 to 53.

Arrival day number	The day number according to the month	Values from 1 to 31.
Stays in weekend nights	Number of nights stayed in weekend days	Numerical discrete values indicate to the number of nights in weekend days.
Stays in weekdays nights	Number of nights stayed in weekdays	Numerical values indicate to the number of nights in weekdays.
Adults	Number of adults assigned for the reservation.	Numerical discrete values.
Children	Number of children assigned for the reservation.	Numerical discrete values.
Babies	Number of babies assigned for the reservation.	Numerical discrete values.
Meal	Meal plan type assigned for the reservation.	Categorical values which are undefined, self-catering, bed and breakfast, half board, full board
Country	Country of origin	Categorical values for countries in ISO 3 letters.
Market segment	Type of segment for the reservation	Categorical values for customers.
Distribution channel	Booking distribution channel.	Categorical values for customer.
Is repeated guest	Whether a guest visited the hotel or not.	Binary values of True or False.
Previous cancellations	Number of cancellations happened from a customer	Numerical discrete values.
Previous bookings not cancelled	Number of bookings happened from a customer	Numerical discrete values.

Reserved room type	Reserved room type that a customer booked.	Categorical values from A to H, L and P.
Assigned room type	The assigned room types for a customer.	Categorical values from A to I, K, L and P.
Booking changes	Number of changes happened for the reservation.	Numerical discrete values.
Deposit type	Type of deposit	Categorical values of No deposit, non-refundable, refundable.
Agent	Id of a company or organization that handles the bookings.	Numerical discrete values.
Company	Id of the company	Numerical discrete values.
Days in waiting list	How many days until the booking booked.	Numerical discrete values.
Customer type	Type of booking.	Categorical values of transient, transient-party, contract and group.
Adr	Average daily price of the room.	Numerical continuous values.
Required car parking spaces	Number of car spaces asked for the reservation.	Numerical discrete values.
Total of special requests	Number of special requests made by a customer.	Numerical discrete values.
Reservation status	The statue of the reservation.	Categorical values of checked out, cancelled or no show.
Reservation statue date	The date of confirming the statue.	Date type.

Hotels reviews dataset as JSON file: consists of 7500 rows and 35 columns and a size of 2.7 megabyte. It expresses attributes of hotels in country capitals of Italy, Spain, German, United Kingdom and France. It describes their hotels in terms of the ratings reviews and score and whether they provided transportation services or not. This data will be treated as supporting evidence for the main dataset (The CSV) in EDA section where we will try to prove some of the assumptions that we have made by using the JSON dataset. For clarification, JSON file known to be semi structured data format written textually and humans readable used in collecting web applications data. [2]

Closure: The target is to predict the reservation status and what features influence the status.

Data sources:

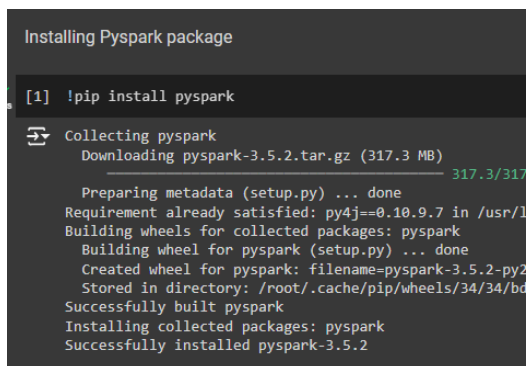
Hotels Reservations (CSV): <https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand>

Hotels Reviews (JSON): <https://www.kaggle.com/datasets/mykhailozub/500-hotels-from-airbnb-booking-and-hotelscom>

Step By Step Coding Processes:

First Section: Installing Pyspark and Resources Calculator:

First, we started installing pyspark which is an open-source framework for processing large data using distributed computing mechanisms. Which is the framework that we will use for solving this problem that is considered big data problem which is good to handle by pyspark due to its features.



```
Installing Pyspark package

[1] !pip install pyspark

Collecting pyspark
  Downloading pyspark-3.5.2.tar.gz (317.3 MB)
    Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/1
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.2-py2
  Stored in directory: /root/.cache/pip/wheels/34/34/bd
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.2
```

Next, importing time and resources usage calculator which will calculate the usage of resources and the time taken for the code to be implemented.

```
Time and Source
[2] import time
import psutil
```

Second Section: Reading Data:

Next, import spark session library to use the constructor to load our tasks on the sessions and using the benefits of pyspark. Starting with declaring “spark” is the object that holds the pyspark session. Then, reading the CSV file using the object to apply using the session and calculate how much did it take from the system to take to run the code in terms of time and resources consumed.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Hotels").getOrCreate()

path = "/content/hotel_bookings.csv"
df = spark.read.csv(path, header=True, inferSchema=True)
df.show(3)

end_time = time.time()
print("Time: ", end_time - start_time)
print("CPU: ", psutil.cpu_percent())
print("RAM: ", psutil.virtual_memory().percent)
```

The output:

```
+-----+-----+-----+-----+-----+
|      hotel|is_canceled|lead_time|arrival_date_year|arrival_date_month|
+-----+-----+-----+-----+-----+
|Resort Hotel|          0|      342|          2015|          July|
|Resort Hotel|          0|      737|          2015|          July|
|Resort Hotel|          0|         7|          2015|          July|
+-----+-----+-----+-----+-----+
only showing top 3 rows

Time:  24.289143323898315
CPU:  93.3
RAM:  16.3
```

All 32 columns are displayed but only shown these columns in the report.

Third Section: Preparing Data & General Statistics:

We will be justifying every process done.

First, we displayed the data size to ensure that all the data is uploaded in the session.

```
rows = df.count()
columns = len(df.columns)
shape = (rows, columns)
print(shape)
```

```
(119390, 32)
```

112930 is the number of rows, 32 is the number of columns.

Then, checking unique values of each feature to see what features that may has 1 unique value (which means all the instances share the same value) or that has same number of unique values as the row size (119390) which will not be useful for EDA nor Modelling.

```
from pyspark.sql import functions as F
unique_counts = {}
for column in df.columns:
    unique_counts[column] = df.select(column).distinct().count()
print(unique_counts)

{'hotel': 2, 'is_canceled': 2, 'lead_time': 479, 'arrival_date_year': 3,
```

After that, checking Duplication instances, the duplicated instances were 31994 rows. We decided to delete them for several reasons which are:

- Duplication instances may cause because of several room bookings occurred by an agent for the same event
- Deleting them will help insights be more realistic without creating biases
- Deleting them will reflect the truth especially for cancellation status. Instead of saying "multiple reservations cancelled by that agent" it will say "The agent cancelled his reservation for that event"

Checking the shape of data after deleting the duplication instances:

```
duplication = df.count() - df.dropDuplicates().count()
print(duplication)

df = df.toPandas()
df = df.drop_duplicates(keep = False)
print(df.shape)
```

```
31994
(79225, 32)
```


For this process, we needed to switch the data frame into pandas that has a unique dropping for duplication which will ensure to drop them all, unlike spark that has multimethod.

Next, we switched the null values into NumPy Nan values as shown:

```
] import numpy as np
df = df.replace('NULL' , np.nan)
df = df.replace('NA' , np.nan)
df.isnull().sum()
```

which is necessary since we switched from spark that treats null values as (None) to pandas that treats null values as NumPy Nan values. After checking the null values for each column, it is shown that these features have null values which are:

- Children column has 4 null values.
- Country column has 442 null values.
- Agent column has 11120.
- company column has 74483.

Each will be treated differently after understanding the data more to fill these nulls in a proper way.

Next, after checking the data more carefully, we decided on removing the following features:

Removing couple of features that will not help:

- Reservation status Date: Express the date of that event. Already have features that express the year, month and the day of the reservation.
- Company: Express the ID of the country. almost 95% of the values of this feature are Null.
- Arrival date week number: Express the week number according to the year's weeks. Not needed since we have the detailed date of that event.
- Is Canceled: Express whether the reservation is cancelled or occurred peacefully. We have a similar feature that shows whether the reservation got cancelled or checked in successfully called reservation status.

```
df = spark.createDataFrame(df)
df = df.drop('reservation_status_date', 'company', 'arrival_date_week_number', 'is_canceled', 'adr')
```

After that, we saw that two features represent the nights spent, one for the weekdays and the other is for weekend days. So, instead of having two features, one for the nights spent in weekend days and one for the nights spent in weekdays. We made an aggregation function to make them a single feature that contains the total nights spent during the reservation. And then removed those two features. In that case, more useful information represents one feature. In addition, applying dimension reduction lowers the load on the machine learning models and reduces the complexity of models. Thus, preventing overfitting.

```
from pyspark.sql.functions import col

df = df.withColumn("number_of_nights", col("stays_in_weekend_nights") + col("stays_in_week_nights"))
df = df.drop("stays_in_weekend_nights", "stays_in_week_nights")
```

Then, we created a data frame that contains only the instances that has the reservation status = Check-Out. Then a condition to see the instances that has number of nights equals to 0.

```
from pyspark.sql.functions import when

checker = df.filter(col('reservation_status') == 'Check-Out')
true_count = checker.filter(col('number_of_nights') == 0).count()
false_count = checker.count() - true_count

print(f"\nCheck-Out With No Nights: {true_count}")
print(f"Check-Out With Nights: {false_count}")
```

```
Check-Out With No Nights: 591
Check-Out With Nights: 57861
```

We considered the reservation statue means reserving the hotel room, which even if the reservation did not complete 12 hours, it will be counted as 1. So, we increased 1 for all instances that has Check-Out value in the reservation status only.

```
df = df.withColumn(
    'number_of_nights',
    when(col('reservation_status') == 'Check-Out', col('number_of_nights') + 1).otherwise(col('number_of_nights'))
```

Lastly, we did preparation for the arrival date month which is transforming each categorical month to its corresponding numerical number.

```

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

for month, month_number in month_mapping.items():
    df = df.withColumn('arrival_date_month', when(col('arrival_date_month') == month, month_number).otherwise(col('arrival_date_month')))

```

In addition, we did feature extracting by applying conditions on the month and day dates to create a new feature called Season that has values of (Summer, Fall, Winter, Spring).

```

conditions = [
    ((col('arrival_date_month') == 12) & (col('arrival_date_day_of_month') >= 21)) |
    ((col('arrival_date_month') <= 3) & ((col('arrival_date_month') != 3) | (col('arrival_date_day_of_month') < 20))),

    ((col('arrival_date_month') == 3) & (col('arrival_date_day_of_month') >= 20)) |
    ((col('arrival_date_month') <= 6) & ((col('arrival_date_month') != 6) | (col('arrival_date_day_of_month') < 21))),

    ((col('arrival_date_month') == 6) & (col('arrival_date_day_of_month') >= 21)) |
    ((col('arrival_date_month') <= 9) & ((col('arrival_date_month') != 9) | (col('arrival_date_day_of_month') < 23))),

    ((col('arrival_date_month') == 9) & (col('arrival_date_day_of_month') >= 23)) |
    ((col('arrival_date_month') <= 12) & ((col('arrival_date_month') != 12) | (col('arrival_date_day_of_month') < 21))) ]

seasons = ['Winter', 'Spring', 'Summer', 'Fall']
df = df.withColumn('season',
    when(conditions[0], seasons[0])
    .when(conditions[1], seasons[1])
    .when(conditions[2], seasons[2])
    .otherwise(seasons[3]))

```

Closure: we calculated the time and resources consumed for the preparation of data before visualization section which was as the following:

```

end_time = time.time()
print("Time: " , end_time - start_time)
print("CPU: " , psutil.cpu_percent())
print("RAM: " , psutil.virtual_memory().percent)

Time: 93.32851266860962
CPU: 92.2
RAM: 29.6

```

Section Four: Visualization:

First, we imported the needed libraries which are seaborn and matplotlib for creating visualizations and plots that will help us build assumptions and theories. In addition to well decisions making enhanced with evidence. Every plot is described, justified and discussed its findings.

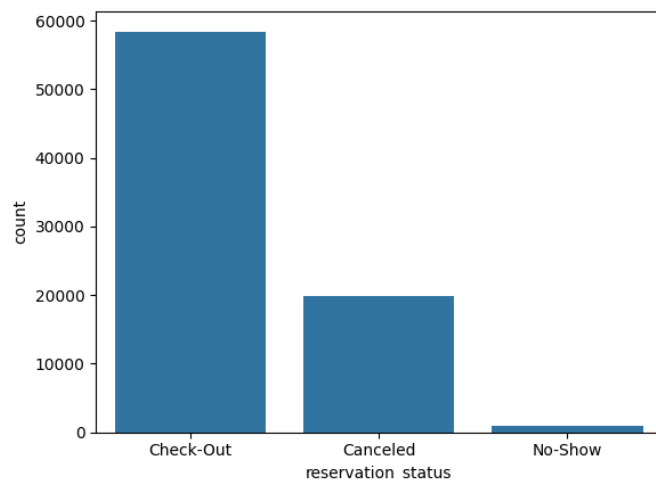
Tools And Techniques:

The tools were python by importing the needed libraries which are seaborn and matplotlib that are specialized for visualizing and creating plots and charts to represent the data in a visualized

way to make it easier to communicate with stakeholders and non-technical people. These two libraries are great for this task since seaborn is easy to use and matplotlib provides the ability to customize the plot in the desired way. These plots would help decision makers to build up a decision enhanced with evidence which resulted in lower risks and mistakes and build up a confident decision will improve the organization outcomes.

The charts that were used are as the following:

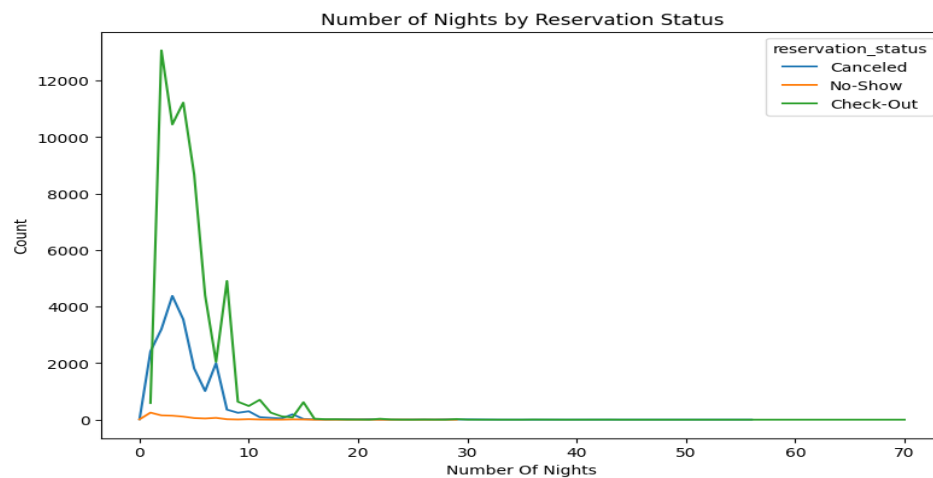
Chart	Description
Bar Chart	Compare values of different categories where each bar represents the value of a specific category.
Pie Chart	Compare multiple categories where each category takes a slice of a circle that represent the proportional of the category frequent in the data.
Line Chart	Visual the change of data according to x axis feature such as dates where it is used to visual trends.



Plot (1)

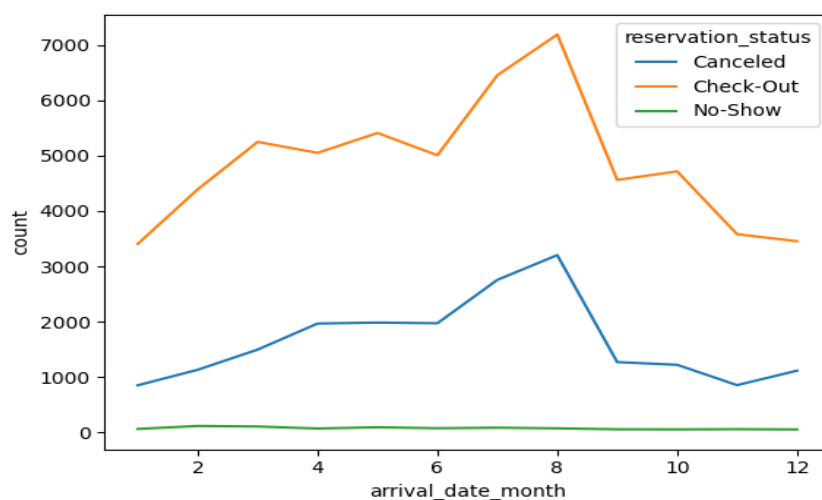
First, plot (1) shows that the target is imbalanced where most of observations are checked-out which approximately 75% of the data. While no show or cancelled fill the remaining 25% of the

data. To remind, checkout also indicates that the reservation is done properly, and no underutilization happened (no empty rooms that are not used) and cancellation and no show indicates that the reservation went mistakenly, and underutilization happened (there are empty rooms that are not used). In addition, it may require data balancing for the modelling step to prevent biases on the model learning.



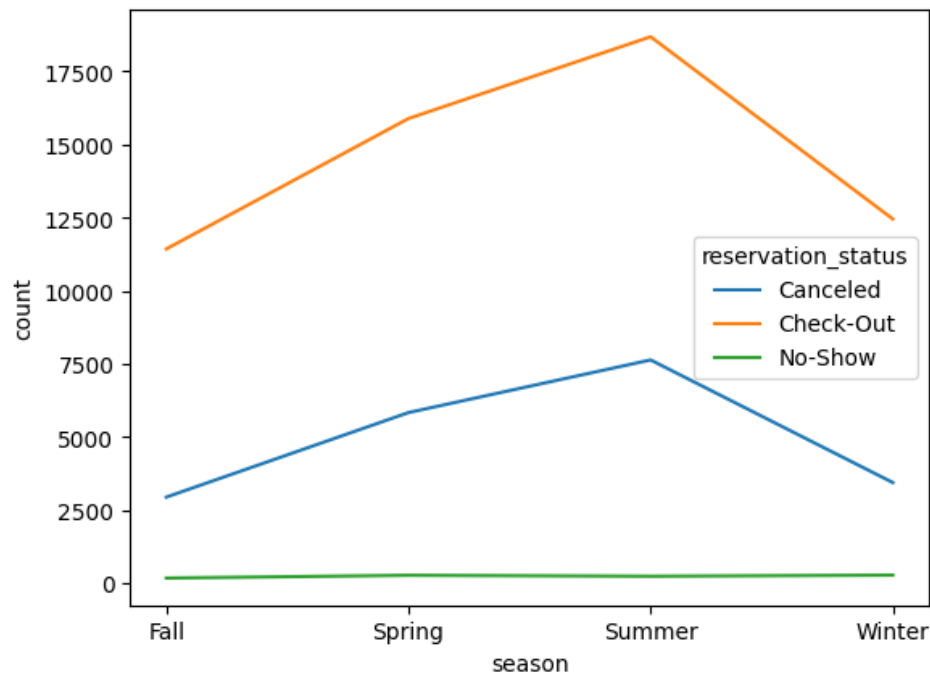
Plot (2).

Grouping the number of nights and the reservation statuses to plot the number of nights and the corresponding reservation as line plot. From the plot, it indicates that guests are most likely to reserve hotel rooms for 2 to 6 nights. Which increases the chance to cancel the reservation since a high number of reservations happened for that number of nights. Approximately the lines are similar which indicates that the increase of reservations would increase the chance of cancellation as the successfully checking out will also increase.



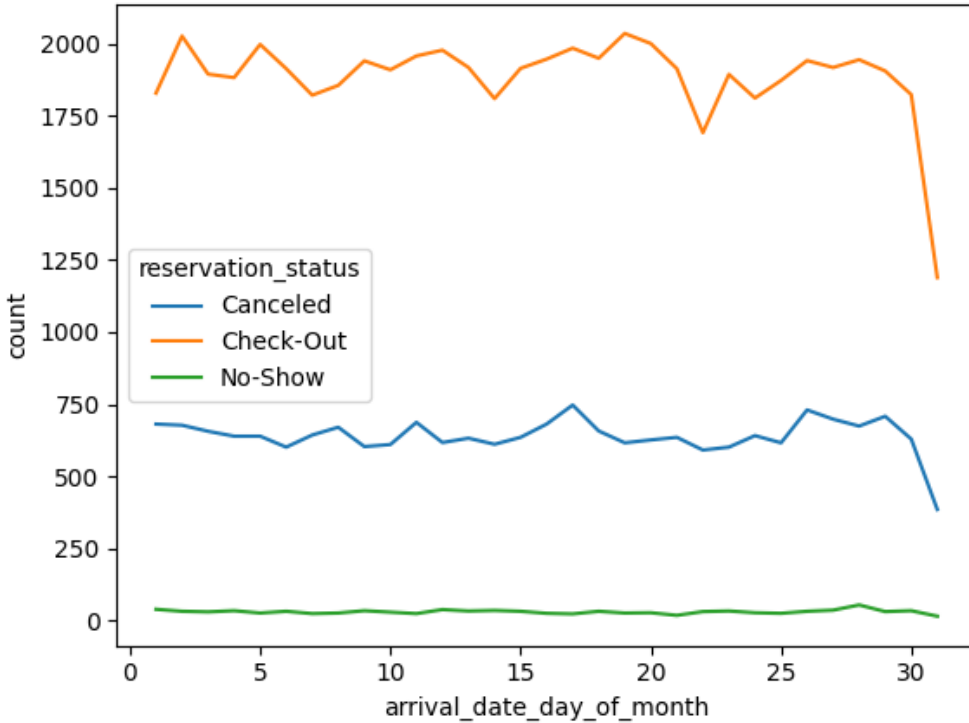
Plot (3)

Grouping the arrival date month and reservation status and plotting them as a line plot to see the number of reservations for each month. It indicates that the reservations occur more in July and August. Which also shows that the increasing in number of reservations would logically increase all status checking out or cancelling but for sure the checking out will be higher. Furthermore, it seems that the highest reservations occur is in summer semester, which the next plot would be an ensuring that what we are saying is true.



Plot (4)

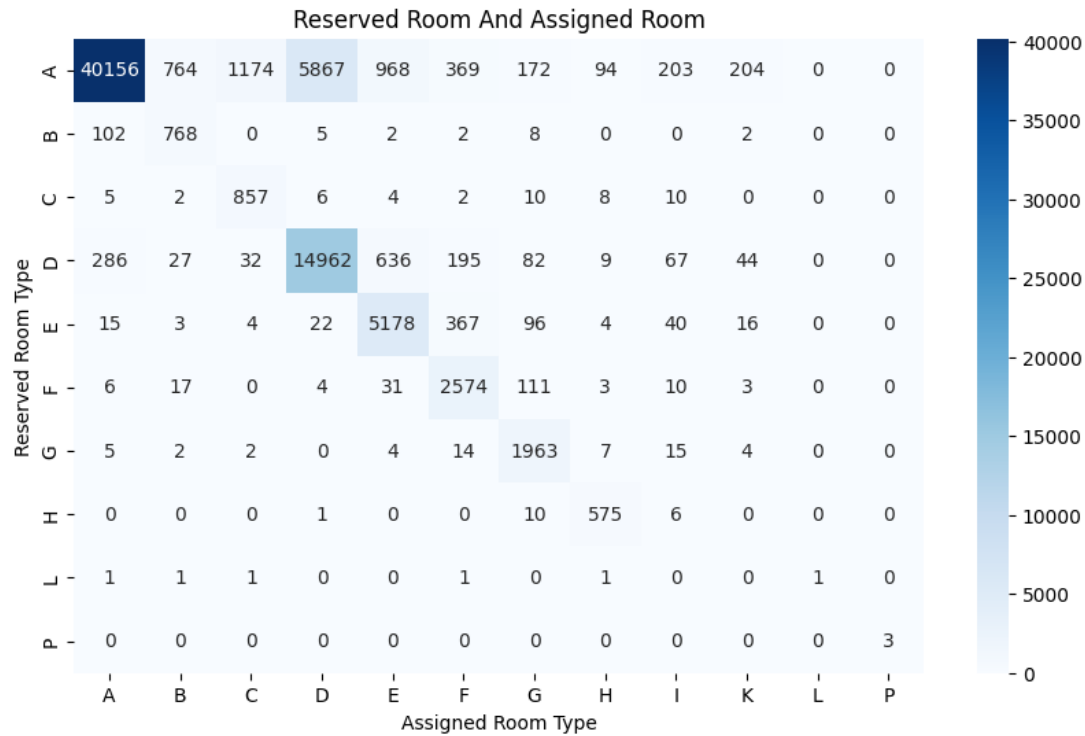
Grouping the season with the reservation status. Like the previous plot (3) but indicates to in which season the most reservations are. Also, it proved that the most reservations happened in the summer whether the status is checking out or cancelling.



Plot (5)

Grouping days of the month and the reservation status. The plot indicates that agents usually do not prefer to reserve in the last days of a month which shows that the least checking out is at the end of the month.

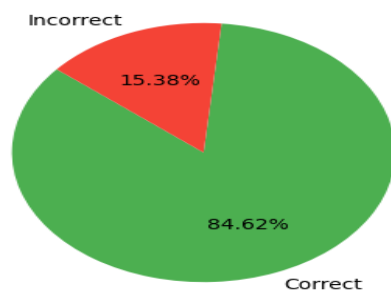
Now, we created a contingency table then use it as a confusion matrix to compare between the reserved room type and the assigned room type. It indicates that most of agents got assigned to their reserved room. However, 15.38% assigning mistakes where agent reserved for one type and got assigned to another type. Then we did calculations to detect how much of that percentage (15.38%) is cancelled reservations. It shows that 3.06% out of the 15.38% mistakes were cancelled reservations but the rest are not. Which indicates that the room type is not a factor affecting the reservation status.



Plot (6).

This is a confusion matrix for the reversed room type and the assigned room type where it shows how often categories occurred together. In another meaning, it describes how often each category happened with another category. For our example, it shows how each room type was reversed and what was assigned for the guests. In the next plot, it shows the mistake percentage, and the proportion of correct assigning based on the reserved room type as Pie Chart which describes proportion of multiple categories where each category takes a slice of the pie to compare between them on what dominate the others based on their occurrence for example. Shows the percentage of a category based on specific target. For example, list of products and the revenue proportion for each product.

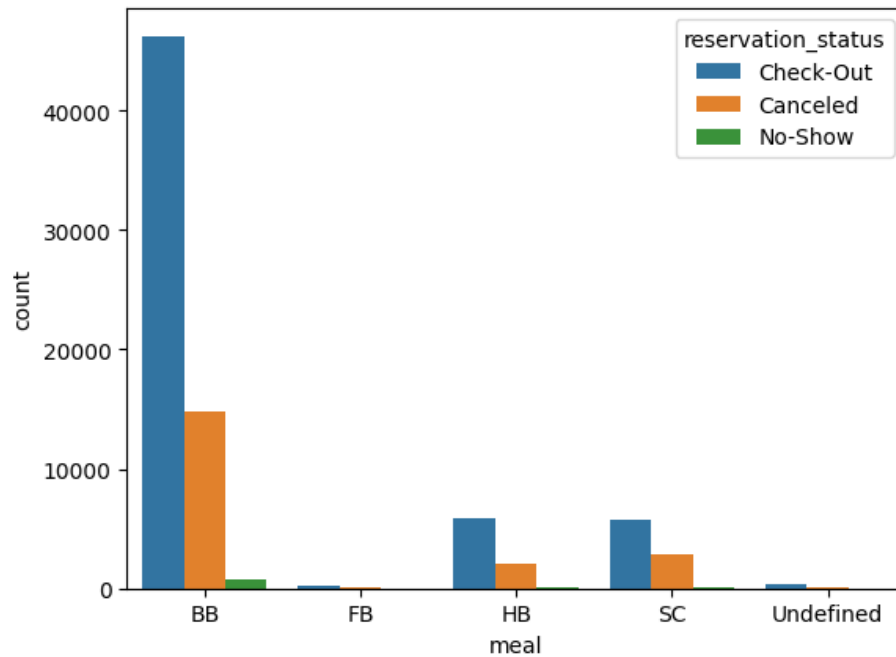
Proportion of Correct vs. Incorrect Room Types



Plot (7)

Cancellation Percentage for Incorrect Room Types: 3.06% Out Of The 15.38%

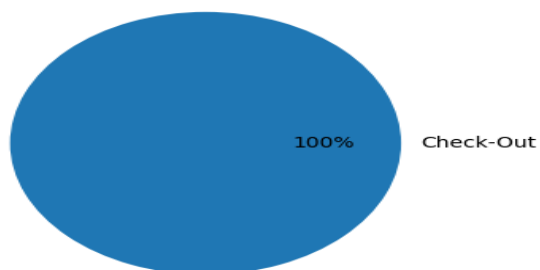
As mentioned, from the 15.38% error percentage, there are only 3.06% of that 15.38% error percentage has cancelled reservation status. In summary, the assigned room could be a factor that affects the target since the room type did not match the guest's desire which could change his mind in terms of cancelling the reservation.



Plot (8)

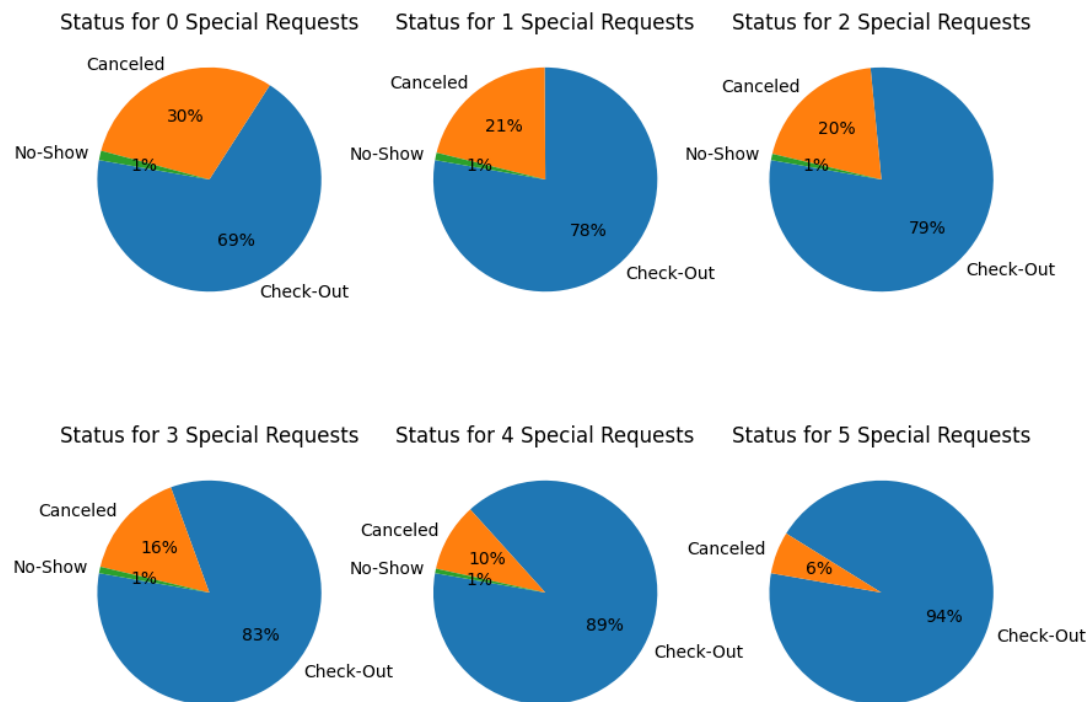
Plotting meal plans according to the reservation status. It indicates that there is no clear relation about meals plan to influence the reservation status. In addition, most of reservations were assigned to BB (bed and breakfast) only.

Reservation Status By Getting 1 Or More Car Parks



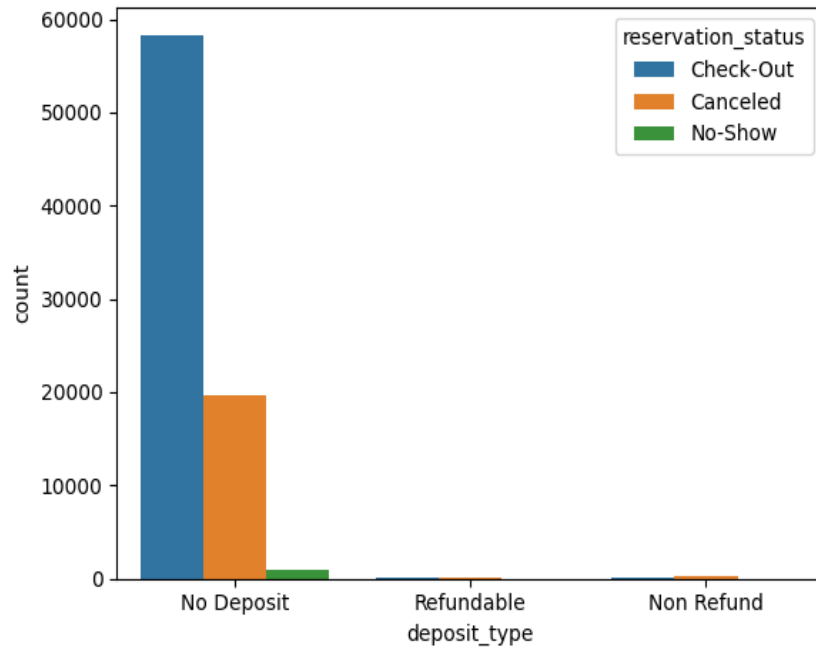
Plot (9)

We created a data frame that only contains 1 or more car parking spaces. Then plotting pie plot that shows how the reservation status is when the condition is applied. According to the data, it indicates that when an agent requests 1 or more car parks spaces, the reservation would be checked out 100%. Which indicates that car parking spaces requests are a huge influence on the reservation status. We could assume that transportation methods in general are important since guests requested car parking spaces which means they would use them for transportation. We could assume that transportation in general is an important factor that may affect the target, which is the reservation status.



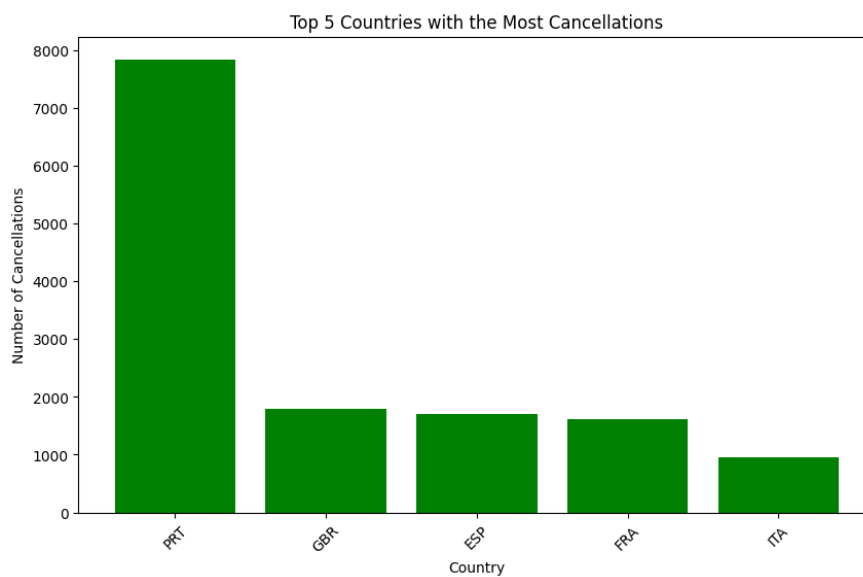
Plot (10)

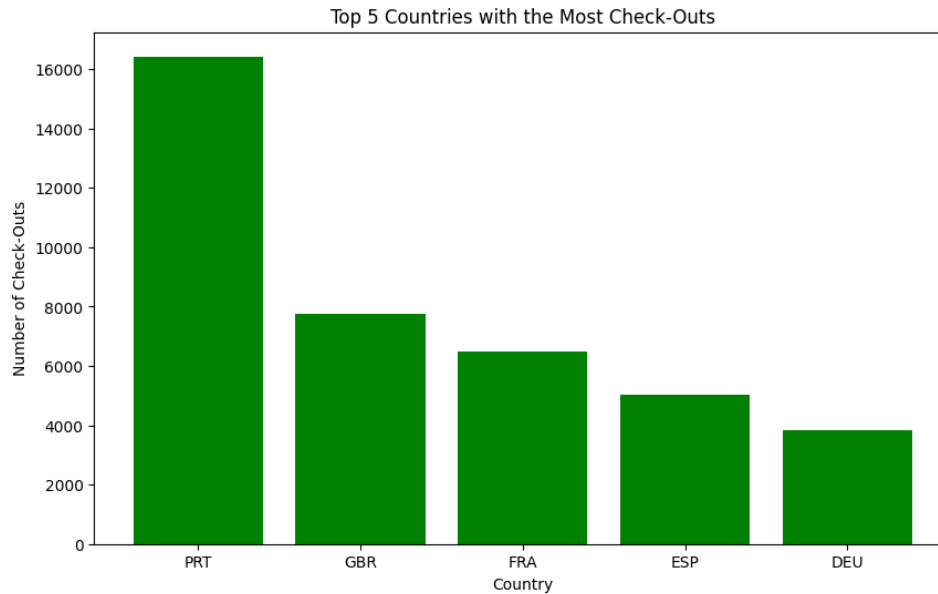
plotting the reservation status when agents ask certain number of special requests. First, if 0 requests occurred, there is a chance of 30% that the reservation would be cancelled. However, when more requests got asked, the percentage of cancellations decreased. 1 special request decreased the cancellation status to 21%. 2 special requests decreased the cancellation status to 20%. 3 special requests decreased the cancellation status to 16%. 4 special requests decreased the cancellation status to 10%. Lastly, 5 special requests decreased the cancellation status to 6%. It indicates that the more special requests are, the less chance of the reservation getting cancelled. Which is a good factor for influencing the target.



Plot (11)

Plotting deposit type according to the reservation status. It indicates that almost all reservations were No deposit type which cannot conclude a relation from it. In these situations, it is preferred to remove these features that almost share one value to all the column because it will be noisy for predictive models to have features that won't help them on their decisions. Therefore, we decided to drop this feature.





Plot (12 and 13)

Plotting the most five cancellation reservations happened in countries. It indicates that Portugal is significantly highest cancellation status, then United Kingdom, Spain, France and Italy share almost the same cancellation observations. However, the most five reservations that occurred successfully was also in Portugal significantly compared to other countries. Which indicates that the null values of the country feature could not be filled randomly or filled on these countries to prevent biases from happening.

Summary Of Findings:

1. Data is imbalanced.
2. Most reservations are between 2 to 6 nights.
3. The highest reservations are in July and August, which is in Summer semester.
4. The least number of reservations are at the end days of the month.
5. Mistaken assigning room types is not a huge factor for affecting the target.
6. Meal plan types are not a huge factor for affecting the target.
7. Requesting one or more car park spaces will grant 0% cancellation probability.
8. The more special requests are asked, the less probability of cancellation.
9. Deposit type has no influence on the target.
10. The most reservations happened and cancelled is in Portugal then UK, Spain and France.

Closure: the time taken for this section to be accomplished is as shown:

```
Time: 14.5709228515625
CPU: 86.2
RAM: 27.1
```

Section Five: JSON:

We have analyzed the five capitals of Italy (Rome), Spain (Madrid), France (Paris), German (Berlin), United Kingdom (London). First, we created a spark session and loaded the JSON file in it to start working.

```
[ ] from pyspark.sql import SparkSession
from pyspark.sql.functions import col, explode
spark = SparkSession.builder.appName("JSONParsing").getOrCreate()
pathjson = "/content/Rome.json"
json_df = spark.read.json(pathjson)

booking_df = json_df.select(explode(col("bookingHotels")).alias("hotel")) \
    .select(
        col("hotel.link").alias("link"),
        col("hotel.price.currency").alias("currency"),
        col("hotel.price.taxesAndCharges").alias("taxes_and_charges"),
        col("hotel.price.value").alias("price_value"),
        col("hotel.rating.score").alias("rating_score"),
        col("hotel.rating.reviews").alias("rating_reviews"),
        col("hotel.rating.scoreDescription").alias("rating_score_description"),
```

Then, selecting all the columns of the booking hotels data frames:

```
booking_df = json_df.select(explode(col("bookingHotels")).alias("hotel")) \
    .select(
        col("hotel.link").alias("link"),
        col("hotel.price.currency").alias("currency"),
        col("hotel.price.taxesAndCharges").alias("taxes_and_charges"),
```

```
hotels_com_df = json_df.select(explode(col("hotelsComHotels")).alias("hotel")) \
    .select(
        col("hotel.link").alias("link"),
        col("hotel.price.currency").alias("currency"),
        col("hotel.price.value").alias("price_value"),
        col("hotel.price.withTaxesAndCharges").alias("price_with_taxes_and_charges"),
```

However, after deep looking, we only would benefit from the price value, rating score and subway access. So, we selected these three columns from the booking hotels data frame. And the hotels com data frame, we took price value and rating score only.

```
booking_df_selected = booking_df.select(
    col("price_value"),
    col("rating_score"),
    col("subway_access")
)
```

```
hotels_com_df_selected = hotels_com_df.select(
    col("price_value"),
    col("rating_score"),
)
```

Then, after getting these two data frames, we switched from spark to pandas and merged them together on a join where if price value and rating score were equal. It would be rare to have a hotel that shares the same price value and rating score from two different websites. However, just to prevent any consistency. And the method is outer which means any additional rows that do not match the price value and rating score will be added to the data frame as new rows.

```
df_hotel = hotels_com_df_selected.toPandas()
merged_df = pd.merge(df_booking, df_hotel, on=['price_value', 'rating_score'], how='outer')
```

Then, the feature that is called rating score has unique value called (no rating) we decided to replace it with 0 because all our interest is the better score for hotels which they would be the influencer on the decision.

```
merged_df['rating_score'] = merged_df['rating_score'].replace('No rating', 0)
merged_df['rating_score'] = merged_df['rating_score'].astype(float)
merged_df = merged_df[merged_df['rating_score'] != 0]
```

Lastly before the visualization of JSON files, we created a list of bins and labels where 0 to 5.9 will be called Poor, 6 to 6.9 will be called Average, 7 to 7.9 will be called Good, 8 to 8.9 will be called Very Good, 9.9 will be called Wonderful. The last number for all intervals is not included because of the function right which is equal to False. In addition, we excluded 10 since it is not reasonable to have a hotel ten out of ten expect having only very few reviewers who rated that hotel. So, to prevent inconsistencies, we decided not to include a score of 10.

```
bins = [0, 6, 7, 8, 9, 10]
labels = ['Poor', 'Average', 'Good', 'Very Good', 'Wonderful']
merged_df['rating_description'] = pd.cut(merged_df['rating_score'], bins=bins, labels=labels, right=False)
```

This method added a new feature called rating description where each instance has a corresponding description according to the rating score for the hotel.

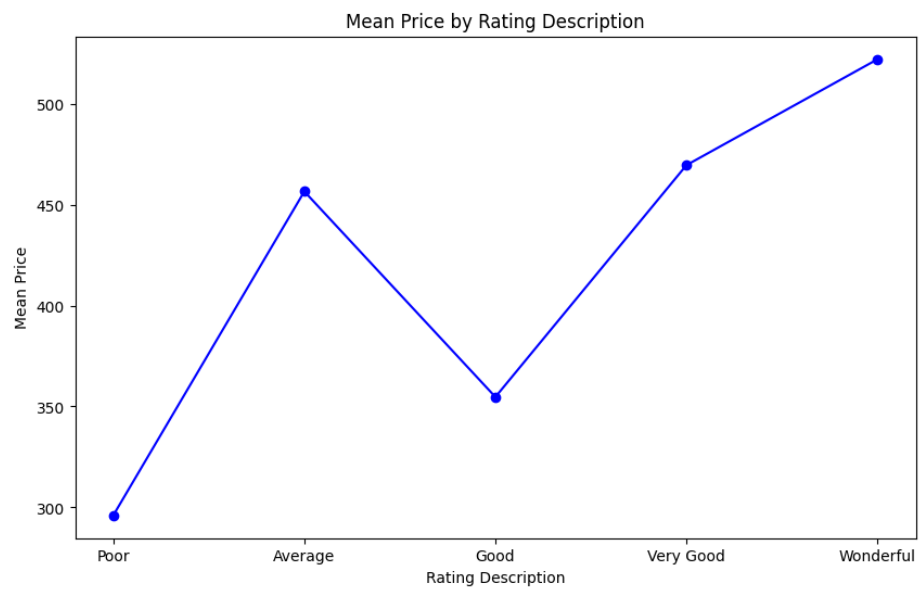
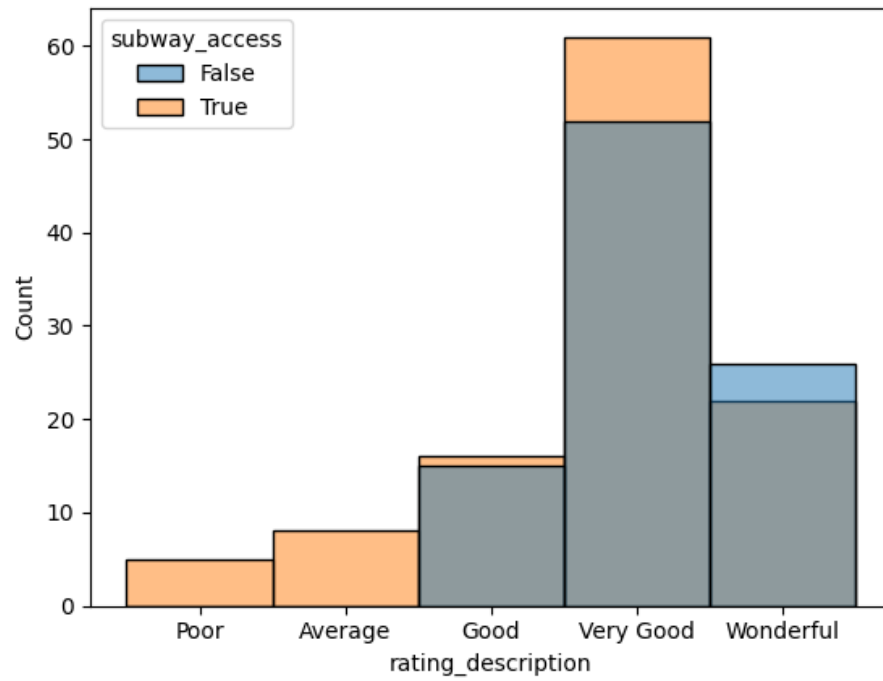
To clarify:

Price value: indicates the price of a night in a specific hotel.

Rating score: rate the hotel from 0 to 10.

Subway access: providing transportation for the subway. We considered that subway access also means the ability to have easy access to transportation.

Now, visualization for each capital, the chosen plots are a bar count plot that measures the frequency of the categories. We added a hue dimension which is a third dimension that clarifies a condition to be applied in the plot. This plot will show the rating description count and the availability of the subway access. The second plot is a line plot where it shows the mean prices for each rating description. These two plots will help us to determine the importance of having transportation method that will reflect on the previous plot number 9 that shows when a guest requests a car the percentage of the cancellation of the reservation is 0% which could be related since car parks and subway accessibility are both transportation methods. In addition, it will help us to determine whether the prices are high or low and what impacts the cancellation of reservation. Also, we have created a data frame for each country of those five capitals from the CSV to measure the cancellation rate of each country. It would help us to determine whether the price factor is a new factor and whether the transportation method is truly important or not. Starting with Italy Rome first, the plots are as the following:



```

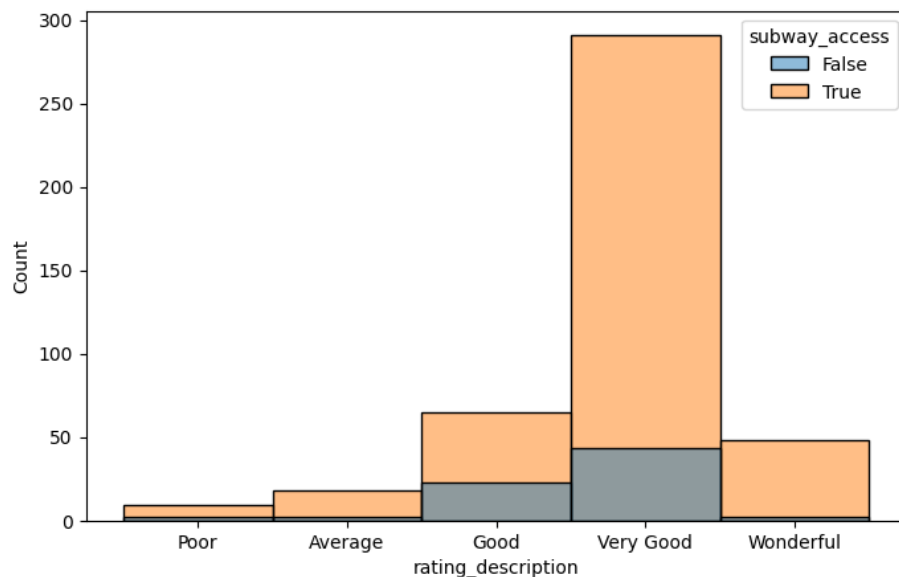
reservation_status  total
0      Check-Out    1772
1      Canceled     947
2      No-Show      16

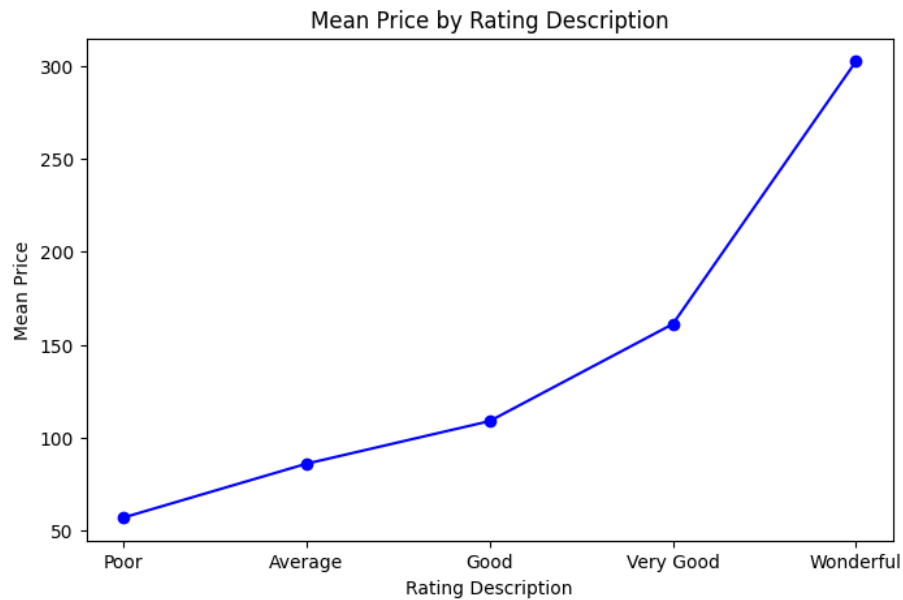
```

Cancellation Rate: 34.63%

The mean prices in Italy for good and better hotels between 370 to more than 500 which is expensive comparing to other Europe capitals such as France and Madrid. In addition, their hotels provide no subway access which means no transportation provided by the hotel nor near the hotel. The fact as from our analysis in the CSV file of hotel reservations, we saw that parking car spaces has huge influence on the reservation status. In another meaning, transportation methods are important for guests. After further analysis, we saw that the cancellation of reservations rate in Italy is 34.63% of all Italy destinations. Which considered high compared to other European capitals. We could assume that transportation is indeed important for all guests. In addition to the prices that may have big influence on the reservation status as well because other capitals (such as Paris for example) has low average prices for good and better hotels and providing subway access (which means availability of transportation method).

That's it for Rome, now with Madrid:



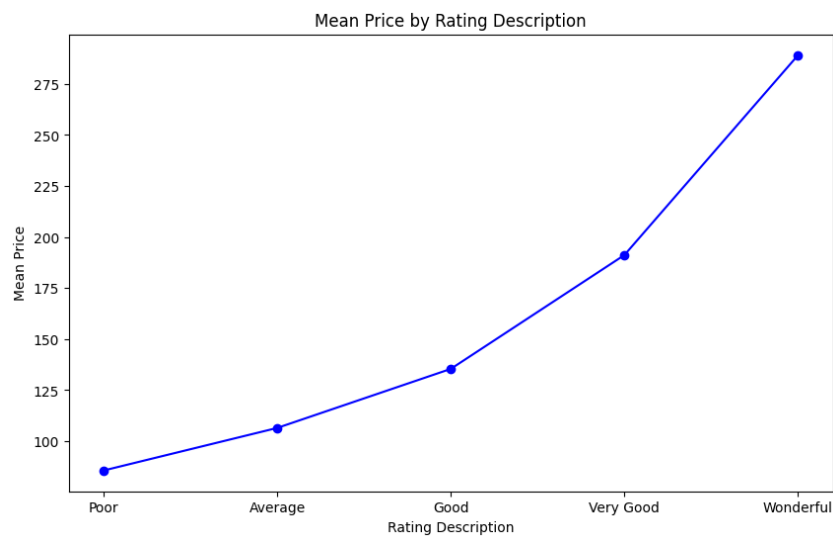
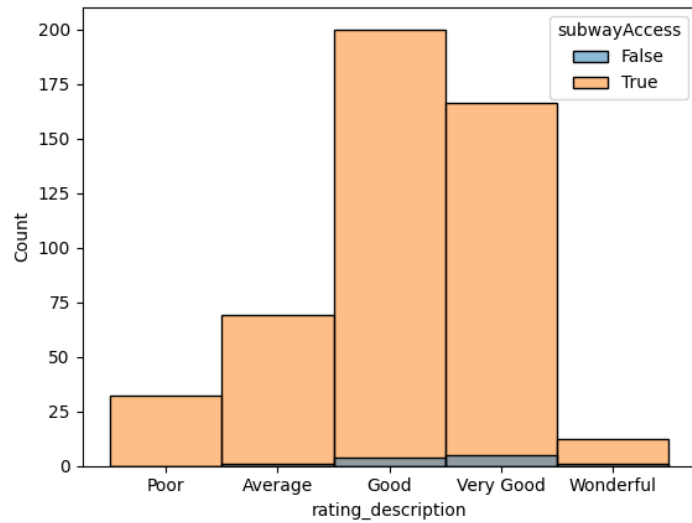


reservation_status	total
0 Check-Out	5026
1 Canceled	1669
2 No-Show	30

Cancellation Rate: 24.82%

We can ensure that transportation methods are important for guests where it shows that hotels would get High Score for having transportations methods whether served by the hotel or nearby. In addition, the mean hotel prices in Spain are relatively low compared to Italy where prices for good and better hotels are between 100 to 300. Which also could be an assumption for why Spain has less cancellation rate that is equals to 24.82% for Spain destinations.

That's it for Madrid, now with Paris:



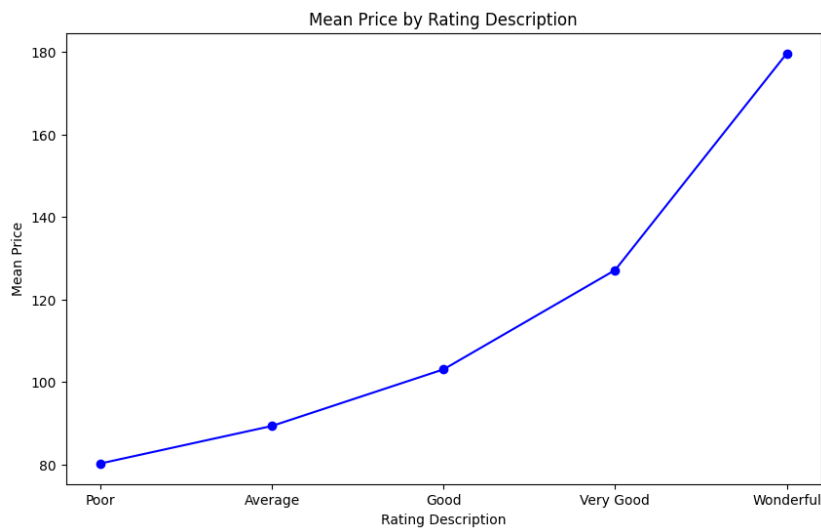
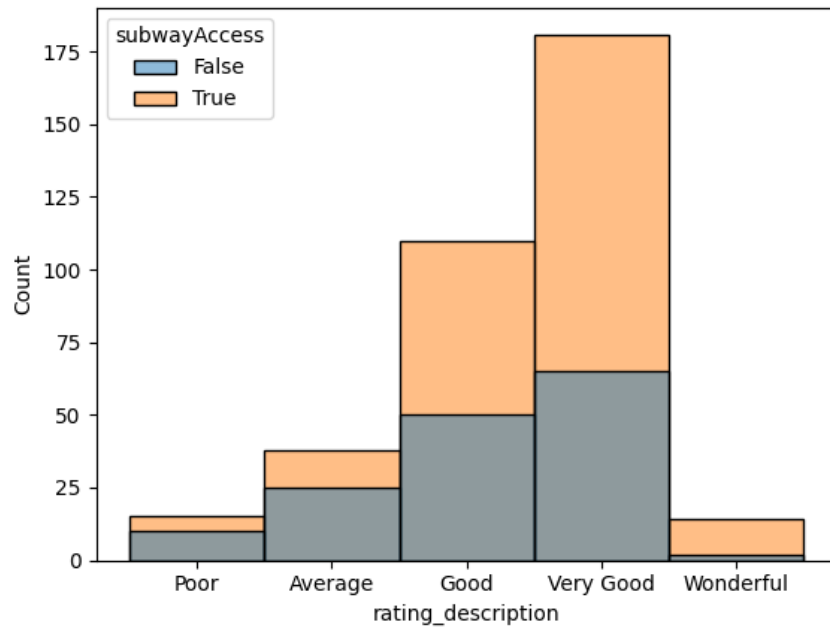
reservation_status	total
0 Check-Out	6498
1 Canceled	1578
2 No-Show	30

Cancellation Rate: 19.47%

Even though transportation method indeed shown its importance on the reservation status. Paris has shown an outstanding performance in providing transportation method by hotels. Which explains why France has lower cancellation rate with 19.47% for France destinations only which is lower than Italy and Spain for now. In addition, the mean hotel prices in France are relatively low compared to Italy where prices for good and better hotels are between 130 to 300. Which

also prove that cancellation rate influenced by the price of hotels because that's what Spain and France proved.

That's it for Paris, now with Berlin:

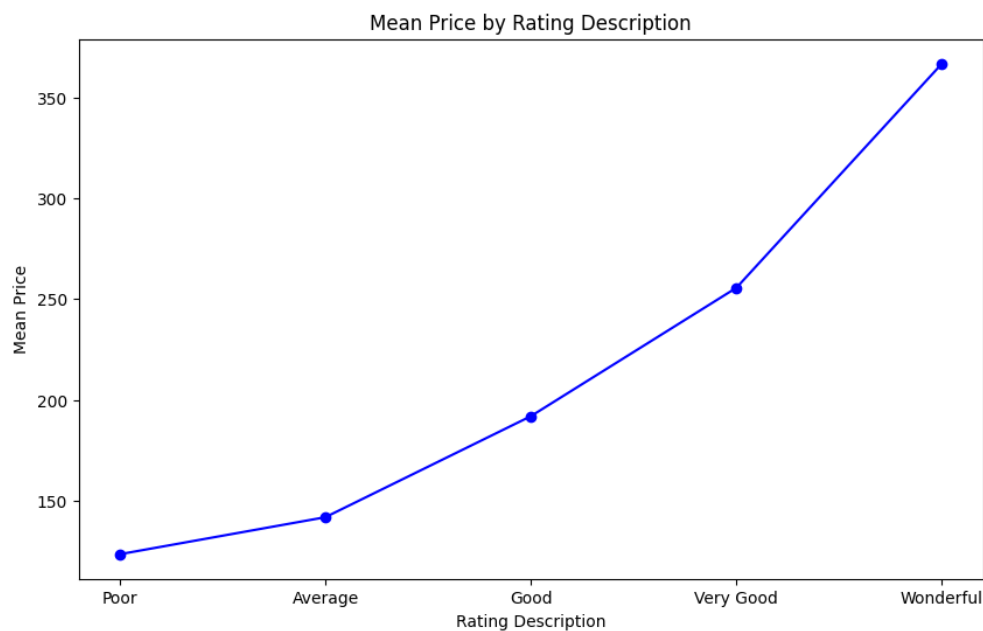
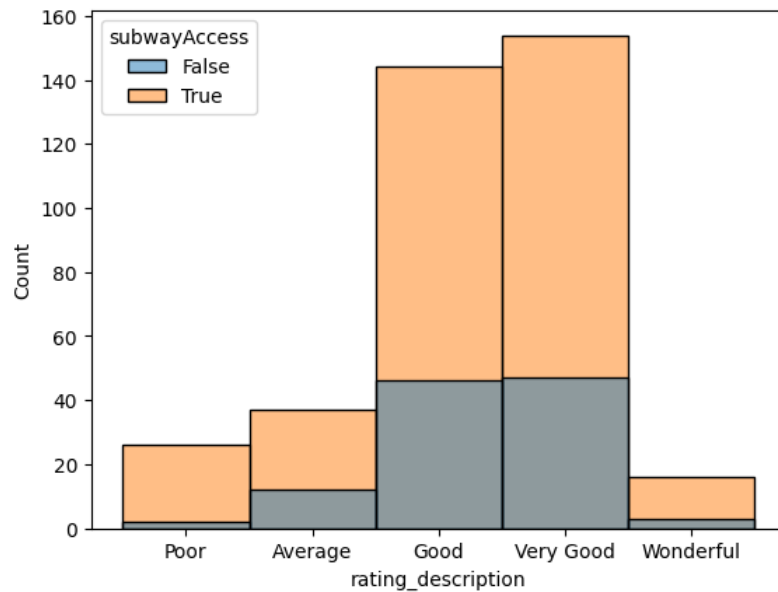


```
reservation_status  total
0      Check-Out    3836
1      Canceled     948
2      No-Show      11

Cancellation Rate: 19.77%
```

For German, prices of hotels are lower than all countries mentioned for now where the mean of prices for good and better hotels are between 110 to 190 which is super cheap for a wonderful hotel to be as price as good hotel in Italy. In addition, even though not all hotels provide transportation methods. Prices has played a crucial role on the cancellation rate in German which is equal to 19.77% of all German destinations.

That's it for Berlin, now with London:



reservation_status	total
0 Check-Out	7770
1 Canceled	1749
2 No-Show	38

Cancellation Rate: 18.30%

In London hotels providing transportation methods are like German but with higher prices where good and better hotels are between 180 to 380. It is almost the average of all these five capitals which is not expensive but not the cheapest. However, no one would miss a flight to London as said in wp-travel website, London is the second most visited capital in the world, which would explain the low cancellation rate between all with a rate of 18.3% for all United Kingdom destinations [1].

To sum up all the assumptions that became clear after analyzing JSON data files:

1. Transportation methods are important for guests, which proved that it affects the cancellation statue. It was proved in both files CSV and the JSON, where the CSV made a clear statement that requesting one or more car park spaces will zero the probability of cancelling the reservation. While the JSON files proved that transportation methods are important and play a crucial role on the reservation statue.
2. Prices proved to be a play maker for the cancellation rate where lower average prices have less cancellation rate. That's what the JSON files proved, which would make us ask the data collectors to start collecting prices of each hotel because as it was proven that prices have an influence on the target which is lowering the cancellation rate.

There is a function called aggregation which is combining data frames together. In simple way, we combine two data frames to create a data frame according to the Join used. There are many Joins methods on how to combine the frames together such as Inner Join where it creates a data frame for shared value of a determined column.

Closure: the time taken to run section five the JSON files is as shown below:

```
Time: 10.652360916137695
CPU: 15.0
RAM: 22.8
```

Section Six: Preprocessing:

Preprocessing is an essential step to redefine the data in a better way that will enhance the model's learning. It involves removing unnecessary features and duplications, filling missing values with the proper values. Also, removing inconsistencies and outliers. We already covered some of the preprocessing techniques in *Section Three: Preparing Data & General Statistics*. In this section, we filled the null values for the three features, country, children and agent as the following:

For Country:

We created a new data frame that contains the instances that have null value in the country feature. And creating a new feature that holds the number of countries that agents has served that country as shown down:

```
from pyspark.sql.functions import col, countDistinct, isnan
df = spark.createDataFrame(df)
hi2 = df.filter(isnan('country'))

contryagentfill = hi2.groupBy('agent').agg(countDistinct('country').alias('unique_countries'))
total_countries_served = df.groupBy('agent').agg(countDistinct('country').alias('countries_served_number'))

contryagentfill = contryagentfill.join(total_countries_served, on='agent', how='left')
contryagentfill.show()
```

agent	unique_countries	countries_served_number
15	1	26
154	1	3
468	1	9
446	1	1
139	1	2
146	1	4
350	1	5

We decided to fill the Null values of the country feature by using the mode of the agent of the corresponding instance that has the country null value. Which means that each agent has a specific mode of what countries he has served the most. So, each null value in the country feature will be filled with the mode of that corresponding agent.

```

from pyspark.sql.functions import col, count, when, first, row_number
from pyspark.sql.window import Window

mode_df = df.groupBy('agent', 'country').agg(count('*').alias('frequency'))

windowSpec = Window.partitionBy('agent').orderBy(col('frequency').desc())
mode_df = mode_df.withColumn('rank', row_number().over(windowSpec)).filter(col('rank') == 1).drop('frequency', 'rank')

df = df.join(mode_df.select('agent', 'country').withColumnRenamed('country', 'mode_country'),
             on='agent', how='left')

df = df.withColumn('country', when(col('country').isNull(), col('mode_country')).otherwise(col('country')))

df = df.drop('mode_country')

```

For example, if agent number 5 served Italy the most, all null values of country instances that has agent number 5 will be filled as Italy.

For Children:

First, we printed the instances of the null values of the children feature. Then created a list of conditions to filter the data frame to display the features and attributes that are in common with the null instances to fill the null values based on the instances that share common features (according to the conditions).

```

filtered_df = df.filter(isnan('children'))
filtered_df = df.filter(
    (col('arrival_date_month') == 8) &
    (col('meal') == 'BB') &
    (col('country') == 'PRT') &
    (col('deposit_type') == 'No Deposit') &
    (col('days_in_waiting_list') == 0) &
    (col('customer_type') == 'Transient-Party') &
    (col('required_car_parking_spaces') == 0) &
    (col('reservation_status') == 'Canceled') &
    (col('babies') == 0) &
    (col('is_repeated_guest') == 0) &
    (col('previous_cancellations') == 0) &
    (col('previous_bookings_not_canceled') == 0) &
    (col('reserved_room_type') == 'B') &
    (col('assigned_room_type') == 'B') &
    (col('booking_changes') == 0)
)
filtered_df.show()

```

It shows that there is one instance that shares similar features between the null instances and has 0 children. So, we decided to fill these 4 instances with 0 children based on the instance that shared the same features with them and have 0 children.

arrival_date_month	arrival_date_day_of_month	adults	children
8		3	2
8		5	3
8		13	2
8		5	2
8		4	2

```
from pyspark.sql.functions import col, lit, when, isnan
df = df.withColumn('children', when(isnan('children'), lit(0)).otherwise(col('children')))
value_counts = df.groupBy("children").count()
value_counts.show()
```

children	count
3	74
0	71052
1	4559
10	1
2	3539

Before we headed to the Agent preprocessing, we decided to use something unique for filling agent null values which is using machine learning algorithms to predict what the missing value should be. Before that, we needed to start the Encoding part which converts categorical values into numerical values so that the machine can understand them.

Encoding is a crucial part for modelling that converts categorical features into numerical features so that machines can handle them. First, using the replacing method,

Hotel: 0 for resort hotel, 1 for city hotel.

Meal: 0 for undefined, 1 for SC, 2 for BB, 3 for HB, 4 for FB.

Season: 1 for Winter, 2 for Spring, 3 for Summer, 4 for Fall.

Deposit Type: 0 for Non-Refund, 1 for No Deposit, 2 for Refundable.

Reservation status: 0 for cancelled or no show since both indicates that the room is empty which caused underutilization, 1 for Check Out or successfully reserved and the room is not empty.

Then, for the features that have a lot of categorical values, we used indexer encoding that automatically encode the values. The features that got applied using the indexer are country, market segment, distribution channel, reserved room type, assigned room type, customer type.

```

indexer_country = StringIndexer(inputCol='country', outputCol='country_index')
indexer_market_segment = StringIndexer(inputCol='market_segment', outputCol='market_segment_index')
indexer_distribution_channel = StringIndexer(inputCol='distribution_channel', outputCol='distribution_channel_index')
indexer_reserved_room_type = StringIndexer(inputCol='reserved_room_type', outputCol='reserved_room_type_index')
indexer_assigned_room_type = StringIndexer(inputCol='assigned_room_type', outputCol='assigned_room_type_index')
indexer_customer_type = StringIndexer(inputCol='customer_type', outputCol='customer_type_index')

```

Then, fitting the indexer in the data frame where each unique country will be assigned to a unique numeric index. Then the transforming will apply the numeric indexes into the corresponding column.

```

df = indexer_country.fit(df).transform(df)
df = indexer_market_segment.fit(df).transform(df)
df = indexer_distribution_channel.fit(df).transform(df)
df = indexer_reserved_room_type.fit(df).transform(df)
df = indexer_assigned_room_type.fit(df).transform(df)
df = indexer_customer_type.fit(df).transform(df)

```

Then, the converted feature will be as a new column in the data frame. So, we deleted the original column and renamed the indexed feature to the original name such as country index will be renamed to country after dropping the original one.

```

df = df.drop('country', 'market_segment', 'distribution_channel', 'reserved_room_type', 'assigned_room_type', 'customer_type')
df = df.withColumnRenamed('country_index', 'country')
df = df.withColumnRenamed('market_segment_index', 'market_segment')
df = df.withColumnRenamed('distribution_channel_index', 'distribution_channel')
df = df.withColumnRenamed('reserved_room_type_index', 'reserved_room_type')
df = df.withColumnRenamed('assigned_room_type_index', 'assigned_room_type')
df = df.withColumnRenamed('customer_type_index', 'customer_type')
df.show()

```

For Agent:

We used a random forest classifier to classify each null agent value to its proper value. We switched to pandas for this case only.

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split

df = df.toPandas()
df = df.replace('NULL' , np.nan)

df_missing = df[df['agent'].isnull()]
df_non_missing = df.dropna(subset=['agent'])

X_non_missing = df_non_missing.drop(['agent'] , axis = 1)
y_non_missing = df_non_missing['agent']
X_missing = df_missing.drop(['agent'] , axis = 1)

model = RandomForestClassifier()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_non_missing, y_non_missing, test_size=0.2, random_state=0)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

f1 = f1_score(y_test, y_pred, average = 'macro')
print(f"R^2 Score: {f1}")

```

The F1-score gave a score equal to 42.6% which is too bad to count on it to fill the nulls. So, after deep thinking, we decided the following, seems that the classifier could not predict the exact number of the agent ID. Furthermore, we tried also regressor model but ended up with predicting new agents IDs that do not exist.

F1-Score: 0.43415537038407676

Even though the performance is too low, it is not proper because each ID represents a company or an organization. In addition, filling in any value that might be wrong is not acceptable as well because it will be illegal to fill the data on behalf of a company that did not commit to the booking. So, the NAN instances will be assigned as number 0 which would indicate as "unknown" and there is no agent that has that ID.

```

df = spark.createDataFrame(df)
df = df.withColumn('agent', when(isnan('agent'), lit(0)).otherwise(col('agent')))

```

Closure: All nulls and inconsistencies are now handled, and the features converted into numerical values which now the modelling phase will start. The time taken for Preprocessing Section to be completed as shown:

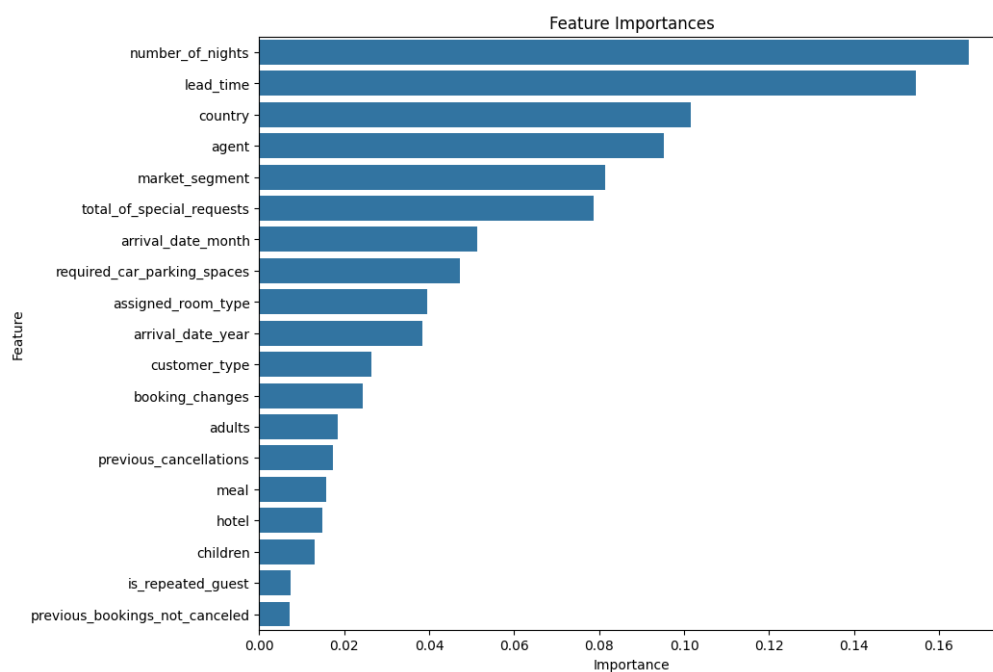
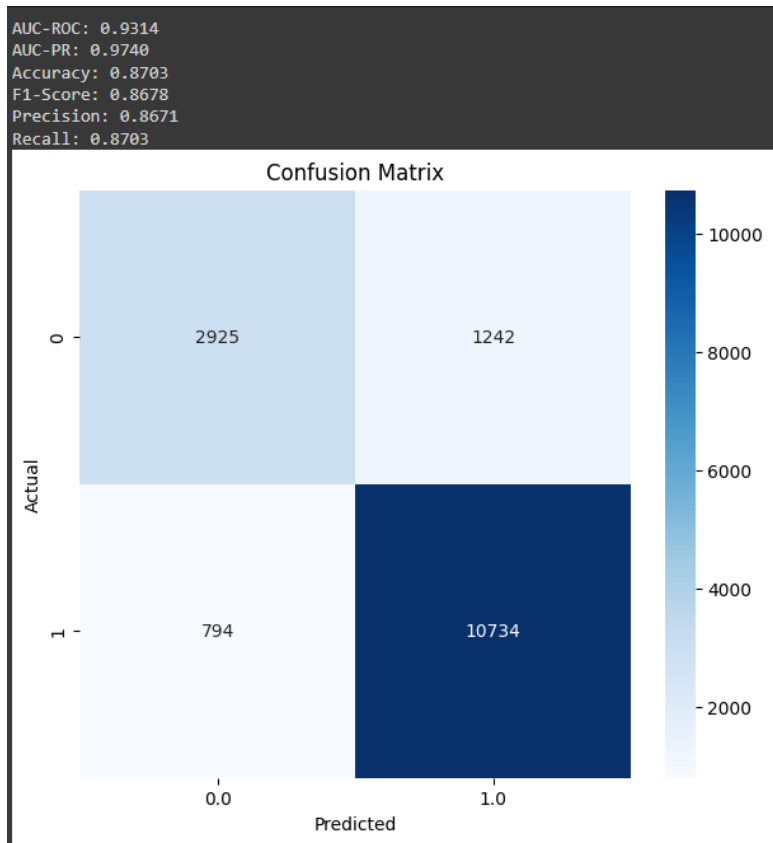
```
Time: 132.20114469528198
CPU: 83.9
RAM: 78.6
```

Section Seven: Modelling Section:

In modelling phase, we used Random Forest Classifier since it is robust against outliers and ensemble model that is enhanced to deal with classification problems. First, defining x that takes the features that will help to predict y which is the target.

X is all features excluding: the target, season, arrival date day of month because the arrival month will be enough. reserved room type because in the EDA it was shown that the assigned room is not an influencer on the target. So, we only took the assigned room type, days in waiting list and babies were shown in the feature importance function that they were not important in the prediction phase (that was shown after multiple experiments). Also, deposit type since nearly all the data has one type which is no deposit. Lastly, distribution channel because the market segment is about the same as the distribution channel with slight differences, so removing it will reduce the model complexity and prevent multicollinearity.

Then, initializing the model that has the classifier with 35 decision trees with a max depth of 20. These are hyperparameters that try to enhance the model to increase or decrease its complexity to give better performance. Starting with splitting the data to 80% for training and 20% for testing. Then, fitting the model with the training set to learn patterns. Lastly start predicting the test set. To measure the performance, we used the evaluation metrics of AUC, ROC, Accuracy, F1-Score, Recall and Precision. The results were reasonable, giving an overall performance of 86%. Also plotting the confusion matrix which shows that the model's mistakes are in the minority class of the data which is the cancellation reservation status.



Feature importance is a method to detect each feature's weight for the model and how much the model depends on these features. As shown, number of nights and lead time are the features that

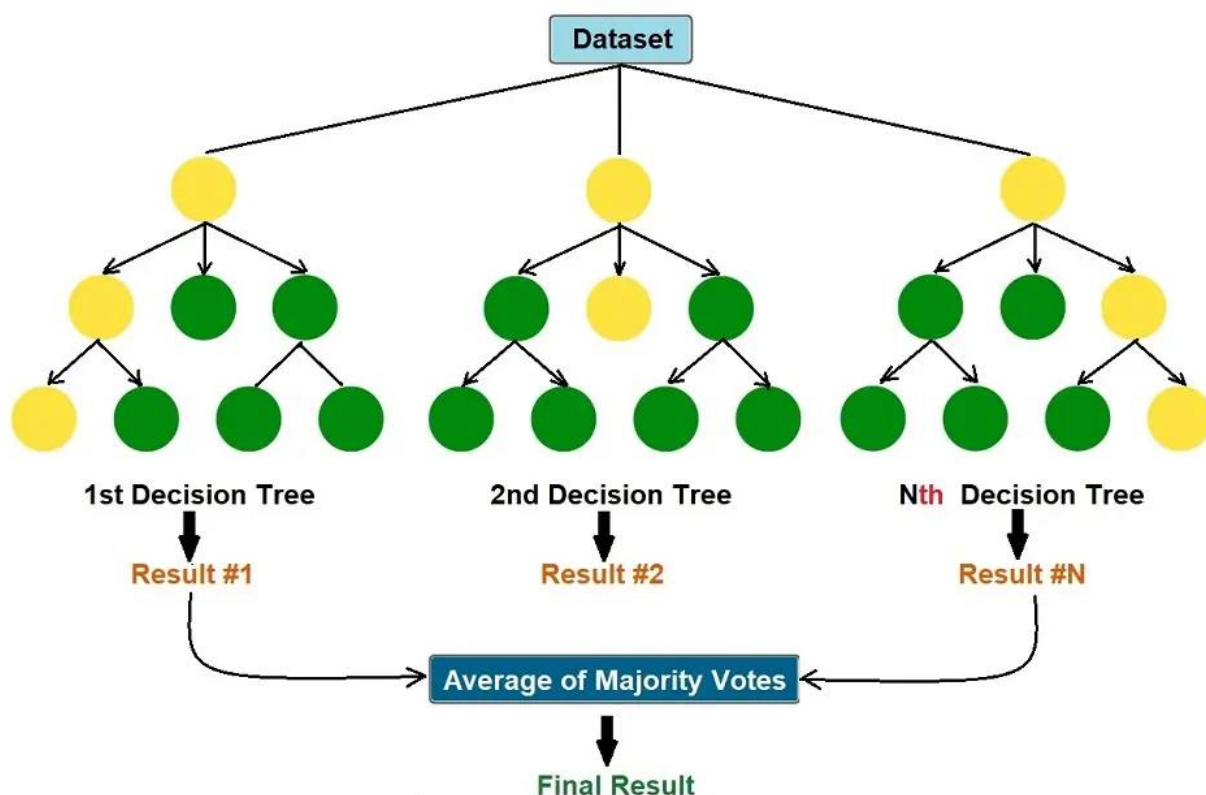
had the highest weight that the model counted on. It is a good technique to detect the model's performance through the features that it counts on. For example, sometimes a model can give a 100% performance which could be because of a particular feature that causes overfitting.

Explanation of the model Random Forest:

Definition: An ensemble tree-based algorithm, referred to be a better version of the model decision tree that works by conditions to give predictions.

Approach: take data subsets with boundaries according to the subset taken and work by creating a tree based that has conditions to vote for the final answer which will be according to all the decisions trees vote by taking the majority.

Advantages: It is proper for our task due to its immunity against noise data. In addition to its strength against complexity of the model.



The whole dataset is divided into subsets of data. then boundaries are created for the specific subset built on a tree of conditions based. the majority vote of these decision trees will be the final answer.

Conclusion: The time taken for the modelling section to be completed is as shown:

```
Time: 147.1330873966217
CPU: 94.2
RAM: 27.2
```

That's it for the technical report, where everything was done is covered step by step. Lastly, the time taken for all the code to be completed is:

```
Time: 400.98433899879456
```

The size of the code notebook is 1.13 Megabyte.

Q4: Advantages and Disadvantages of Data Preparation & Sourcing:

Advantage:

Ease To Access: we gathered the data from Kaggle which is a website that offers a variety of datasets in many fields with different formats such as CSV and JSON files that we used in the given problem. Just by typing Hotels in the search bar in Kaggle, we could find what we wanted.

Disadvantage:

Difficulties Finding Relevant Json Files: searching up for a specific field with CSV/Excel format is easy to find where data scientists handle these formats all the time. However, searching up for a specific field with JSON format is not an easy task to do. In the beginning, we found multiple JSON files that were talking about hotels, but we could not find dataset that is related to the cancellation of reservations.

Advantage:

We dealt with different sources of data (CSV and JSON) which provided more evidence and credibility to the insights that we extract because many sources of data agree on a specific insight

which force stakeholders to believe the impact of these features and attributes which also will eventually lead to better outcomes such as increasing the profit and reducing the risks and losses.

Disadvantage:

Even though we benefited from the JSON files that we used, it was difficult to detect the needed columns and attributes. Also, the way of work that we should follow is to try and extract useful information from JSON datasets to support our theories and assumptions that occurred in the CSV file. The intention was using the CSV dataset as the main target, while using JSON files as supporting datasets to help aiding the theories we conducted to make our story more believable and enhanced with multiple evidence from different data sources which leave stakeholders with no doubts and hesitation.

General Comparison:

In data preparation, starting with exploring the data, we found duplicated instances using pandas. The problem was with PySpark that it has different methods for calculating that matter. For example, pandas show that there are about 32k duplicated instances, while Pyspark shows about 8k duplicated instances which also would differ if used another method unlike pandas that it's built in functions are easy to understand and meet data scientists' desires. On the other hand, in the preprocessing section, there was no difference in doing the preprocessing techniques that were used in pandas or pyspark, but it depends on the implementer himself/herself. Personally, I was trained more in pandas which made me take some time to understand Pyspark syntaxes. Lastly, in exploratory data analysis specifically in visualizations, it was handled by pandas which was easy and fast since we were using great libraries that are widely known for their simplicity such as seaborn. Also,

Advantage:

In the collected datasets, the null values were not a problem since there are many features that could help to understand what the best filling for that specific value was. In addition to the relationships that features have in common, which was needed more thinking to understand the hidden patterns.

Disadvantage:

There was a problem where data should be 119390 rows which is considered a huge volume, but due to the duplication instances and the dropping for these instances, the rows dropped to 79225 rows which is a drop rate equals to 33.6%.

Ultimately, data sourcing and preparation was a great challenge but will be worth it eventually to understand the business events by understanding all the hotel's features that may affect our target. The second we understand, we would increase the revenue by eliminating underutilization from happening.

Q3: Applying Hadoop & Evaluating Between Hadoop and PySpark:

First, we will discuss the case scenario that we have, then an overview of Hadoop and Spark, and evaluating the differences between the two in terms of performance on the given scenario.

The Scenario:

The given scenario is that we have datasets of hotel reservations with huge amounts of data that requires time and a lot of effort to deal with it. The data size was approximately 17 megabytes of 119390 rows and 32 columns. It is batched-structured data which means that the data is historical that can be received to the model in periods and batches. In another meaning, multiple instances will be received at once as a batch for a duration of time such as receiving the batch each month where it could be thousands of rows. In addition, structured which means the data can be stored in rows and columns and are easy to analyze and handling it.

The task is to build an end-to-end data science project which starts with reading, handling, analyzing, preprocessing and modelling the data to get results and forecasting. In this case, the time and resources needed to accomplish the task would be high which requires powerful tools for the matter.

Hadoop Overview:

Hadoop is an open-source framework to handle big data tasks effectively by using a parallelization approach. It consists of three main components which are Hadoop Distributed File (HDFS) System, Map Reduce and Yet Another Resource Negotiator (Yarn). It is an ecosystem that handles different types of files without the necessity of having schema for the data.

HDFS:

It is the file manager for Hadoop, it consists of a big node that is divided into two nodes:

1. Name Node: That stores the meta data for all files and the directory structure.
2. Data Node: That physically stores the data inside them.

In the data node, the node is divided into multiple blocks where the data is distributed among these blocks. If any block is corrupted, there is a mechanism called replication factor which is a value of the number of backup blocks for the main block in case of corruption happens to the block. The replication location is chosen to maximize the redundancy which is increasing the backup blocks as many as possible while minimizing the write bandwidth which is decreasing the distance between the main block and its backup blocks. In this case, the Name node also knows about the replicated blocks. However, if the name node is corrupted, the data will be lost even if it is still stored physically. Which is a sign of the necessity to use backing up mechanisms of FSImage and Edits which stores a snapshot of filesystem metadata at a certain time to store the status and edits where any changes on the data is applied on.

Map Reduce:

As mentioned about the data shape, they are stored in blocks inside the data node. Map Reduce is three functions:

Map: takes the input data from the blocks and divide them into smaller chunks to transform them to make them as key and value pairs structure. Then send the output to the Sort/Merge function.

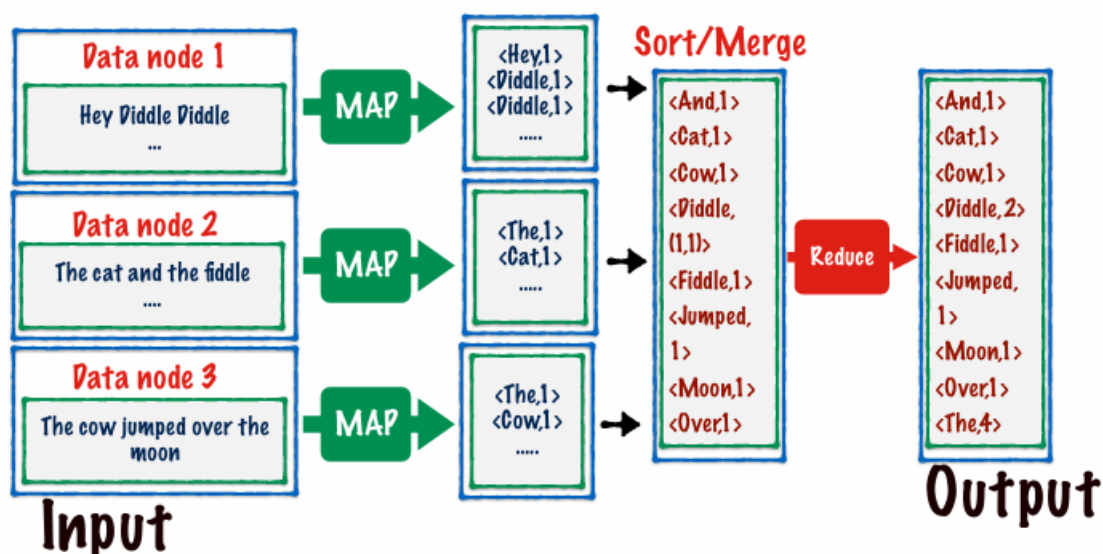
Sort/Merge: takes the output from the Map function and sort these pairs alphabetically then merge the similar keys into a key and list of values structure. Then send the output to the last function which is Reduce.

Reduce: takes the output from the Sort/Merge function then applies a specific operation.

Example:

Textual data is stored in blocks, the mapper takes the data and transforms each word to a key value pairs structure. Then the sorter and merger will take the output and sort the pairs alphabetically for example and merge the similar key value pairs into key and list of values pairs

structure. Lastly, the reducer takes the output and performs an operation such as calculating the frequency of each word by calculating the values of each key.



YARN:

It provides two services which are resource manager and node manager:

Resource Manager: Each cluster has a resource manager and can be considered as it is working on the name node. It is responsible to give resources for each node to ensure their functionality for executions of the tasks. It optimizes the resources for each node according to scheduling policies.

Node Manager: Works on the data nodes where it monitors the state of the nodes and the resources given to the node and reports it to the resource manager to ensure everything is running smoothly.

When a task is triggered, a container is made which is considered as an encapsulation of resources for the specific task that the task is using. Also, the application master process that is in the node manager inside the container starts. Its job is to request resources from the resource manager.

The policies of scheduling are FIFO which is first in first out operation that all resources will be given for the first process and start the next after the first finishes. It is rarely used because of the huge amount of waiting time. Another scheduling is capacity scheduling which is dividing the

resources into queues that send tasks bigger tasks to the bigger queue and the smaller tasks to the smaller queue. It causes underutilization because the queues may be over for the task and should not take all of that. Lastly is fair scheduling where resources are split equally for all tasks, which causes no waiting time.

Differences Between Hadoop & Spark:

Now, after understanding what Hadoop and the components of it are, we can delve into the differences between it and spark in terms of performance and what might happen if Hadoop was applied in the scenario. All information is from reference 5 [5].

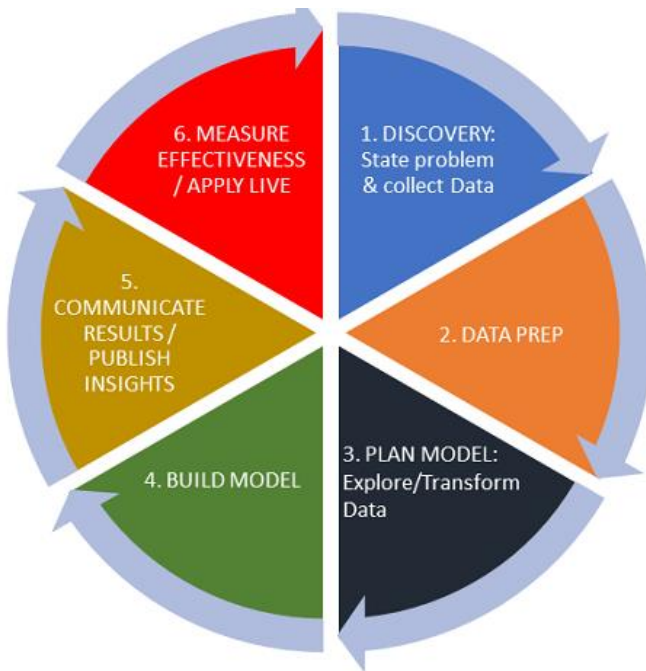
Difference	Hadoop	Spark
Speed	MapReduce functions read and write from a disk which cause slower processing speed.	Reduce read and write cycles and stores intermediate data in the memory which resulting in faster processing speed.
Usage	Better handling for batched data.	Better handling for streaming data.
Latency	High latency since it takes longer time to process data.	Low latency since it takes less time processing data.
Interactive Mode	No support for streaming data processes which means lacking in interactives.	Yes, support for streaming data processes which means processing interactively.
Data	Developers process data in batches.	Developers can process in batches and streaming data.
Cost	Cheap.	Expensive due to high RAM required.
Algorithm Used	PageRank.	Graph computation.
Fault Tolerance	Uses replication data blocks so whenever a corruption	Has chain of transformations which means whenever

	happens the tasks can be sent into another blocks.	corruption happens the chain of transformations can be recomputed on the original data.
Security	Extreme secure by supporting SLAs, LDAP, ACLs.	Lack of secure.
Machine Learning	Data fragments are large which creates bottlenecks which cause slowness in machine learning processes.	Has in memory processing and uses machine learning libraries thus faster in machine learning processes.

In summary, spark will be faster than Hadoop in processing the task. That is due to the system of Hadoop that requires much time to finish tasks such as the data science project. For example, preprocessing and exploratory data analysis requires high computational requirements, since spark uses in memory processing, it will be faster than Hadoop that read and write from the disk which is slower than the RAM. In addition, the spark can have interactive EDA which performs the visuals, for example much faster than Hadoop that must wait for map reduce functions to finish to perform the task. Also, in the machine learning section, spark would absolutely outperform Hadoop because spark contains built in libraries for machine learning. In the other hand, Hadoop requires help from third party systems such as Mahout which is a member of Hadoop ecosystem that provides machine learning libraries for Hadoop [6], which in this case it would be more complex to the pipeline of the machine learning since calling out the third parties would increase the overhead and slower performance. However, since our data is not truly very big and matches all standards of big data, Hadoop still has advantages on batched data types while Spark is more specialized and focused on streaming data types. But also, for the given case, spark would perform better than Hadoop for this specific case scenario since we want speed for manageable size of data. Lastly, the cost of applying Hadoop is cheaper due to the dependence on disks which are cheaper than RAM that Spark uses.

Q5: Big Data Life Cycle:

The life cycle of big data consists of multiple stages which all discussed previously in the step-by-step coding. But for more generalization, better understanding and summarization [3]:



1. Discovery Of Problem & Data Collection:

Explanation: Identify and state a problem and understand that needs to be taken into consideration by understanding what the problem is, why it matters, what needs to be solved and how. In addition, collecting the needed data for that problem will help to solve the problem stated.

Our Scenario: Our problem is the underutilization of hotel rooms because of the cancellations of reservations which leads to losing revenue and causing empty rooms without benefitting from them. The main target is how we might increase the revenue by reducing cancellations from occurring to ensure rooms are not empty to get maximum utilization. We need to collect hotels reservations data from multiple sources to understand the patterns and customers behaviors that affect the target which is the cancellation of reservations. In addition to get insights from different sources to prove all extracted insights.

2. Data Preparation:

Explanation: Consists of steps after collecting the data which are processing and cleaning the data which is a crucial part where most of the time is spent on processing and analyzing the data. Preparing the data means making it completely ready to be inserted into the predictive modellings.

Our Scenario: First general insights about the data in terms of shape, duplications instances, null values. inconsistencies detection and dimensionality reduction. We detected those and processed them with proper techniques with justification for every action we have done. We deleted the duplications, filled the null values in the best way possible. Also, handling some inconsistencies such as increasing one night for every check out happened for a valid reason as mentioned above. In addition, dropping unwanted features and encode categorical features.

3. Plan Model:

Explanation: Create exploratory data analysis to understand the data deeply and apply loading mechanisms for the data such as ETL which extract data, transform it then load it.

Our Scenario: Creating visualizations and plots to understand the data and the correlations between features. In addition, understanding the biggest influences that affect the reservations status. In the visualization section, we showed great insights such as requesting a car park space will grant a 0% probability of cancellation of the reservation according to the data. Therefore, discovering hidden patterns and complex relations. The preferred model type for pyspark and this problem is Extract, transform and load since they can be sent in batches and are structured data [4].

4. Build A Model:

Explanation: Building a predictive model for future forecasting which is helpful for organizations to understand upcoming potential events to take them into consideration. Model selection should be according to the problem which usually requires a domain expert to determine the proper model for the task.

Our Scenario: Building a random forest classifier algorithm for predicting binary classification problem whether the reservation would be successfully done or got cancelled. For our task, it

gave an overall 86% performance which is relatively great but still has some mistakes in predicting the minority class. In addition, the results may be overfitting since there is no external dataset tested on the model yet.

5. Communicating Results & Insights:

Explanation: It is the process of communicating with stakeholders about the insights and results figured which involves all the important information and insights that may affect the decision making. Providing the proper insights and findings will result in producing better decisions. Thus, better outcomes.

Our Scenario: We have made a step-by-step explanation for every work was done. In addition, explanation and interpretation of findings from the visualization and plots which shown features that has influences on the target which is important for stakeholders to know that to build up a decision that will help achieve the goal of the problem statement.

6. Measure effectiveness & Application:

Explanation: This step involves applying the recommendations and decisions to measure its validity and effectiveness and analyze the success of all the previous stages whether it came up with great outcomes or failed at some point.

Our Scenario:

Applying the recommendations that will be listed below may help us to achieve the best outcomes.

Recommendations Based On Findings:

1. Incorporate features that are related to the prices since it shown a sign of influence on the target.
2. Make part of rooms reservations as non-refundable that forces guests to pay in advance.
3. Use the waiting list of guests as plan B for cancelled reservations. This would fill the rooms with guests from the waiting list. In addition, give a discount for the waiting list to prevent as much loss as possible.
4. Offer a discount for people who are willing to ask for a car parking space. This would encourage those people to book, which is a guaranteed proper status of no cancellation.

5. Avoid discounts in Summer semester since every summer the bookings are relatively higher than any other semester which indicates that people are willing to book regardless of prices.
6. Ask bookers for special requests from them to ensure their commitment and fully preparation for not cancelling.
7. Since reservations are lower at the end of month days, making slight discounts for normal room types to prevent higher quality room types that for sure use more electricity and have higher features. This way, there would be reservations at the end of month days for lower quality rooms that do not have bigger features that consume the hotel's resources.

These stages of the big data life cycle are important for all organizations since they provide insights into their problems. Thus, evidence and discovering hidden patterns. Thus, better planning for decisions. Thus, better outcomes. All will be for the best for the organization, which leads to higher revenue and lower the chances of falling into risks or high profit losses. For example, an organization has multiple branches of selling products. Their organization is in a setback where they are losing profits and facing risks. Without the aid of data science, it may be impossible to understand the events and the story of the organization and why everything is happening. On the other hand, with the aid of data science, it would discover the organization's facts and leads to better understanding of what is happening. For example, with the use of exploratory data analysis, the problem could be because of specific products that cost them too much without selling any of them. Or a branch that is getting funded with a lot of resources without making the minimal profit needed to fund itself. These examples are obviously simple, that's why data life cycle is important to discover hidden patterns and complex relationships to save organizations from falling into setbacks by avoiding risks or bigger losses. In addition to increasing profits and better outcomes due to better decisions.

Q6: Hadoop & Pyspark, Hive & Spark:

Before more delving into Hadoop and spark comparison, let's talk about hive and spark components and the architecture of each one of them:

Hive:

Apache Hive is an open-source warehousing tool for accomplishing distributed processing and processes related to data analysis. It uses hive query language which is a programming language like SQL. This language is being proceeded by MapReduce programs. It is used for structured and semi structured data by replacing the complex java codes of MapReduce with Hive queries. Hive is good for analyzing huge datasets and data encapsulation.

Architecture & Components of Hive [7] [8]:

It is made of four main layers where each layer serves a purpose:

1. Hive Client

Consists of three components,

- Thrift client which serves the request from all clients that their programming languages support thrift.
- JDBC client that is used to create a connection between hive and java applications.
- ODBC client which allows all applications that have open database connectivity protocol to be able to connect to hive.

2. Hive Servers:

The servers that hive provides:

- Hive CLI: command line interface which is a shell that executes hive commands.
- Hive Web User Interface: web-based graphical user interface for executing hive commands.
- Hive MetaStore: a map of all data stored in hive's warehouses with its metadata.

- Hive Server: referred to Apache thrift client which accepts requests from clients and sends it to hive driver.
- Hive Driver: receives queries from sources like CLI, web UI and thrift to transfer these queries to the compiler.
- Hive Compiler: converts received queries to MapReduce jobs that perform the processes.
- Hive Execution Engine: execute the MapReduce jobs that have done by the compiler.

3. Processing & Resource Management:

Hive uses MapReduce functions for executing the received queries.

4. Distributed Storage:

Storing huge amounts of data using Hadoop Distributed File System.

Example:

Hive Metadata stores all tables, schemas and data locations and monitors them. So, in the first, a client asks for a request using command line or using the hive web user interface. Next, the Hive Server handles the request incoming from clients and sends them to the Hive Driver. After that, the Hive Driver manages the request and passes it to the proper processing components of the Hive Compiler that converts the queries of the requests to MapReduce Jobs. Lastly, Hive Execution, which executes the request asked to retrieve the data from the Hadoop distributed file system.

Apache Spark:

It is an open source that uses distributed computing systems to process big data applications. As mentioned before, it's characterized by its speed due to the usage of RAM. In addition to its capabilities to handle streaming real time data due to in memory processing technique which has low latency. Spark uses the master-slave mechanism which states that a master node distributes the work among the worker nodes (slaves).

Architecture & Components of Spark [9] [10]:

1. Spark Driver:

Controls the execution of the applications by planning how to execute by telling the executors what to do. The driver manages how the data will be processed by breaking down the job into smaller tasks and keeping track of the running jobs.

2. Spark Executors:

The nodes that apply the jobs and execution where each node has its own resources and memory). After finishing the execution, they report back to the driver for the results. It has multiple modes for execution:

- Cluster mode: applications run on a cluster (multiple nodes). The driver will run in one of the workers (slave) nodes and the other nodes will execute the task.
- Cluster client: applications run on the device that requested the task. The driver will be within that device and the worker nodes will be from a cluster.

3. Cluster Manager: everything will be executed from the local device.

Allocating resources across all nodes to ensure they have enough resources to do the assigned tasks. Example for a cluster manager is YARN in Hadoop ecosystem (Yarn was explained in question 3).

The architecture consists of two abstractions layers:

1. Resilient Distributed Datasets:

Collections of data items that are stored in memory on the worker nodes. Resilient means restoring data on failure, distributed means data is distributed among the nodes and datasets means a collection of data. Its main job is to keep track of the data processing which grants it the ability to recover lost data if an error occurs. In another meaning, it has fault tolerance.

2. Directed Acyclic Graph:

Diagram that manages the flow of how the data will be processed and how all tasks are connected. Its job is whenever a task is triggered, the spark driver transforms the task into a directed acyclic graph to make spark able to manage the execution of the task by understanding how the task parts are dependent on each other.

Components of Spark [10] [11]:

- Spark Core: It is the engine of spark that deals with the fundamental tasks that are essential for spark to function properly. For example, uses in memory computations that speed up the processing functionality. In addition, for the usage of parallel mechanisms which distribute the work among multiple nodes in a cluster which make it able to handle big data tasks properly. Also, for having task scheduling and the fault tolerance feature.
- Spark SQL: a component that helps users to handle structured data. also provides the ability to perform interactive analytical applications for both streaming and historical batched data.
- Spark Streaming: provides the ability to handle and process real time data.
- MLlib machine learning: provides the ability to use machine learning models such as clustering, regression, and classification because spark has built in functions for that matter.
- GraphX graph processing: provides the ability to analyze graphs data and implement social network analysis and graph mining which is beneficial for identifying relationships in the data.

Now, let us talk about the advantages and disadvantages of using Hadoop and Apache Spark for decision-making. Some differences were already handled in Question 3.

Spark Advantages:

1. Great for interactive processing such as real time data that is because of the mechanism which is working in-memory that give faster results and computations.
2. Flexible and powerful because spark supports multiple programming languages which make users use their preferred language [14]. In addition, the many features that spark provides such as SparkSQL, GraphX and built in machine learning libraries.

3. Integration feature which allows spark to work alongside Hadoop ecosystems components.

Spark Disadvantages:

1. Consumes a lot of memory because spark depends on in-memory resources which consumes high resources especially for tasks that are huge that demands high computational speed and resources. In addition, it is expensive since the main component is the random-access memory.
2. Have issues for small files. Each file has metadata, with many small files it will cause overhead to many reasons such as [13]:
 - a. Increase memory pressure on executors where each process has few records per file.
 - b. Large number of metadata getting stored in memory.

Hadoop Advantages:

1. Cost effective: unlike spark, Hadoop depends on disks that are cheaper than random-access memory.
2. Applies penalization mechanism which is working on multiple tasks concurrently which results in faster results.
3. Data diversity: Hadoop is great for organizations that have multiple data formats such as structured, semi structured, and unstructured. Which is flexible to handle big data applications.
4. Replications factor which is creating backups for data blocks which save the data from corruption.

Hadoop Disadvantages:

1. Not efficient for small datasets due to the overhead of setting up the environment of clusters which may outweigh the benefits of Hadoop.
2. Vulnerability: Hadoop framework is written in Java which is the most popular and used programming language which makes it easy to be targeted for cyber-attacks [15].

3. Only supporting batched data arrival types [15].

[12] Both technologies are well suited for companies and organizations, while both have their advantages and challenges, the use case will determine what organizations should pick. For example, for data warehousing, it is preferred to use Hadoop since it has Hadoop distributed file system that stores massive amounts of data regardless of the type whether structured or unstructured. Then, the organization can integrate Hive tools to use queries and analyze the data efficiently. In the other hand, spark is better for cases that require fast data processing because as we know that spark uses in memory processing which is very fast for computing. In addition, spark is well suited for streaming data such as real time data of internet of things. In addition, spark is used widely in machine learning models because it has built-in algorithms.

Ultimately, choosing between one of the two will depend on how the organization wants to increase its revenue and prevent losses. in addition to aid them in making better decisions.

If the organization wants to build decisions for future cases of storing and processing massive amounts of data at a lower cost, then they should choose Hadoop because it is well-suited batched data and warehousing tasks. In addition, Hadoop follows the Extract-Transform-Load model plan.

If the organization wants to build decisions for future cases of real-time processing and analysis and streaming data types, then they should choose Spark because it is well-suited for data streaming, machine learning and interactive data analysis cases. In addition, Spark follows the Extract-Load-Transform model plan.

References:

1. Yam Chhetri (2024). *Most Visited Cities in the World 2024: Statistics - WP Travel*.
2. Mozilla (2022). *Working with JSON*. [online] MDN Web Docs.
3. Glen, S. (2019). *The Lifecycle of Data - DataScienceCentral.com*.
4. <https://www.datacamp.com/blog/etl-vs-elt>
5. GeeksforGeeks. (2020). *Difference Between Hadoop and Spark*.
6. taha, A. (2024). *Hadoop vs Mahout and Machine learning Issue?* [online] Stack Overflow.
7. www.javatpoint.com. (n.d.). *Hive Architecture - Javatpoint*.
8. <https://data-flair.training/blogs/apache-hive-architecture/#:~:text=The%20major%20components%20of%20Apache%20Hive%20are%20the%20Hive%20clients,Hive%20query%20to%20the%20compiler.>
9. Simplilearn (2022). *A Detailed Guide Into Apache Spark Architecture*.
10. ayushjoshi599 (2020). *Components of Apache Spark*.
11. *Components of Apache Spark | Apache Spark Tutorial*
12. proxify.io. (n.d.). *Hadoop vs Spark: A comprehensive guide to big data processing - Proxify*.
13. Siraj (2023). *Small File, Large Impact — Addressing the Small File Issue in Spark*.
14. Apache.org. (2019). *Overview - Spark 2.4.4 Documentation*.
15. GeeksforGeeks. (2020). *Hadoop - Pros and Cons*.