Hotel Bookings Cancellation Forecasting

Numitha Devi Oguri
Department of
Computer Science and Engineering
University Of North Texas
Denton, Texas, USA
Email: numithadevioguri@my.unt.edu

Ketha Tirumuru
Department of
Computer Science and Engineering
University Of North Texas
Denton, Texas, USA
Email: kethatirumuru@my.unt.edu

Sheela Pabbareddy
Department of
Computer Science and Engineering
University Of North Texas
Denton, Texas, USA
Email: Sheelapabbareddy@my.unt.edu

Yamuna yennem
Department of
Computer Science and Engineering
University Of North Texas
Denton, Texas, USA
Email:Yamunayennem@my.unt.edu

1. Introduction

1.1. Motivation:

The cancellation of hotel reservations places a substantial burden on the income and operational planning of the hospitality sector. In addition to improving the income stream by reducing the impact of cancellations, this also improves the guest experience by enabling hotels to dynamically modify their goods and services considering foreseen data. By constantly altering offerings based on predictive data, creating a predictive model for reservation cancellations can reduce financial consequences and simplify adaptive customer relationship management.

By constantly altering offerings based on predictive data, creating a predictive model for reservation cancellations can reduce financial consequences and simplify adaptive customer relationship management.

1.2. Significance:

1.2.1. Financial Impact: . By lowering the cost of cancellations, hotels are able to increase their overall financial stability and income streams.

1.2.2. Guest Experience: . By proactively handling cancellations, services may be dynamically adjusted, improving the overall happiness and experience of the guest.

1.2.3. Operational Efficiency: . Adaptive management techniques are made possible by predictive models, which simplify resource allocation and operational planning.

1.3. Objectives:

• Understanding and exploring the dataset. • Performing EDA and data- preprocessing. • Customizing Marketing Plans Depending on the Source of Visitors by determining the most common countries of origin for visitors who choose to either maintain or cancel their bookings. • Analyzing cancellation trends unique to various hotel categories (city, resort etc.), this analysis seeks to shed light on the best ways to deploy resources for each one. • Determining how visits are distributed among various market sectors and the booking results that correspond with them. • Identifying trends and patterns in the cancelation rates across several years. • Finding Peak Visitor Periods to pinpoint and measure the month that sees the largest influx of tourists. • Examine the relationship that exists between the deposit types and the probability of cancellations. • Assessing the percentage distribution of meal reservations to determine the frequency of various meal options among visitors. • Calculating and examining how lead time variations affect the probability of cancellations. • Examining the variations in nightly prices across the year's several seasons. • Determining the trends and patterns in the percentage of cancellations for each kind of hotel during the various seasons. • Build ML models and training with dataset. • Comparing Accuracy, scores among various ML models.

1.4. Features:

To effectively forecast cancellations, the model will take into account a number of features, including:

Details of the reservation: dates, number of visitors, duration of stay, and other pertinent information. Booking requirements: A variety of elements, including deposit type and reservation technique. Probability Score: A probability score representing each reservation's likelihood of cancellation will be the result. The F1 score is used as the evaluation metric because it strikes the right mix between precision and recall, which is essential when false positives and false negatives have significant financial repercussions.

Performances of the developed models will be compared to the baseline to see if there is any discernible improvement in the selected metric.

1.5. Related Work

1.5.1. Proposed Methods:. Utilise ML models like Decision Trees and Random Forest followed by taking into account their potency in handling categorical data and resilience to outliers.

This project sets out to create a predictive framework in an effort to lessen the financial and operational effects of cancellations of hotel reservations. It goes through stages of thorough data preparation, strong model construction, and tactical deployment. To ensure a perfect dataset suitable for model training, missing values, outliers, and categorical data are first managed during the data processing step. We then enter a model development stage, which entails not only building predictive models but also meticulously tuning hyperparameters and ensuring model generalizability through rigorous cross-validation. Here, we employ models like Decision Trees and Random Forest, renowned for their interpretability and efficacy in various data scenarios.

1.5.2. Evaluation Metric:. The F1 score is used as the evaluation metric because it strikes the right mix between precision and recall, which is essential when false positives and false negatives have significant financial repercussions.

Performances of the developed models will be compared to the baseline to see if there is any discernible improvement in the selected metric.

2. Dataset

2.1. Dataset Description:

The dataset contains necessary booking information, such as the dates of the reservation, the number of guests, the duration of the stay, the kind of deposit, and the method of the reservation (such as direct or through a travel agency or operator).

We have selected our dataset from Kaggle.

Link: https://www.kaggle.com/code/sanjana08/hotel-booking-cancellation-prediction/input

We have taken a dataset to check hotel demand data. A resort hotel (H1) and a city hotel (H2) are the two types of hotels are considered in this dataset. The structure of the dataset has 32 variables accounting for the 119210 observations. Every observation is a hotel reservation. Dataset has reservations that were scheduled to arrive between July 1, 2015, and August 31, 2017, including reservations that actually arrived and reservations that were cancelled. Since this is actual hotel data, all information related to guest or hotel identification has been removed. These dataset can play a significant role in revenue research and teaching due to the dearth of genuine company data for scientific and pedagogical reasons.

• **Hotel:** Indicates the kind of hotel (such as a resort or city hotel).

• **Is Cancelled:** denotes the status of the booking—zero (0) or one (1).

• **Lead Time:** the period of days from the date of booking to the date of arrival.

• **Arrival Date:** Contains the day of the month, week number, year, and month.

• **Length of Stay:** Total number of weeknights and weekend nights spent.

• **Information on the guests:** Total number of adults, kids, and infants.

• **Meal Type:** The type of meal that was reserved e.g., Half Board, Bed and Breakfast.Country: Country of origin of the guests.

• **Market Segment and Distribution Channel:** Direct, Corporate, and other market segments are examples of the distribution channels used by Booking.

• **Repeated Guest:** This field denotes if a guest has been here before (1) or not (0).

• **Previous Cancellations:** The total number of times a visitor has cancelled a reservation.

• **Previous Bookings Not Cancelled:** The total number of prior reservations that the visitor did not cancel.

• **Assigned and Reserved Room Type:** Classification of rooms that have been both assigned and reserved.

• **Booking Changes:** Total amount of modifications made to the reservation.

• **Deposit Type:** The kind of deposit made (such as Refundable or No Deposit).

• **Agent/Company:** The identification of the travel agency or firm that booked the reservation.

• **Days on Waiting List:** The duration of time the reservation was placed on the waiting list.

• **Customer Type:** Describes the kind of customer (e.g., Contract, Transient).

• **Average Daily Rate (ADR):** The price each night on average.

• **Required Car Parking places:** The quantity of places needed for parking cars.

• **Total Special Requests:** The total amount of the visitor's special requests.

• **Reservation Status:** The present state of the reservation (such as Cancelled or Checked out).

• **Reservation Status Date:** The time the previous status was established.

This dataset, covers a variety of hotel booking-related topics such as guest demographics, booking specifics, and stay preferences, is well-organized and thorough. Customer behaviour, revenue management, and hotel management analysis may all benefit from this data.

2.2. Data Overview:

Dimensions: The dataset consists of 119,390 rows (entries) and 32 columns (features).

Type of Data:

• **Numerical Columns :** 'lead_time', 'stays_ in_ weekend_ nights', 'adults', 'children', 'babies', 'previous_cancellations', 'adr', etc.

• **Categorical Columns :** 'hotel', 'arrival_ date_month', 'meal', 'country', 'market_segment', 'deposit_type', 'customer_type', etc.

• **Mixed:** Some columns contain both numerical and categorical data, like 'agent' and 'company'.

Data Types of the attributes:

• The object data type makes up—12 columns—and is often used to represent texts or categorical data.

• The data type int64, which has 16 columns, represents integer values.

• The data type float64, which normally represents decimal numbers, is used in 4 columns.

2.3. Quantitative Variables:

• **lead_time:** The duration, which varies from 0 to 737 days, between the booking and arrival dates is around 104 days on average . • **date of arrival, week number:** From 1 to 53, the average week number of arrivals each year is around 27.17.

• **arrival_date_day _of _month:** From 1 to 31, the average day of the month of arrival is around 15.80.

• **stays_in_weekend_nights:** Visitors spend an average of 0.93 weekend nights here, however some remain for up to 19 nights.

• **stays_in_week _nights:** Visitors stay anywhere from 0 to 50 week nights, with an average of 2.5 week nights.

• **adults:** The range of adult reservations is 0 to 55, with an average of 1.86 adults per booking.

• **children:** There are typically 0 children per reservation, with a maximum of 10 children in some reservations.

• **babies:** With a maximum of 10 babies in certain reservations, the average number of babies per booking is 0 babies.

• **previous_cancellations:** Guests have cancelled 0.09 times on average in the past, with some having cancelled up to 26 times.

• **previous_bookings_not_canceled:** Guests have, on average, not cancelled 0.14 reservations in the past; some guests have up to 72 reservations that they haven't cancelled.

• **booking_changes:** Reservations are altered 0.22 times on average, and occasionally they are altered up to 21 times.

• **days_in_waiting _list:** Reservations are typically kept on the waiting list for 2.32 days, but occasionally they are kept on for 391 days.

• **Adr:** The average daily rate is around 101.83, with values ranging from 5400 to -6.38 (which may represent inaccuracies or exceptional situations).

• **required_car_parking_spaces:** Guests typically need 0.06 parking spaces, while some reservations call for up to 8 spots.

• **total_of _special _requests:** Guests typically make 0.57 special requests, while some have been known to make as many as five.

The Average Daily Rate, or ADR, has a minimum value of -6.38, which indicates negative rates. This observation raises an oddity.

To determine whether there was a data entry error or whether this was a unique instance, we would look into the records with negative ADR values. We would think about deleting or updating these items if it is shown to be a mistake.

adults: It appears that reservations without any adults are possible because the minimum value for the 'adults' feature is 0. we would review entries that show 0 adults to determine whether there was a data entry error or whether there were legitimate reasons why bookings without adults couldn't be made. We may choose to edit or clean these submissions based on the results.

children and babies: It appears that the maximum value of 10 for a single booking for "children" and "babies" is abnormally high. we would look at entries containing ten infants or children to see if there are any possible outliers or data input issues. To decide how best to address certain circumstances, further context or domain expertise could be required. In light of this, 23 would think about authenticating or cleansing these entries.

Categorical Variables:

hotel: Out of 119,390 entries, "City Hotel" is the most often occurring category with 79,330 occurrences. And other one is "Resort Hotel"

Cancelled: There are two distinct values: 0 for not cancelled and 1 for cancelled. With 75,166 occurrences, "0" (not cancelled) is the most frequent value.

arrival_date_year: Of the three distinct years seen, 2016 is the most frequent arrival year with 56,707 observations.

entry_date_month: There are twelve distinct months. With 13,877 observations, August is the most frequent month of arrival.

meal: Five distinct meal options are reserved. "BB" is the most prevalent kind, with 92,310 observations.

country: There are 178 distinct nations of origin. With 48,590 observations, "PRT" (Portugal) is the most often observed nation.

market_segment: There are eight distinct market segments. With 56,477 views, the section "Online TA" is the most often seen.

distribution_channel: Out of the five distinct booking distribution channels, "TA/TO" is the most commonly seen, with 97,870 observations.

is_repeated_guest: There are two distinct values: 0 for not repeated and 1 for repeated. The most common value, found in the majority of entries, is "0" (not repeated).

Types of reserved and allocated rooms: There are several kinds of rooms, some more common than others.

deposit_type: Deposits come in three different varieties. With 104,641 observations, "No Deposit" is the most prevalent kind.

agent: '9.0' is the most common agent, appearing 31,961 times, out of 334 distinct agents.

company: Of the 353 distinct firms, the most common is 'nan' (missing values), appearing 112,593 times. This shows a significant percentage of missing entries in the 'company' field.

reservation_status: There are three different possible reservation statuses. With 75,166 observations, "Check-Out" is the most prevalent status.

date of reservation_status: 926 distinct values are present, with '2015-10-21' being the most frequent with 1,461 occurrences.

3. DETAIL DESIGN OF FEATURES:

Creating a machine learning model that can reliably anticipate hotel booking cancellations while taking into account a variety of booking parameters is the main technological challenge. Finding out if a model like this can support proactive management practises in the hotel sector is the aim.

Basic Features:

Utilize existing columns such as lead_time, arrival_date_year, arrival_date_month, stays_in_weekend_nights, stays_in_week_nights, adults, children, babies, country, market _segment, previous_ cancellations, previous _bookings _not _canceled, required_ car_ parking spaces, total _of _special _requests, and adr.

3.1. Depending attributes:

• **Booking Duration:** The sum of stays_in_weeknights and stays_in_weeknights to determine the overall length of stay.

• **Cancellation Rate:** If a returning visitor, the cancellation rate in the past for that guest.

• **Seasonality attributes:** Using arrival_date_month and arrival_date_year as a basis, extract attributes such as peak season, holiday season, etc.

• **Change in Room kind:** Whether the kind of room that was reserved and the type that was allocated differed.

• **Encoding Categorical data:** Use methods like One-Hot Encoding or Label Encoding to convert categorical data into numerical representations, such as nation, market_segment, meal, and customer_type.

Managing Missing entries: In columns such as agent, firm, nation, and children, replace or omit missing entries.

Standardisation and normalisation are important steps in many machine learning models that involve bringing numerical features to a common scale.

To achieve standardisation in the context of cancelled hotel reservations, numerical characteristics are transformed to have a mean of 0 and a standard deviation of 1. As a result, all features are guaranteed to be on a same scale, avoiding the occurrence of specific variables that, because of their intrinsic scale disparities, would dominate the model training process. For example, standardisation would provide uniformity to the scale for features such as Average Daily Rate (ADR), enabling the model to train with the proper weight assigned to each feature.

In contrast, normalisation entails scaling numerical properties to a predetermined range, frequently between 0 and 1. This is especially important when making a hotel reservation because there may be a range of characteristics, such as the number of people, kids, and infants. Normalising these characteristics strengthens the model's resistance to the varying magnitudes, which enhances convergence when training. The 'number of children' feature, for instance, is normalised to guarantee that its influence on the cancellation forecast is commensurate with other normalised data.

3.2. Measures of evaluation:

• **Accuracy:** A measure of the model's overall accuracy.

• When it comes to forecasting cancellations, precision and recall are very crucial; we want to reduce the number of false positives and false negatives.

• **ROC-AUC:** To comprehend how the true positive rate and false positive rate are traded off.

• **Cross-validation:** To make sure the model is resilient and to avoid overfitting, use k-fold cross-validation.

4. ANALYSIS:

• **Data Collection:** Get the dataset of hotel reservations.

• **Preprocessing of Data: o Cleaning:** Take care of missing values and get rid of repetitions.

• **Feature Engineering:** o Modify category variables and produce new features.

o Standardization/Normalization: Align numerical attributes.

o Analysing exploratory data (EDA): Recognise relationships, find outliers, and comprehend the distribution of the data. See important data visually to learn more about booking and cancellation trends.

o Feature Choice: Determine which attributes are pertinent by using EDA and domain expertise. If required, reduce the number of dimensions (e.g., by applying PCA).

• **Model Construction:** o Divide the data into sets for testing and training.

o First, train your models (Random Forest, Decision Trees, and Logistic Regression).

o Determine which model is the best by evaluating its early performance.

• **Model Improvement:**

o Develop sophisticated models (such as Neural Networks and XGBoost).

o Analyse feature relevance and adjust hyperparameters.

o Use cross-validation to increase resilience

5. IMPLEMENTATION:

Data preparing: In accordance with the intended feature set, we will be cleaning and preparing the dataset. This entails fixing problems such cases with zero adults and negative Average Daily Rates (ADR) in order to prepare the dataset for efficient model training.

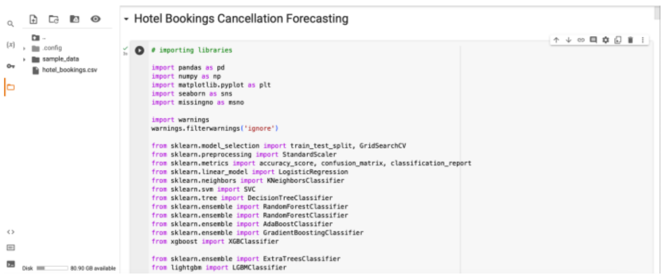
Model Validation and Training: Using the preprocessed dataset, we will train the chosen models. I'll carefully evaluate the models' performance on unknown data by utilising either a hold-out set or cross-validation to validate the models and guarantee their resilience.

Performance Evaluation: In order to determine which model performs the best, we need to compare the models using the selected metrics. This entails a careful assessment of the models' predictive accuracy for hotel booking cancellations, taking into account precision, recall, or F1 score.

Model Refinement: we will spearhead the work to improve the model in light of the evaluation's findings. To improve the model's predictive power, this might entail changing the model's architecture or updating its hyperparameters.

6. Preliminary Results:

6.1. Initially loading the libraries:

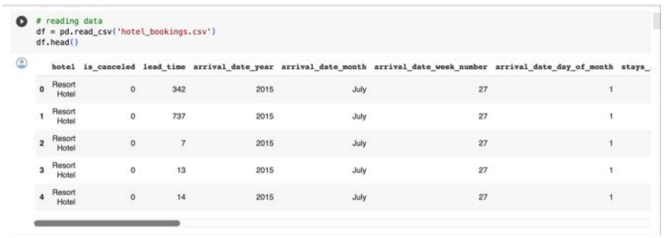


Hotel Bookings Cancellation Forecasting

```
# importing libraries

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

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier

from sklearn.ensemble import ExtraTreesClassifier
from lightgbm import LGBMClassifier
```

Figure 1. loading the libraries

6.2. Reading the dataset, hotel_bookings.csv file :

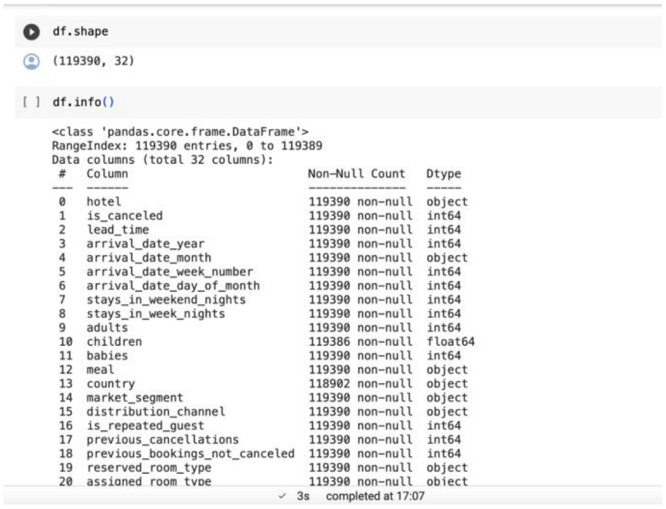


```
# reading data
df = pd.read_csv('hotel_bookings.csv')
df.head()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_
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	

Figure 2. by using the read_csv,loaded the dataset

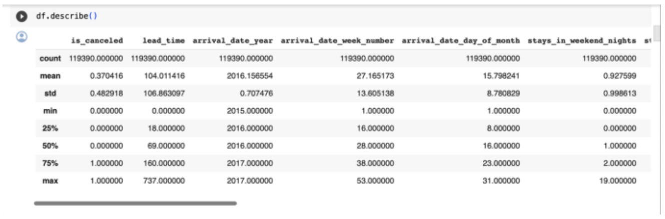
6.3. Understanding the Dataframe



```
df.shape
(119390, 32)

[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
```

Figure 3. Finding out the number of columns and rows: We have 119390 rows and 32 columns.We used shape and info attributes to get details about the dataset. Describe function is used to know the statistical summary of the dataset from numerical variables:

6.4. Statiscal understanding of numerical columns:

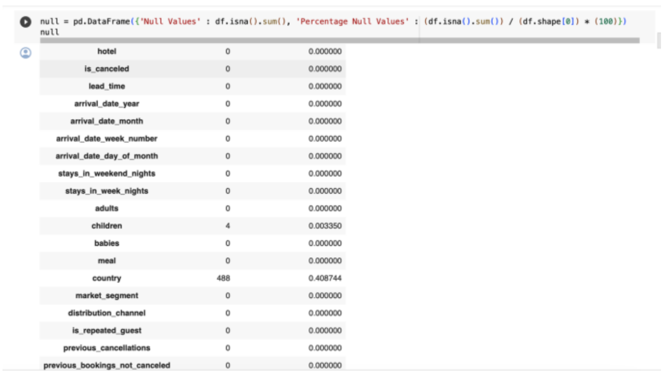


```
df.describe()
```

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	st
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.927599	
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613	
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000	
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000	
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000	
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000	
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000	

Figure 4. · lead_time: The duration, which varies from 0 to 737 days, between the booking and arrival dates is around 104 days on average. · date of arrival, week number: From 1 to 53, the average week number of arrivals each year is around 27.17. · arrival_date_day_of_month: From 1 to 31, the average day of the month of arrival is around 15.80. · stays_in_weekend_nights: Visitors spend an average of 0.93 weekend nights here, however some remain for up to 19 nights. · stays_in_week_nights: Visitors stay anywhere from 0 to 50 week nights, with an average of 2.5 week nights. · adults: The range of adult reservations is 0 to 55, with an average of 1.86 adults per booking. · children: There are typically 0 children per reservation, with a maximum of 10 children in some reservations. · babies: With a maximum of 10 babies in certain reservations, the average number of babies per booking is 0 babies. · previous_cancellations: Guests have cancelled 0.09 times on average in the past, with some having cancelled up to 26 times. · previous_bookings_not_canceled: Guests have, on average, not cancelled 0.14 reservations in the past; some guests have up to 72 reservations that they haven't cancelled. · booking_changes: Reservations are altered 0.22 times on average, and occasionally they are altered up to 21 times. · days_in_waiting_list: Reservations are typically kept on the waiting list for 2.32 days, but occasionally they are kept on for 391 days. · Adr: The average daily rate is around 101.83, with values ranging from 5400 to -6.38 (which may represent inaccuracies or exceptional situations). · required_car_parking_spaces: Guests typically need 0.06 parking spaces, while some reservations call for up to 8 spots.· total_of_special_requests: Guests typically make 0.57 special requests, while some have been known to make as many as five.

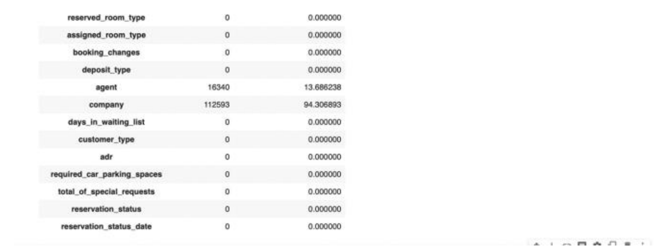
6.5. Finding out if there are any null values:



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

hotel	0	0.000000
is_canceled	0	0.000000
lead_time	0	0.000000
arrival_date_year	0	0.000000
arrival_date_month	0	0.000000
arrival_date_week_number	0	0.000000
arrival_date_day_of_month	0	0.000000
stays_in_weekend_nights	0	0.000000
stays_in_week_nights	0	0.000000
adults	0	0.000000
children	4	0.003350
babies	0	0.000000
meal	0	0.000000
country	488	0.408744
market_segment	0	0.000000
distribution_channel	0	0.000000
is_repeated_guest	0	0.000000
previous_cancellations	0	0.000000
previous_bookings_not_canceled	0	0.000000

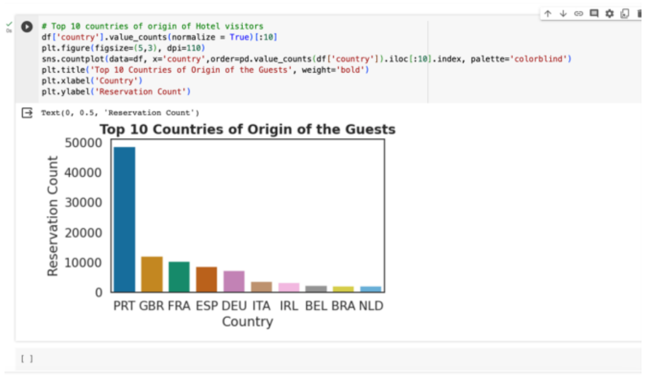
Figure 5. It is clear from above, we have null values in few columns, agent, company and country.Missing values: The column children has 4 missing values. The column country has 488 missing values



reserved_room_type	0	0.000000
assigned_room_type	0	0.000000
booking_changes	0	0.000000
deposit_type	0	0.000000
agent	16340	13.686238
company	112593	94.306893
days_in_waiting_list	0	0.000000
customer_type	0	0.000000
adr	0	0.000000
required_car_parking_spaces	0	0.000000
total_of_special_requests	0	0.000000
reservation_status	0	0.000000
reservation_status_date	0	0.000000

Figure 6. The column agent has 16,340 missing values. The column company has a significant number of missing values, totaling 112,593

6.6. Determining the most common countries of origin for visitors:



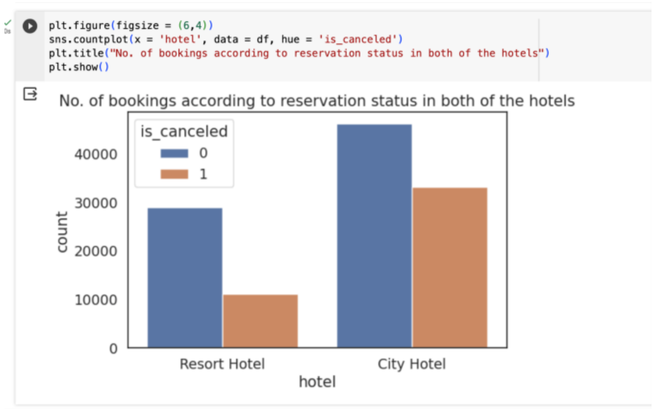
```
# Top 10 countries of origin of Hotel visitors
df['country'].value_counts(normalize = True)[:10]
plt.figure(figsize=(5,3), dpi=110)
sns.countplot(data=df, x='country',order=pd.value_counts(df['country']).iloc[:10].index, palette='colorblind')
plt.title('Top 10 Countries of Origin of the Guests', weight='bold')
plt.xlabel('Country')
plt.ylabel('Reservation Count')
```

```
Text(0, 0.5, 'Reservation Count')
```

Top 10 Countries of Origin of the Guests
Reservation Count
50000
40000
30000
20000
10000
0
PRT GBR FRA ESP DEU ITA IRL BEL BRA NLD
Country

Figure 7. Out of the visitors for the hotels in 2015,2016,2017, most visitors are from PRT followed by GBR.

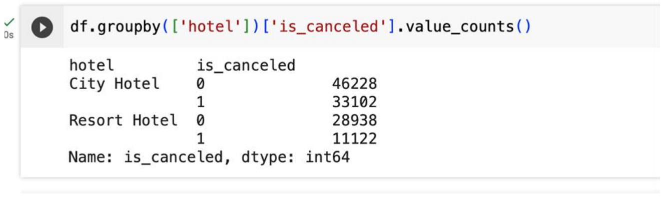
6.7. Analysing cancellation trends unique to various hotel categories:



```
plt.figure(figsize = (6,4))
sns.countplot(x = 'hotel', data = df, hue = 'is_canceled')
plt.title("No. of bookings according to reservation status in both of the hotels")
plt.show()
```

No. of bookings according to reservation status in both of the hotels
is_canceled
0
1
count
40000
30000
20000
10000
0
Resort Hotel
City Hotel
hotel

Figure 8. We have most bookings for city hotel compared to resort type hotel. The number of bookings that are cancelled for city hotel are 33102, and not cancelled are 46228, where as for the Resort hotel are 28938, and cancelled are 11122.

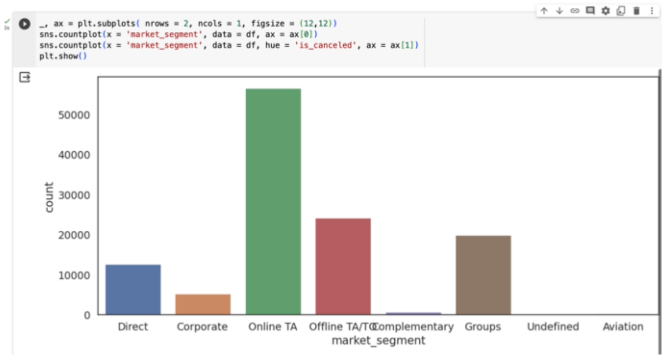


```
df.groupby(['hotel'])['is_canceled'].value_counts()
```

```
hotel         is_canceled
City Hotel    0              46228
              1              33102
Resort Hotel  0              28938
              1              11122
Name: is_canceled, dtype: int64
```

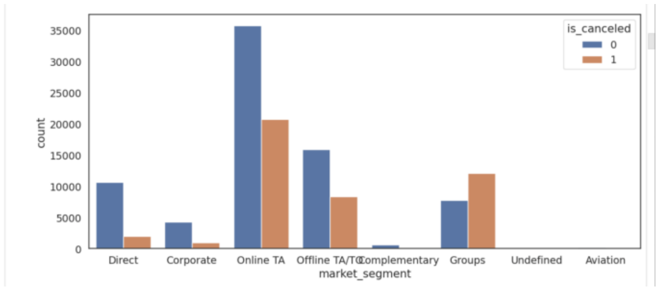
Figure 9. Count of bookings done with and without cancellations hotel wise

6.8. Determining how visits are distributed among various market sectors:

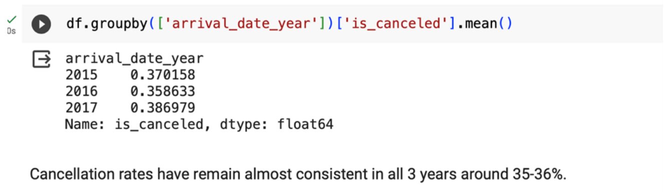


```
_, ax = plt.subplots( nrows = 2, ncols = 1, figsize = (12,12))
sns.countplot(x = 'market_segment', data = df, ax = ax[0])
sns.countplot(x = 'market_segment', data = df, hue = 'is_canceled', ax = ax[1])
plt.show()
```

Figure 10. Online TA occupies the top market sector for hotel booking followed by offline TA and least being complementary.



6.9. Identifying trends and patterns in the cancellation rates across several years:

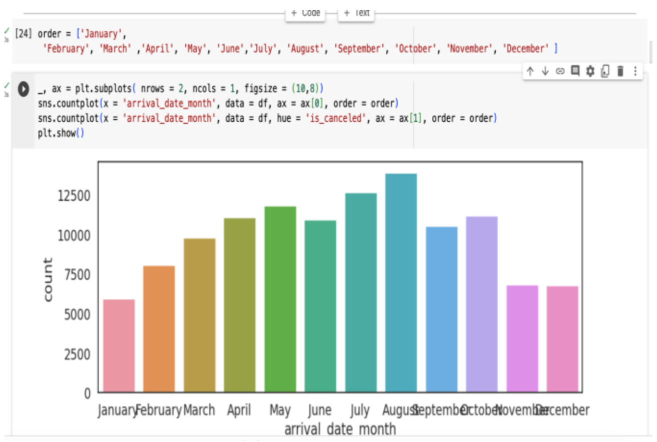


```
df.groupby(['arrival_date_year'])['is_canceled'].mean()
```

```
arrival_date_year
2015    0.370158
2016    0.358633
2017    0.386979
Name: is_canceled, dtype: float64
```

Cancellation rates have remain almost consistent in all 3 years around 35-36%.

6.10. Finding Peak Visitor Periods to pinpoint and measure the month that sees the largest influx of tourists:

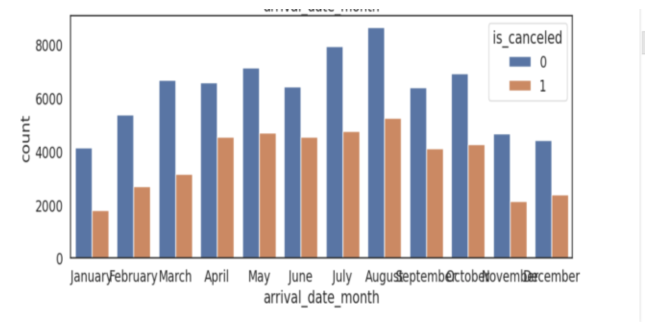


```
[24] order = ['January',
      'February', 'March' ,'April', 'May', 'June','July', 'August', 'September', 'October', 'November', 'December' ]
```

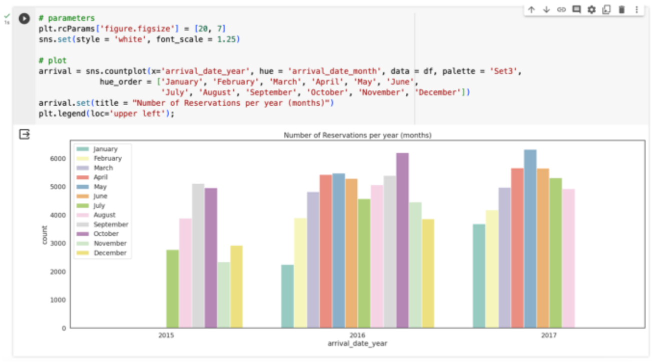
```
_, ax = plt.subplots( nrows = 2, ncols = 1, figsize = (10,8))
sns.countplot(x = 'arrival_date_month', data = df, ax = ax[0], order = order)
sns.countplot(x = 'arrival_date_month', data = df, hue = 'is_canceled', ax = ax[1], order = order)
plt.show()
```

Figure 11. From graph, it is clear we have most visitors In August followed by July, May, and April.

6.11. Count of bookings divided by cancelled and not cancelled:

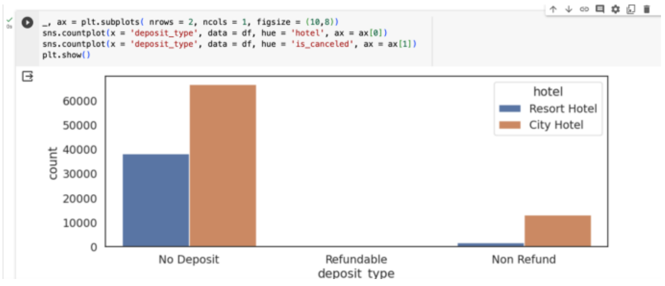


6.12. Bookings based on month and year Wise:

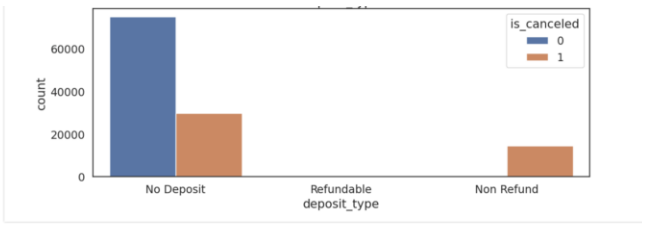


```
# parameters
plt.rcParams['figure.figsize'] = [20, 7]
sns.set(style = 'white', font_scale = 1.25)

# plot
arrival = sns.countplot(x='arrival_date_year', hue = 'arrival_date_month', data = df, palette = 'Set3',
              hue_order = ['January', 'February', 'March', 'April', 'May', 'June',
                          'July', 'August', 'September', 'October', 'November', 'December'])
arrival.set(title = "Number of Reservations per year (months)")
plt.legend(loc='upper left');
```

6.13. Examining the relationship that exists between the deposit types and the probability of cancellations:

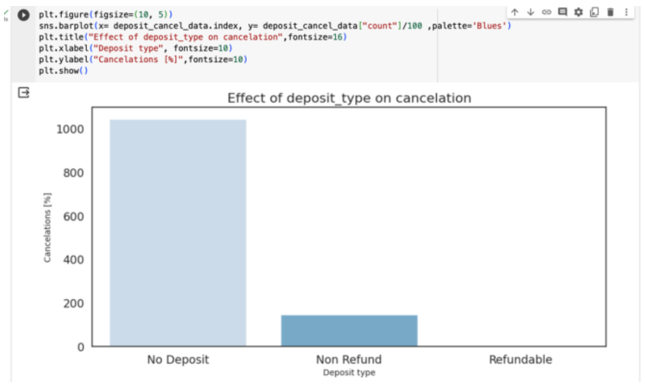


```
_, ax = plt.subplots( nrows = 2, ncols = 1, figsize = (10,8))
sns.countplot(x = 'deposit_type', data = df, hue = 'hotel', ax = ax[0])
sns.countplot(x = 'deposit_type', data = df, hue = 'is_canceled', ax = ax[1])
plt.show()
```



Deposit type has 3 categories - No Deposit, refundable, Non Refund Either customers have opted for no deposit or non refundable deposits. Maybe refundable deposit type is not offered by the hotels. All of the non refund bookings have been cancelled in our dataset. That might prove important feature based on how many such bookings are part of cancelled bookings.

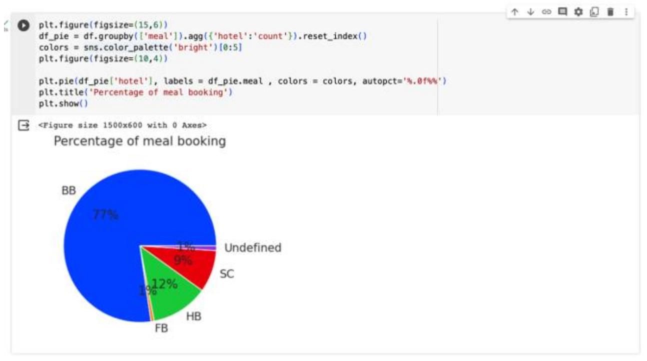
No hotel has refundable deposit type



```
plt.figure(figsize=(10, 5))
sns.barplot(x= deposit_cancel_data.index, y= deposit_cancel_data["count"]/100 ,palette='Blues')
plt.title("Effect of deposit_type on cancelation",fontsize=16)
plt.xlabel("Deposit type", fontsize=10)
plt.ylabel("Cancelations [%]",fontsize=10)
plt.show()
```

Figure 12. NO Deposit is more cancellation , let’s show it in visualization 99% of people how No deposit have canceled their hotel bookings

6.14. Assessing the percentage distribution of meal reservations to determine the frequency of various meal options among visitors:



```
plt.figure(figsize=(15,6))
df_pie = df.groupby(['meal']).agg({'hotel':'count'}).reset_index()
colors = sns.color_palette('bright')[0:5]
plt.figure(figsize=(10,4))

plt.pie(df_pie['hotel'], labels = df_pie.meal , colors = colors, autopct='%.0f%%')
plt.title('Percentage of meal booking')
plt.show()
```

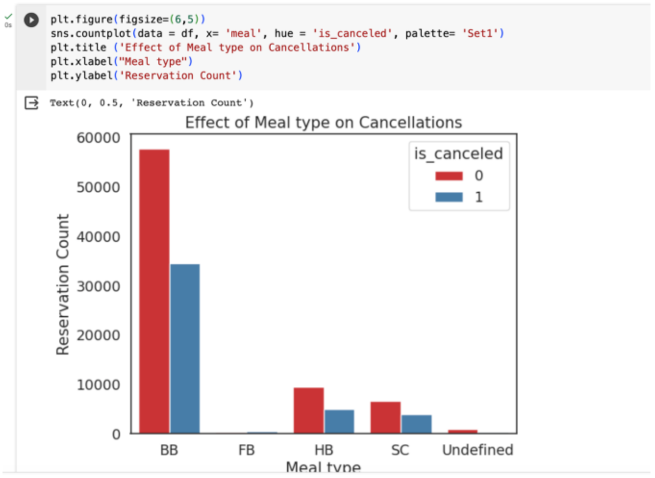
```
<Figure size 1500x600 with 0 Axes>
```

Figure 13. most of the visitors prefer bed & breakfast (BB) followed by full board (FB), half board (HB) and self cater (SC)

6.15. Meal type effect on Cancellations:

```
df.groupby('meal')['is_canceled'].value_counts()
```

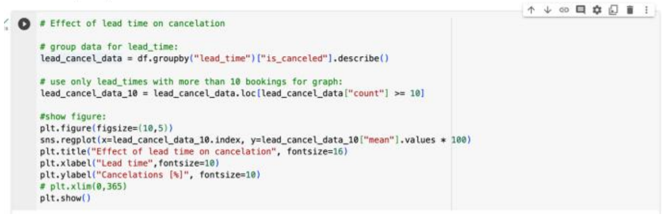
```
meal       is_canceled
BB         0              57800
           1              34510
FB         1                478
           0                320
HB         0               9479
           1               4984
SC         0               6684
           1               3966
Undefined  0                883
           1                286
Name: is_canceled, dtype: int64
```



```
plt.figure(figsize=(6,5))
sns.countplot(data = df, x= 'meal', hue = 'is_canceled', palette= 'Set1')
plt.title ('Effect of Meal type on Cancellations')
plt.xlabel("Meal type")
plt.ylabel('Reservation Count')
```

```
Text(0, 0.5, 'Reservation Count')
```

6.16. Calculating and examining how lead time variations affect the probability of cancellations:



```
# Effect of lead time on cancelation

# group data for lead_time:
lead_cancel_data = df.groupby("lead_time")["is_canceled"].describe()

# use only lead_times with more than 10 bookings for graph:
lead_cancel_data_10 = lead_cancel_data.loc[lead_cancel_data["count"] >= 10]

#show figure:
plt.figure(figsize=(10,5))
sns.regplot(x=lead_cancel_data_10.index, y=lead_cancel_data_10["mean"].values * 100)
plt.title("Effect of lead time on cancelation", fontsize=16)
plt.xlabel("Lead time",fontsize=10)
plt.ylabel("Cancelations [%]", fontsize=10)
# plt.xlim(0,365)
plt.show()
```

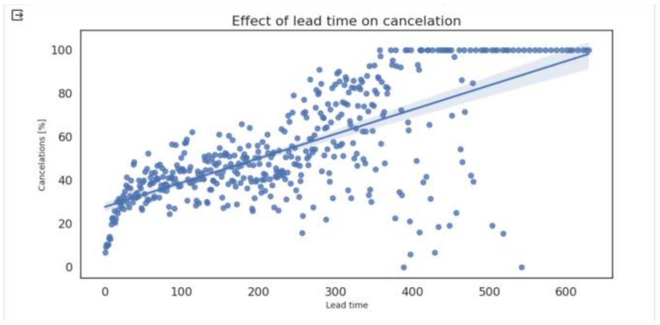
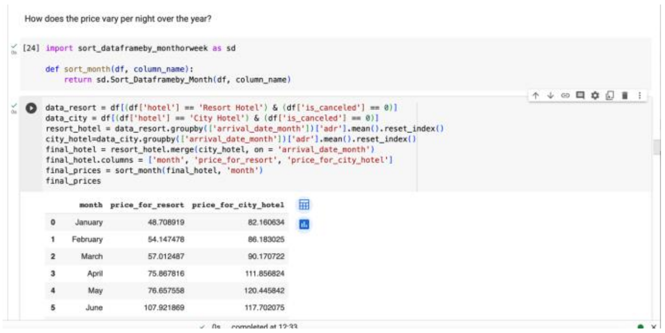


Figure 14. Bookings made a few days before the arrival date are rarely canceled, whereas bookings made over one year in advance are canceled very often.

6.17. Examining the variations in nightly prices across the year's several seasons:



How does the price vary per night over the year?

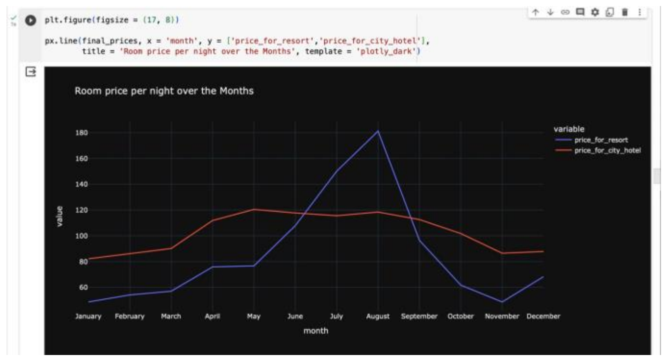
```
[24] import sort_dataframeby_monthorweek as sd

def sort_month(df, column_name):
    return sd.Sort_Dataframeby_Month(df, column_name)
```

```
data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] == 0)]
data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] == 0)]
resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'].mean().reset_index()
city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index()
final_hotel = resort_hotel.merge(city_hotel, on = 'arrival_date_month')
final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']
final_prices = sort_month(final_hotel, 'month')
final_prices
```

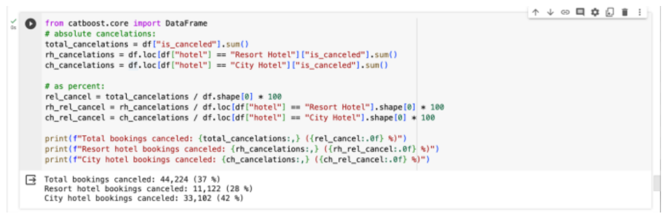
	month	price_for_resort	price_for_city_hotel
0	January	48.708919	82.160634
1	February	54.147478	86.183025
2	March	57.012487	90.170722
3	April	75.867816	111.856824
4	May	76.657558	120.445842
5	June	107.921869	117.702075

Figure 15. Seasonal Variation: Rates for city and resort hotels typically fluctuate throughout the year, with noticeable hikes in the months that see the highest volume of tourism.August is the month with the highest costs, showing that this is the busiest time of year for resort hotels.November had the lowest prices, indicating a period of less demand. City Hotel Rates: City hotels have yearly rates that are largely consistent.There is a minor price rise in August, maybe as a result of more demand.

6.18. Month wise room price in visualization



```
plt.figure(figsize = (17, 8))

px.line(final_prices, x = 'month', y = ['price_for_resort','price_for_city_hotel'],
        title = 'Room price per night over the Months', template = 'plotly_dark')
```



```
from catboost.core import DataFrame
# absolute cancelations:
total_cancelations = df["is_canceled"].sum()
rh_cancelations = df.loc[df["hotel"] == "Resort Hotel"]["is_canceled"].sum()
ch_cancelations = df.loc[df["hotel"] == "City Hotel"]["is_canceled"].sum()

# as percent:
rel_cancel = total_cancelations / df.shape[0] * 100
rh_rel_cancel = rh_cancelations / df.loc[df["hotel"] == "Resort Hotel"].shape[0] * 100
ch_rel_cancel = ch_cancelations / df.loc[df["hotel"] == "City Hotel"].shape[0] * 100

print(f"Total bookings canceled: {total_cancelations:,} ({rel_cancel:.0f} %)")
print(f"Resort hotel bookings canceled: {rh_cancelations:,} ({rh_rel_cancel:.0f} %)")
print(f"City hotel bookings canceled: {ch_cancelations:,} ({ch_rel_cancel:.0f} %)")
```

```
Total bookings canceled: 44,224 (37 %)
Resort hotel bookings canceled: 11,122 (28 %)
City hotel bookings canceled: 33,102 (42 %)
```

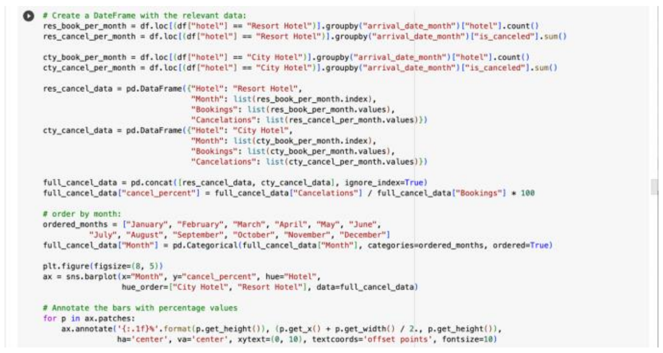
Total Cancellations Of the bookings in the dataset, 44,224 have been cancelled in total, or 37% of all bookings. This shows a sizable percentage of cancellations, highlighting how crucial it is for the hotel sector to comprehend and handle cancellations.

Refunds according to Hotel Type:

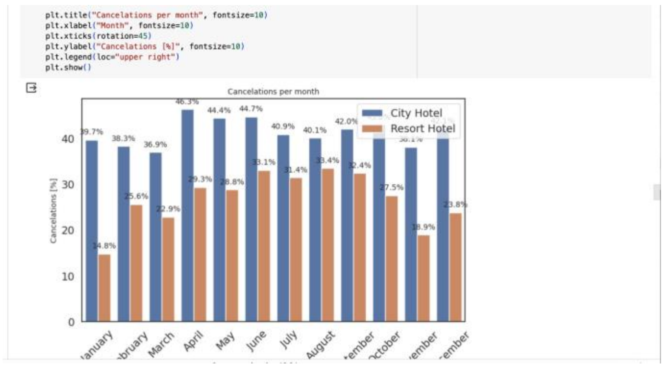
Resort Hotel: Of all the reservations made, the Resort Hotel had 11,122 cancellations, or 28% of the total. This implies that, in comparison to the entire dataset, Resort Hotels have a comparatively lower cancellation rate.

City Hotel: In contrast, the City Hotel saw a larger percentage of cancellations—33,102, or 42% of all reservations—than any other hotel. This suggests that reservations for City Hotels are more erratic, which might be impacted by things like last-minute travel or business travel. The City Hotel has a greater cancellation rate than the Resort Hotel. This might be because reservations for city destinations—such as business trips—are different from those for resort destinations—such as leisure holidays. Given that resort hotels have comparatively lower cancellation rates, travellers who book these kinds of lodgings may have more flexible schedules or longer booking windows.

6.19. Determine the trends and patterns in the percentage of cancellations for each kind of hotel during the various seasons



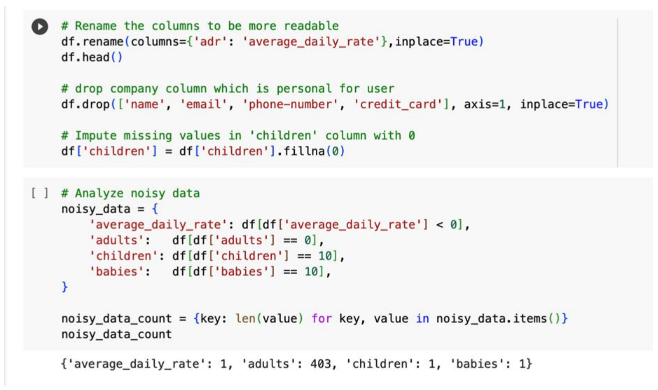
```
# Create a DateFrame with the relevant data:
res_book_per_month = df.loc[(df["hotel"] == "Resort Hotel")].groupby("arrival_date_month")["hotel"].count()
res_cancel_per_month = df.loc[(df["hotel"] == "Resort Hotel")].groupby("arrival_date_month")["is_canceled"].sum()

cty_book_per_month = df.loc[(df["hotel"] == "City Hotel")].groupby("arrival_date_month")["hotel"].count()
cty_cancel_per_month = df.loc[(df["hotel"] == "City Hotel")].groupby("arrival_date_month")["is_canceled"].sum()

res_cancel_data = pd.DataFrame({"Hotel": "Resort Hotel",
                                "Month": list(res_book_per_month.index),
                                "Bookings": list(res_book_per_month.values),
                                "Cancelations": list(res_cancel_per_month.values)})
cty_cancel_data = pd.DataFrame({"Hotel": "City Hotel",
                                "Month": list(cty_book_per_month.index),
                                "Bookings": list(cty_book_per_month.values),
                                "Cancelations": list(cty_cancel_per_month.values)})

full_cancel_data = pd.concat([res_cancel_data, cty_cancel_data], ignore_index=True)
full_cancel_data["cancel_percent"] = full_cancel_data["Cancelations"] / full_cancel_data["Bookings"] * 100

# order by month:
ordered_months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]
full_cancel_data["Month"] = pd.Categorical(full_cancel_data["Month"], categories=ordered_months, ordered=True)

plt.figure(figsize=(8, 5))
ax = sns.barplot(x="Month", y="cancel_percent", hue="Hotel",
                 hue_order=["City Hotel", "Resort Hotel"], data=full_cancel_data)

# Annotate the bars with percentage values
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points', fontsize=10)
```



```
plt.title("Cancelations per month", fontsize=10)
plt.xlabel("Month", fontsize=10)
plt.xticks(rotation=45)
plt.ylabel("Cancelations [%]", fontsize=10)
plt.legend(loc="upper right")
plt.show()
```

Figure 16. From graph, it is clear that highest cancellations are in April followed by June and may.

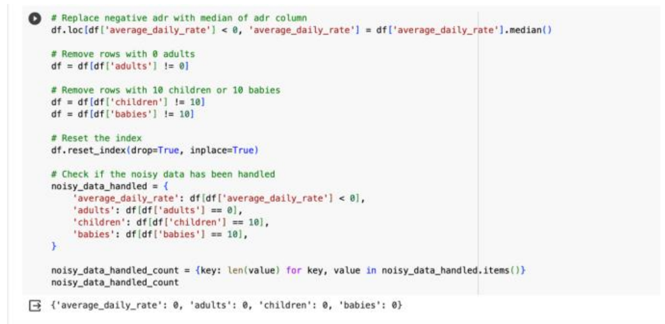
6.20. Cleaning the data from above understandings:



```
# Rename the columns to be more readable
df.rename(columns={'adr': 'average_daily_rate'},inplace=True)
df.head()

# drop company column which is personal for user
df.drop(['name', 'email', 'phone-number', 'credit_card'], axis=1, inplace=True)

# Impute missing values in 'children' column with 0
df['children'] = df['children'].fillna(0)
```

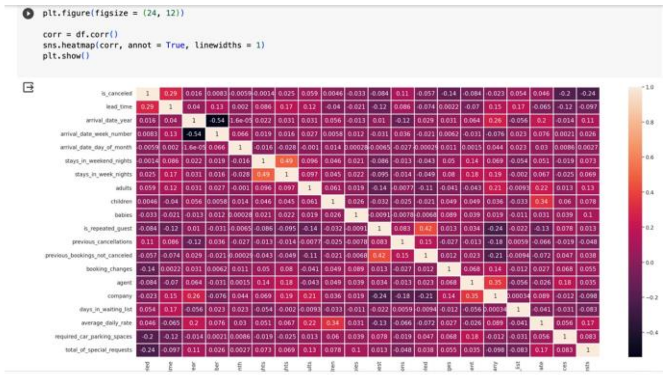
```
[ ] # Analyze noisy data
noisy_data = {
    'average_daily_rate': df[df['average_daily_rate'] < 0],
    'adults':   df[df['adults'] == 0],
    'children': df[df['children'] == 10],
    'babies':   df[df['babies'] == 10],
}

noisy_data_count = {key: len(value) for key, value in noisy_data.items()}
noisy_data_count
```

```
{'average_daily_rate': 1, 'adults': 403, 'children': 1, 'babies': 1}
```



```
# Replace negative adr with median of adr column
df.loc[df['average_daily_rate'] < 0, 'average_daily_rate'] = df['average_daily_rate'].median()

# Remove rows with 0 adults
df = df[df['adults'] != 0]

# Remove rows with 10 children or 10 babies
df = df[df['children'] != 10]
df = df[df['babies'] != 10]

# Reset the index
df.reset_index(drop=True, inplace=True)

# Check if the noisy data has been handled
noisy_data_handled = {
    'average_daily_rate': df[df['average_daily_rate'] < 0],
    'adults': df[df['adults'] == 0],
    'children': df[df['children'] == 10],
    'babies': df[df['babies'] == 10],
}

noisy_data_handled_count = {key: len(value) for key, value in noisy_data_handled.items()}
noisy_data_handled_count
```

```
{'average_daily_rate': 0, 'adults': 0, 'children': 0, 'babies': 0}
```



```
plt.figure(figsize = (24, 12))

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

Findings from the Target Variable (is_canceled) Feature Correlation: Strong Positive Correlations:

· **lead_time (0.293):** The chance of cancellation and lead time have a moderately positive association. This implies that the likelihood of a cancellation tends to grow along with the lead time, or the number of days from the date of booking to the arrival.

· **total_of_special_requests (0.235):** A moderately positive correlation suggests that there is a positive link between the number of special requests and the chance of not cancelling.

· **required_car_parking_spaces (0.196):**Bookings that include a necessity for car parking spaces are a little more likely to be cancelled, according to the moderately positive correlation.

Weak Positive Correlations:

· **booking_changes (0.145):** According to this weak positive correlation, there is a marginally greater chance of cancellation for every rise in booking changes.

· **previous_cancellations (0.110):** A weakly positive correlation suggests that there is a somewhat higher chance of cancellations in the future if there has been a history of cancellations.

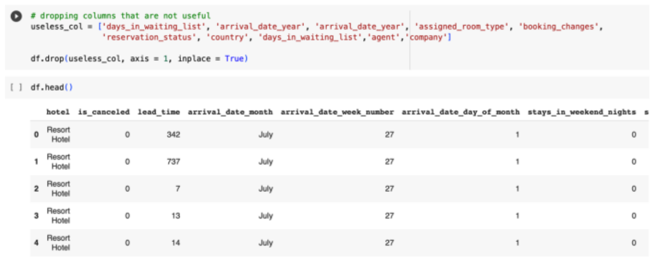
Weak Negative Correlations:

· **is_repeated_guest (0.084):** This correlation indicates that there is a minor reduction in the likelihood of cancellation for repeated visitors.

· **agent (0.084):** A somewhat negative correlation suggests that reservations booked via an agent have a marginally lower chance of being cancelled.

· **adults (0.059):** Bookings with more adults appear to have a somewhat lower chance of being cancelled, according to a weak negative association. Additional factors like newborns, stays in weekend nights, company, arrival date year, arrival date week number, arrival date day of month, prior reservations not cancelled, days in waiting list, and average daily rate have weak correlations. Now lets drop few columns which we don't have much correlation with the is_cancelled.

6.21. Defining a dataframe which will have categorical columns:

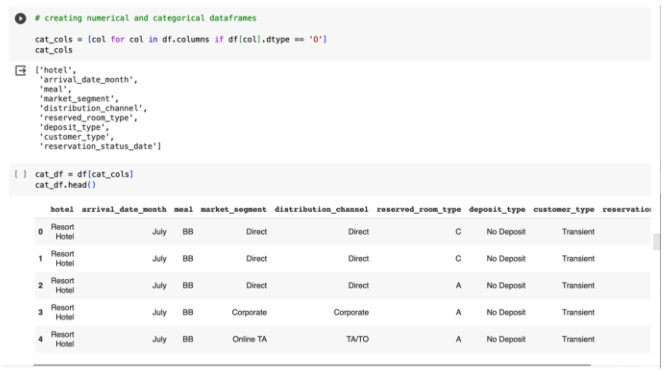


```
# dropping columns that are not useful
useless_col = ['days_in_waiting_list', 'arrival_date_year', 'arrival_date_year', 'assigned_room_type', 'booking_changes',
               'reservation_status', 'country', 'days_in_waiting_list','agent','company']

df.drop(useless_col, axis = 1, inplace = True)
```

```
[ ] df.head()
```

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

Figure 17. Defining a dataframe which will have categorical columns



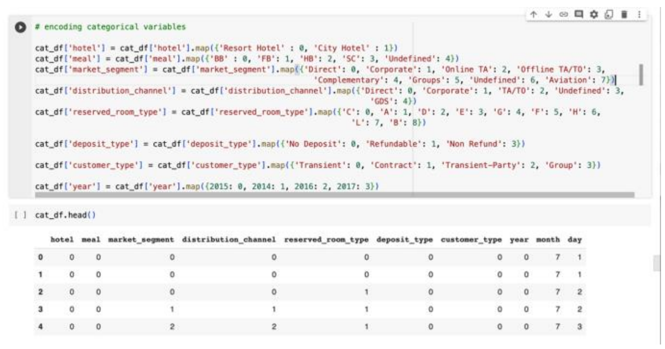
```
# creating numerical and categorical dataframes

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

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

```
[ ] cat_df = df[cat_cols]
    cat_df.head()
```

	hotel	arrival_date_month	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	reservatio
0	Resort Hotel	July	BB	Direct	Direct	C	No Deposit	Transient	
1	Resort Hotel	July	BB	Direct	Direct	C	No Deposit	Transient	
2	Resort Hotel	July	BB	Direct	Direct	A	No Deposit	Transient	
3	Resort Hotel	July	BB	Corporate	Corporate	A	No Deposit	Transient	
4	Resort Hotel	July	BB	Online TA	TA/TO	A	No Deposit	Transient	

Figure 18. We need encode the categorical variables, which is useful for the model creation and analysis:



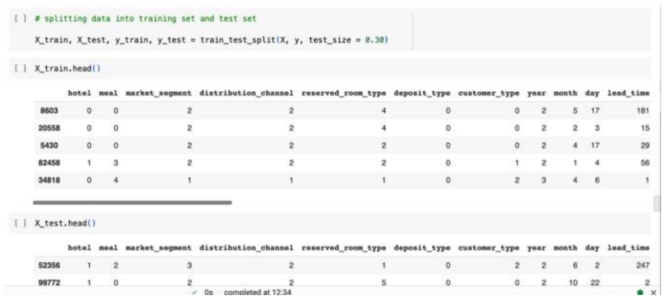
```
# encoding categorical variables

cat_df['hotel'] = cat_df['hotel'].map({'Resort Hotel' : 0, 'City Hotel' : 1})
cat_df['meal'] = cat_df['meal'].map({'BB' : 0, 'FB': 1, 'HB': 2, 'SC': 3, 'Undefined': 4})
cat_df['market_segment'] = cat_df['market_segment'].map({'Direct': 0, 'Corporate': 1, 'Online TA': 2, 'Offline TA/TO': 3,
                                                          'Complementary': 4, 'Groups': 5, 'Undefined': 6, 'Aviation': 7})
cat_df['distribution_channel'] = cat_df['distribution_channel'].map({'Direct': 0, 'Corporate': 1, 'TA/TO': 2, 'Undefined': 3,
                                                                      'GDS': 4})
cat_df['reserved_room_type'] = cat_df['reserved_room_type'].map({'C': 0, 'A': 1, 'D': 2, 'E': 3, 'G': 4, 'F': 5, 'H': 6,
                                                                  'L': 7, 'B': 8})

cat_df['deposit_type'] = cat_df['deposit_type'].map({'No Deposit': 0, 'Refundable': 1, 'Non Refund': 3})

cat_df['customer_type'] = cat_df['customer_type'].map({'Transient': 0, 'Contract': 1, 'Transient-Party': 2, 'Group': 3})

cat_df['year'] = cat_df['year'].map({2015: 0, 2014: 1, 2016: 2, 2017: 3})
```

```
[ ] cat_df.head()
```

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

Figure 19. categorical values

6.22. Splitting the data for Training and Testing



```
[ ] # splitting data into training set and test set

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

```
[ ] X_train.head()
```

	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day	lead_time
8603	0	0	2	2	4	0	0	2	5	17	181
20558	0	0	2	2	4	0	0	2	2	3	15
5430	0	0	2	2	2	0	0	2	4	17	29
82458	1	3	2	2	2	0	1	2	1	4	56
34818	0	4	1	1	1	0	2	3	4	6	1

```
[ ] X_test.head()
```

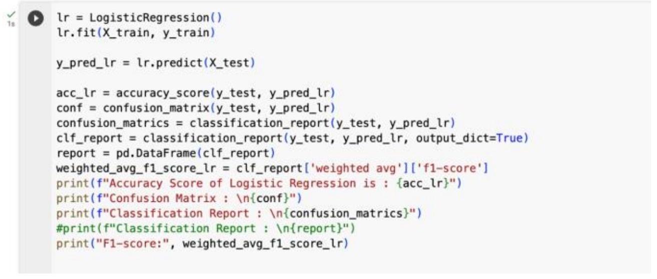
	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day	lead_time
52356	1	2	3	2	1	0	2	2	6	2	247
99772	1	0	2	2	5	0	0	2	10	22	2

Figure 20. Splitting the data for Training and Testing

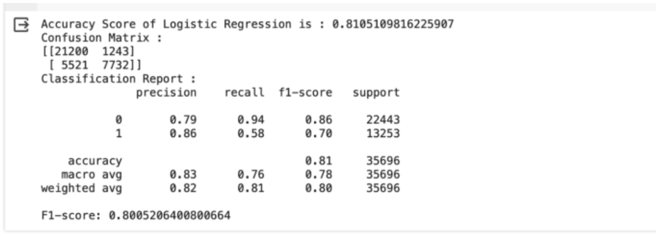
6.23. Model Creation

6.23.1. Logistic Regression.

Logistic Regression



```
lr = LogisticRegression()
lr.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)

acc_lr = accuracy_score(y_test, y_pred_lr)
conf = confusion_matrix(y_test, y_pred_lr)
confusion_matrics = classification_report(y_test, y_pred_lr)
clf_report = classification_report(y_test, y_pred_lr, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_lr = clf_report['weighted avg']['f1-score']
print(f"Accuracy Score of Logistic Regression is : {acc_lr}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
#print(f"Classification Report : \n{report}")
print("F1-score:", weighted_avg_f1_score_lr)
```

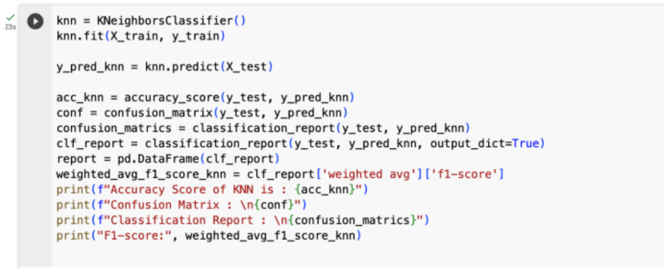


```
Accuracy Score of Logistic Regression is : 0.8105109816225907
Confusion Matrix :
[[21200  1243]
 [ 5521  7732]]
Classification Report :
              precision    recall  f1-score   support

           0       0.79      0.94      0.86     22443
           1       0.86      0.58      0.70     13253

    accuracy                           0.81     35696
   macro avg       0.83      0.76      0.78     35696
weighted avg       0.82      0.81      0.80     35696

F1-score: 0.8005206400800664
```

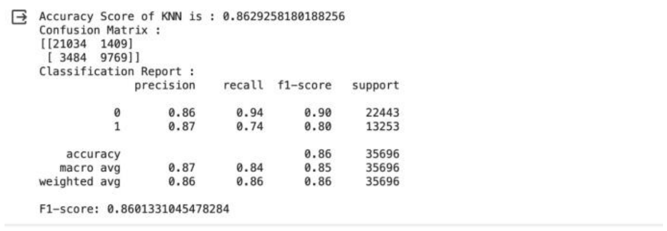
Figure 21. The Logistic Regression model’s accuracy score is around 81.1%, meaning that for roughly 81.1% of the test set’s instances, the model accurately predicted the target variable.True Positives (TP): 7,732 cases were accurately classified as class 1 in the confusion matrix.True Negatives (TN): 21,200 cases were classified as class 0 with accuracy.False Positives (FP): 1,243 cases were misclassified as class 1 in error.False Negatives (FN): 5,521 cases were misclassified as class 0 in error.F1-score, recall, and precision:Precision: Class 1 has a precision of 0.86, meaning that 86% of the time the model predicts class 1 correctly.Recall (Sensitivity): Class 1 has a recall of 0.58, meaning that 58% of the real instances of class 1 were caught by the model. f1-score: A weighted average F1-score of 0.80 strikes a balance between recall and accuracy. It offers a general indicator of the model’s effectiveness across all classes.

6.23.2. KNN.

KNN



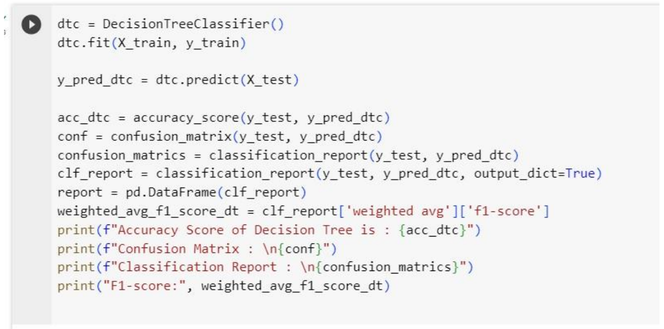
```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

acc_knn = accuracy_score(y_test, y_pred_knn)
conf = confusion_matrix(y_test, y_pred_knn)
confusion_matrics = classification_report(y_test, y_pred_knn)
clf_report = classification_report(y_test, y_pred_knn, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_knn = clf_report['weighted avg']['f1-score']
print(f"Accuracy Score of KNN is : {acc_knn}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
print("F1-score:", weighted_avg_f1_score_knn)
```



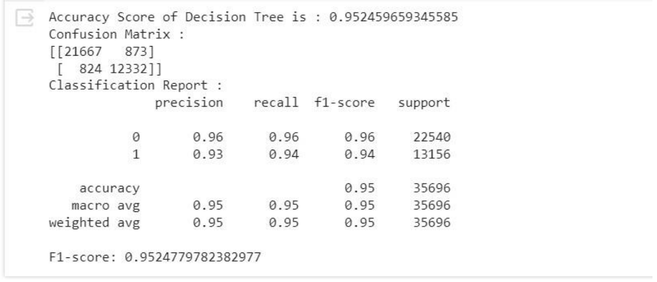
```
Accuracy Score of KNN is : 0.8629258180188256
Confusion Matrix :
[[21034  1409]
 [ 3484  9769]]
Classification Report :
              precision    recall  f1-score   support

           0       0.86      0.94      0.90     22443
           1       0.87      0.74      0.80     13253

    accuracy                           0.86     35696
   macro avg       0.87      0.84      0.85     35696
weighted avg       0.86      0.86      0.86     35696

F1-score: 0.8601331045478284
```

Figure 22. The KNN model’s accuracy score is around 86.3%, meaning that for roughly 86.3% of the test set’s instances, the model successfully predicted the target variable. This is a useful place to start when assessing the model’s overall effectiveness. Confusion Matrix: The confusion matrix gives a thorough explanation of what forecasts were right and wrong.True Positives (TP): 9,769 cases had class 1 predictions made accurately.True Negatives (TN): 21,034 cases were classified as class 0 with accuracy. False Positives (FP): 1,409 cases were misclassified as class 1 in error.False Negatives (FN): 3,484 cases were misclassified as class 0 in error. F1-score, recall, and precision: Precision: Class 1 accuracy is 0.87, meaning that 87% of the time the model is right when it predicts class 1. Recall (Sensitivity): Class 1 has a recall of 0.74, meaning that 74% of real class 1 occurrences were caught by the model. F1-score: A weighted average F1-score of 0.86 strikes a balance between recall and accuracy

6.23.3. Decision Tree Classifier:.



```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

y_pred_dtc = dtc.predict(X_test)

acc_dtc = accuracy_score(y_test, y_pred_dtc)
conf = confusion_matrix(y_test, y_pred_dtc)
confusion_matrics = classification_report(y_test, y_pred_dtc)
clf_report = classification_report(y_test, y_pred_dtc, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_dt = clf_report['weighted avg']['f1-score']
print(f"Accuracy Score of Decision Tree is : {acc_dtc}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
print("F1-score:", weighted_avg_f1_score_dt)
```

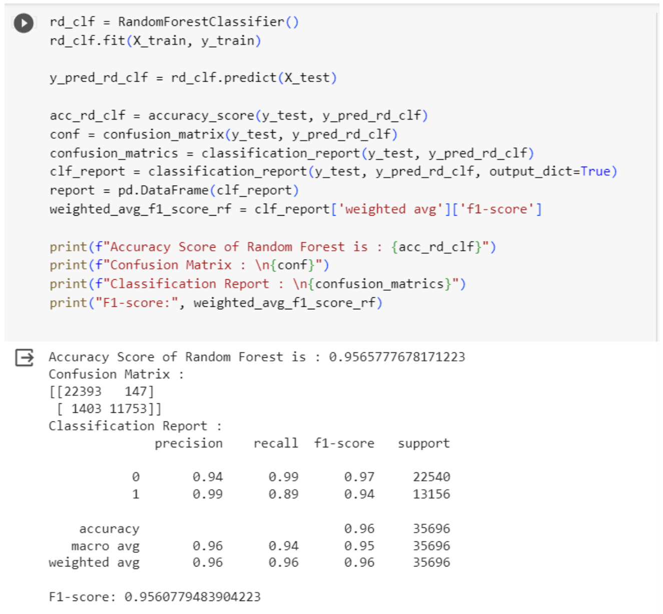


```
Accuracy Score of Decision Tree is : 0.952459659345585
Confusion Matrix :
[[21667   873]
 [  824 12332]]
Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     22540
           1       0.93      0.94      0.94     13156

    accuracy                           0.95     35696
   macro avg       0.95      0.95      0.95     35696
weighted avg       0.95      0.95      0.95     35696

F1-score: 0.9524779782382977
```

Figure 23. Here, Precision is 0.96 for class 0 and 0.93 for class 1.(Sensitivity) Recall is 0.96 for class 0 and 0.94 for class 1. For both courses, the weighted average F1-score is almost 0.9525. In the given dataset, support is the total number of real instances of the class. It is 22540 for class 0 and 13156 for class 1.F1-Score Weighted Average: 0.9524779782382977 This represents the weighted average of the F1-scores for the two classes, accounting for the unequal distribution of class sizes. It is roughly 0.9525, suggesting that recall and precision are generally well-balanced. Overall, it appears that your Decision Tree model is working well, exhibiting high accuracy and a balance between precision and recall.

6.23.4. Random Forest Classifier

The accuracy score of the model is around 95.66%. precision is 0.94 for class 0 and 0.99 for class 1. sensi-

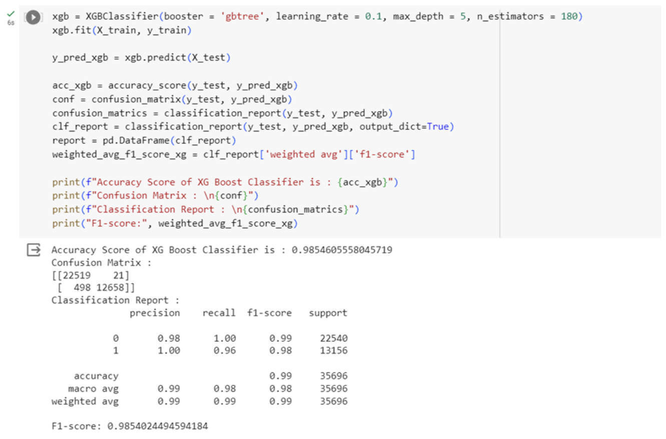


```
rd_clf = RandomForestClassifier()
rd_clf.fit(X_train, y_train)

y_pred_rd_clf = rd_clf.predict(X_test)

acc_rd_clf = accuracy_score(y_test, y_pred_rd_clf)
conf = confusion_matrix(y_test, y_pred_rd_clf)
confusion_matrics = classification_report(y_test, y_pred_rd_clf)
clf_report = classification_report(y_test, y_pred_rd_clf, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_rf = clf_report['weighted avg']['f1-score']

print(f"Accuracy Score of Random Forest is : {acc_rd_clf}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
print("F1-score:", weighted_avg_f1_score_rf)
```

```
Accuracy Score of Random Forest is : 0.9565777678171223
Confusion Matrix :
[[22393   147]
 [ 1403 11753]]
Classification Report :
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     22540
           1       0.99      0.89      0.94     13156

    accuracy                           0.96     35696
   macro avg       0.96      0.94      0.95     35696
weighted avg       0.96      0.96      0.96     35696

F1-score: 0.9560779483904223
```

tivity 0.89 for class 1 and 0.99 for class 0. The weighted average of accuracy and recall is known as the F1-score. When there is an imbalance in the courses, this statistic is helpful. For both groups, the weighted average F1_score is almost 0.9561. Support: In the given dataset, support is the total number of real instances of the class. It is 22540 for class 0 and 13156 for class 1. F1_Score Weighted Average: 0.9560779483904223

6.23.5. XgBoost Classifier



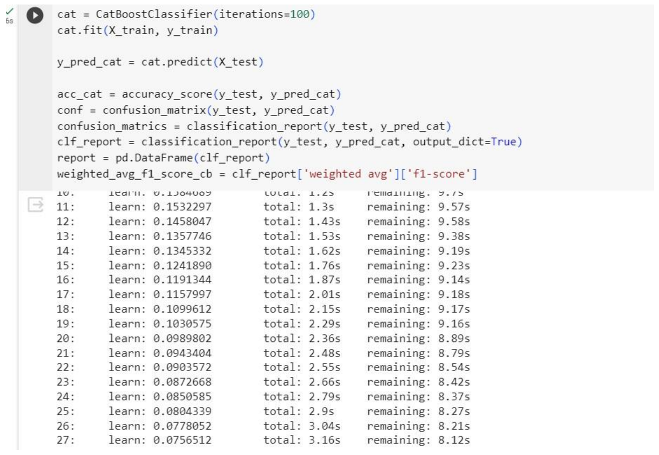
```
xgb = XGBClassifier(booster = 'gbtree', learning_rate = 0.1, max_depth = 5, n_estimators = 180)
xgb.fit(X_train, y_train)

y_pred_xgb = xgb.predict(X_test)

acc_xgb = accuracy_score(y_test, y_pred_xgb)
conf = confusion_matrix(y_test, y_pred_xgb)
confusion_matrics = classification_report(y_test, y_pred_xgb)
clf_report = classification_report(y_test, y_pred_xgb, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_xg = clf_report['weighted avg']['f1-score']

print(f"Accuracy Score of XG Boost Classifier is : {acc_xgb}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
print("F1-score:", weighted_avg_f1_score_xg)
```

```
Accuracy Score of XG Boost Classifier is : 0.9854605558045719
Confusion Matrix :
[[22519    21]
 [  498 12658]]
Classification Report :
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     22540
           1       1.00      0.96      0.98     13156

    accuracy                           0.99     35696
   macro avg       0.99      0.98      0.98     35696
weighted avg       0.99      0.99      0.99     35696

F1-score: 0.9854024494594184
```

Figure 24.

With an accuracy score of 0.9854605558045719, the model's accuracy is around 98.55%. precision is 0.98 for class 0 and 1.00 for class 1. Recall is 1.00 for class 0 and 0.96 for class 1. F1-Result: For both courses, the weighted average F1-score is around 0.9854. In the given dataset, support is the total number of real instances of the class. It is 22540 for class 0 and 13156 for class 1. F1-Score Weighted Average: 0.98540244494594184, this represents the weighted average of the F1-scores for the two classes, accounting for the unequal distribution of class sizes. It is around 0.9854, which shows that recall and accuracy are generally well balanced. All things considered, our XGBoost Classifier model works quite well, scoring highly in accuracy while maintaining a balance between precision and recall.

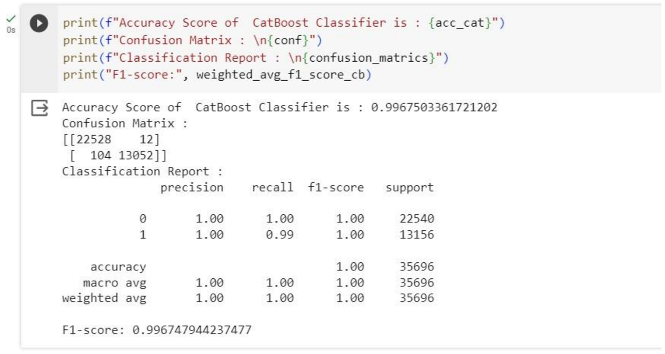
6.23.6. Cat Boost Classifier



```
cat = CatBoostClassifier(iterations=100)
cat.fit(X_train, y_train)

y_pred_cat = cat.predict(X_test)

acc_cat = accuracy_score(y_test, y_pred_cat)
conf = confusion_matrix(y_test, y_pred_cat)
confusion_matrics = classification_report(y_test, y_pred_cat)
clf_report = classification_report(y_test, y_pred_cat, output_dict=True)
report = pd.DataFrame(clf_report)
weighted_avg_f1_score_cb = clf_report['weighted avg']['f1-score']
```

```
10:	learn: 0.1584689	total: 1.2s	remaining: 9.7s
11:	learn: 0.1532297	total: 1.3s	remaining: 9.57s
12:	learn: 0.1458047	total: 1.43s	remaining: 9.58s
13:	learn: 0.1357746	total: 1.53s	remaining: 9.38s
14:	learn: 0.1345332	total: 1.62s	remaining: 9.19s
15:	learn: 0.1241890	total: 1.76s	remaining: 9.23s
16:	learn: 0.1191344	total: 1.87s	remaining: 9.14s
17:	learn: 0.1157997	total: 2.01s	remaining: 9.18s
18:	learn: 0.1099612	total: 2.15s	remaining: 9.17s
19:	learn: 0.1030575	total: 2.29s	remaining: 9.16s
20:	learn: 0.0989802	total: 2.36s	remaining: 8.89s
21:	learn: 0.0943404	total: 2.48s	remaining: 8.79s
22:	learn: 0.0903572	total: 2.55s	remaining: 8.54s
23:	learn: 0.0872668	total: 2.66s	remaining: 8.42s
24:	learn: 0.0850585	total: 2.79s	remaining: 8.37s
25:	learn: 0.0804339	total: 2.9s	remaining: 8.27s
26:	learn: 0.0778052	total: 3.04s	remaining: 8.21s
27:	learn: 0.0756512	total: 3.16s	remaining: 8.12s
```

Figure 25.



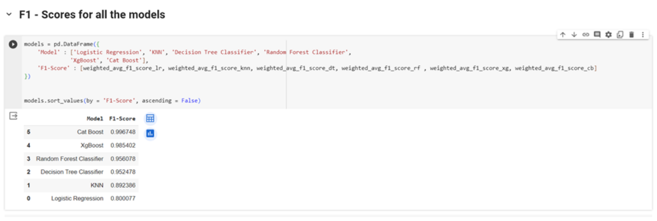
```
print(f"Accuracy Score of  CatBoost Classifier is : {acc_cat}")
print(f"Confusion Matrix : \n{conf}")
print(f"Classification Report : \n{confusion_matrics}")
print("F1-score:", weighted_avg_f1_score_cb)
```

```
Accuracy Score of  CatBoost Classifier is : 0.9967503361721202
Confusion Matrix :
[[22528    12]
 [  104 13052]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     22540
           1       1.00      0.99      1.00     13156

    accuracy                           1.00     35696
   macro avg       1.00      1.00      1.00     35696
weighted avg       1.00      1.00      1.00     35696

F1-score: 0.9967477944237477
```

Figure 26.

Figure 25,26. At around 99.68%, the CatBoost Classifier attained an extraordinarily high accuracy. Class 0: Perfect predictions are indicated by precision, recall, and F1-score of 1.00. For class 1, recall is somewhat below 1.00 (99.09%) but still represents excellent performance, with precision and F1-score of 1.00. The weighted and macro averages are both 1.00, demonstrating the outstanding overall performance of the model. F1-Score Weighted Average: 0.9967477944237477.The model's robustness is further supported by the weighted average F1-score, which is a very high value of around 99.67% and a balanced indicator of precision and recall.All things considered, the excellent scores obtained for the CatBoost Classifier indicate that it is incredibly accurate and precise in predicting cancellations of hotel reservations. The model appears to work really well with the dataset that was supplied.

6.24. F1 - Scores for all the models


F1 - Scores for all the models

```
models = pd.DataFrame({
    'Model' : ['Logistic Regression', 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier',
               'XgBoost', 'Cat Boost'],
    'F1-Score' : [weighted_avg_f1_score_lr, weighted_avg_f1_score_knn, weighted_avg_f1_score_dt, weighted_avg_f1_score_rf , weighted_avg_f1_score_xg, weighted_avg_f1_score_cb]
})


models.sort_values(by = 'F1-Score', ascending = False)
```

	Model	F1-Score
5	Cat Boost	0.996748
4	XgBoost	0.985402
3	Random Forest Classifier	0.956078
2	Decision Tree Classifier	0.952478
1	KNN	0.892386
0	Logistic Regression	0.800077

With a strong F1-score, the Random Forest model demonstrates its ability to manage the trade-off between recall and accuracy. While it works well, in this particular scenario CatBoost and XGBoost exceed it.

Classifier using Decision Trees: F1-Rank: 0.952478 The Decision Tree model has a competitive F1-score, indicating good performance. It appears to be marginally less efficient than previous one's.

KNN: F1-Rank: 0.892386 In comparison to tree-based models Here, ranking of the model performance is provided by the F1-scores, where CatBoost is at the top, followed by XGBoost, Random Forest, Decision Tree, KNN, and Logistic Regression.

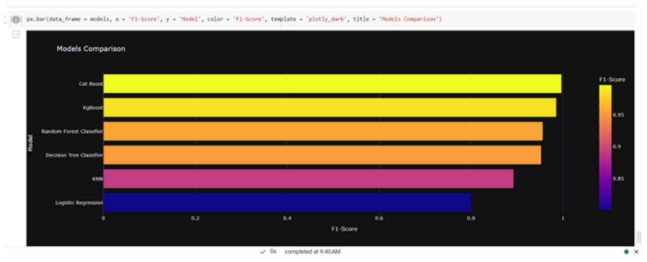
CatBoost:

F1-Score: 0.996748 With the greatest F1-score of all the models, CatBoost exhibits a superb trade-off between recall and accuracy. It implies more accuracy in anticipating both favorable and unfavorable events.

XGBoost: F1-Score: 0.985402 With a high F1-score, XGBoost likewise performs admirably, but little worse than CatBoost. It shows excellent categorization task performance with a well-balanced accuracy and recall.

The Random Forest Classifier F1-Rank: 0.956078 KNN (K-Nearest Neighbors) has a lower F1-score, indicating that it might not generalize to this particular dataset as well. The kind of data typically determines how successful KNN is.

The Logistic Regression Model F1-Rank: 0.800077 Out of all the models given, Logistic Regression has the lowest F1-score. Even though it's a potentially very strong algorithm, it might not be able to identify intricate links in the data as well as ensemble approaches like XGBoost and CatBoost.

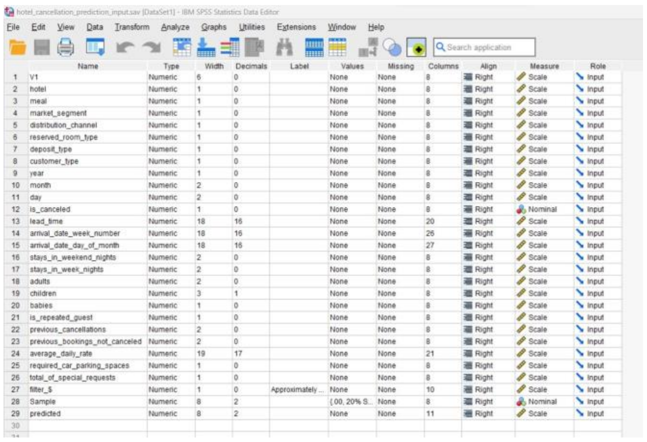


```
px.bar(data_frame = models, x = 'F1-Score', y = 'Model', color = 'F1-Score', template = 'plotly_dark', title = 'Models Comparison')
```

Figure 27.

7. Cross Validation Using SPSS:

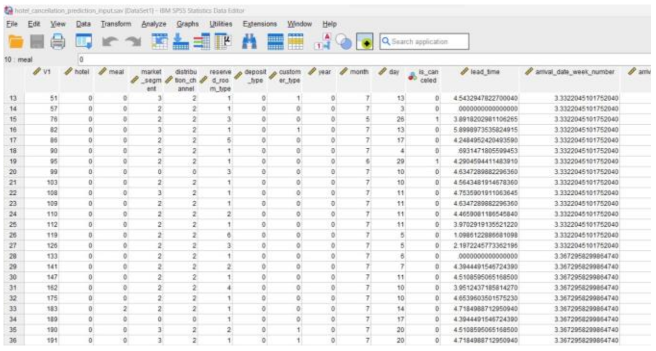
The processed data from pandas is exported as csv and imported into SPSS. Then split the data as 20% test data and 80% train data. Build linear regression model and from Pearson Correlation value of 0.578 for 20% sample (Testing) and 0.573 for 80% Sample (Training). So, the values are almost same. Hence, we can conclude that our model is neither underfitting nor overfitting.


hotel_cancellation_prediction_input.sav [DataSet1] - IBM SPSS Statistics Data Editor

File Edit View Data Transform Analyze Graphs Utilities Extensions Window Help
Search application

	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	V1	Numeric	6	0		None	None	8	Right	Scale	Input
2	hotel	Numeric	1	0		None	None	8	Right	Scale	Input
3	meal	Numeric	1	0		None	None	8	Right	Scale	Input
4	market_segment	Numeric	1	0		None	None	8	Right	Scale	Input
5	distribution_channel	Numeric	1	0		None	None	8	Right	Scale	Input
6	reserved_room_type	Numeric	1	0		None	None	8	Right	Scale	Input
7	deposit_type	Numeric	1	0		None	None	8	Right	Scale	Input
8	customer_type	Numeric	1	0		None	None	8	Right	Scale	Input
9	year	Numeric	1	0		None	None	8	Right	Scale	Input
10	month	Numeric	2	0		None	None	8	Right	Scale	Input
11	day	Numeric	2	0		None	None	8	Right	Scale	Input
12	is_canceled	Numeric	1	0		None	None	8	Right	Nominal	Input
13	lead_time	Numeric	18	16		None	None	20	Right	Scale	Input
14	arrival_date_week_number	Numeric	18	16		None	None	26	Right	Scale	Input
15	arrival_date_day_of_month	Numeric	18	16		None	None	27	Right	Scale	Input
16	stays_in_weekend_nights	Numeric	2	0		None	None	8	Right	Scale	Input
17	stays_in_week_nights	Numeric	2	0		None	None	8	Right	Scale	Input
18	adults	Numeric	2	0		None	None	8	Right	Scale	Input
19	children	Numeric	3	1		None	None	8	Right	Scale	Input
20	babies	Numeric	1	0		None	None	8	Right	Scale	Input
21	is_repeated_guest	Numeric	1	0		None	None	8	Right	Scale	Input
22	previous_cancellations	Numeric	2	0		None	None	8	Right	Scale	Input
23	previous_bookings_not_canceled	Numeric	2	0		None	None	8	Right	Scale	Input
24	average_daily_rate	Numeric	19	17		None	None	21	Right	Scale	Input
25	required_car_parking_spaces	Numeric	1	0		None	None	8	Right	Scale	Input
26	total_of_special_requests	Numeric	1	0		None	None	8	Right	Scale	Input
27	filter_$	Numeric	1	0	Approximately ...	None	None	10	Right	Scale	Input
28	Sample	Numeric	8	2		{.00, 20% S...	None	8	Right	Nominal	Input
29	predicted	Numeric	8	2		None	None	11	Right	Scale	Input
30											

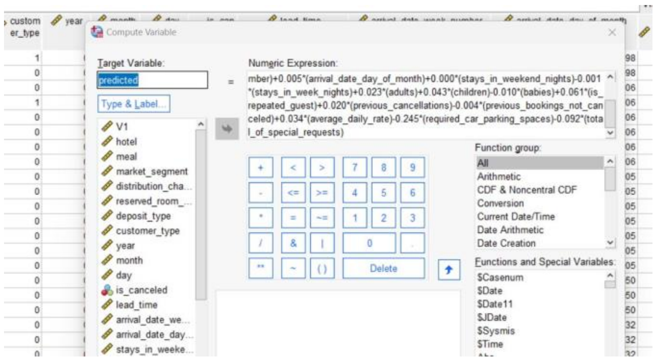
Figure 28,29 and 30: Inputs of SPSS.


hotel_cancellation_prediction_input.sav [DataSet1] - IBM SPSS Statistics Data Editor

File Edit View Data Transform Analyze Graphs Utilities Extensions Window Help
Search application

10 : meal 0

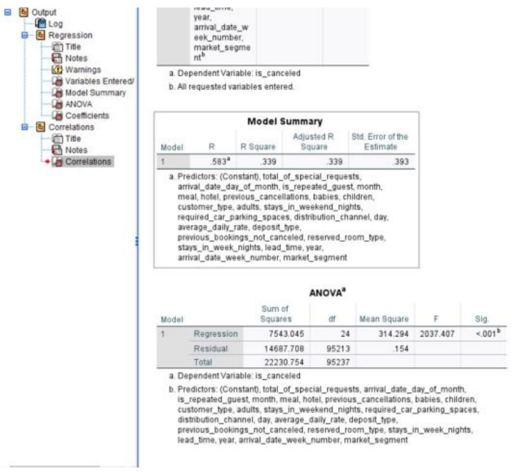
	V1	hotel	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type	year	month	day	is_canceled	lead_time	arrival_date_week_number
13	51	0	0	3	2	1	0	1	0	7	13	0	4.5432947822700040	3.3322045101752040
14	57	0	0	2	2	1	0	0	0	7	3	0	.0000000000000000	3.3322045101752040
15	76	0	0	2	2	3	0	0	0	5	26	1	3.8918202981106265	3.3322045101752040
16	82	0	0	3	2	1	0	1	0	7	13	0	5.8998973535824915	3.3322045101752040
17	86	0	0	2	2	5	0	0	0	7	17	0	4.2484952420493590	3.3322045101752040
18	90	0	0	2	2	1	0	0	0	7	4	0	.6931471805599453	3.3322045101752040
19	95	0	0	2	2	1	0	0	0	6	29	1	4.2904594411483910	3.3322045101752040
20	99	0	0	0	0	3	0	0	0	7	10	0	4.6347289882296360	3.3322045101752040
21	103	0	0	2	2	1	0	0	0	7	10	0	4.5643481914678360	3.3322045101752040
22	108	0	0	3	2	1	0	0	0	7	11	0	4.7535901911063645	3.3322045101752040
23	109	0	0	2	2	1	0	0	0	7	11	0	4.6347289882296360	3.3322045101752040
24	110	0	0	2	2	2	0	0	0	7	11	0	4.4659081186545840	3.3322045101752040
25	112	0	0	2	2	1	0	0	0	7	11	0	3.9702919135521220	3.3322045101752040
26	119	0	0	2	2	6	0	0	0	7	5	0	1.0986122886681098	3.3322045101752040
27	126	0	0	2	2	3	0	0	0	7	5	0	2.1972245773362196	3.3322045101752040
28	133	0	0	2	2	1	0	0	0	7	6	0	.0000000000000000	3.3672958299864740
29	141	0	0	2	2	2	0	0	0	7	7	0	4.3944491546724390	3.3672958299864740
30	147	0	0	2	2	1	0	0	0	7	11	0	4.5108595065168500	3.3672958299864740
31	162	0	0	2	2	4	0	0	0	7	10	0	3.9512437185814270	3.3672958299864740
32	175	0	0	2	2	1	0	0	0	7	10	0	4.6539603501575230	3.3672958299864740
33	183	0	2	2	2	1	0	0	0	7	14	0	4.7184988712950940	3.3672958299864740
34	189	0	0	0	0	1	0	0	0	7	17	0	4.3944491546724390	3.3672958299864740
35	190	0	0	3	2	2	0	1	0	7	20	0	4.5108595065168500	3.3672958299864740
36	191	0	0	3	2	1	0	1	0	7	20	0	4.7184988712950940	3.3672958299864740


Compute Variable

Target Variable: predicted

Numeric Expression:

```
mber)+0.005*(arrival_date_day_of_month)+0.000*(stays_in_weekend_nights)-0.001
*(stays_in_week_nights)+0.023*(adults)+0.043*(children)-0.010*(babies)+0.061*(is_
repeated_guest)+0.020*(previous_cancellations)-0.004*(previous_bookings_not_can
celed)+0.034*(average_daily_rate)-0.245*(required_car_parking_spaces)-0.092*(tota
l_of_special_requests)
```



year,
arrival_date_w
eek_number,
market_segme
nt[b]

a. Dependent Variable: is_canceled
b. All requested variables entered.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.583[a]	.339	.339	.393

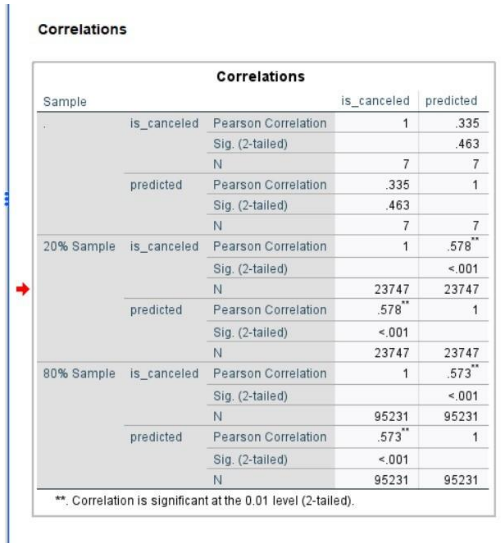
a. Predictors: (Constant), total_of_special_requests, arrival_date_day_of_month, is_repeated_guest, month, meal, hotel, previous_cancellations, babies, children, customer_type, adults, stays_in_weekend_nights, required_car_parking_spaces, distribution_channel, day, average_daily_rate, deposit_type, previous_bookings_not_canceled, reserved_room_type, stays_in_week_nights, lead_time, year, arrival_date_week_number, market_segment

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	7543.045	24	314.294	2037.407	<.001[b]
	Residual	14687.708	95213	.154		
	Total	22230.754	95237			

a. Dependent Variable: is_canceled
b. Predictors: (Constant), total_of_special_requests, arrival_date_day_of_month, is_repeated_guest, month, meal, hotel, previous_cancellations, babies, children, customer_type, adults, stays_in_weekend_nights, required_car_parking_spaces, distribution_channel, day, average_daily_rate, deposit_type, previous_bookings_not_canceled, reserved_room_type, stays_in_week_nights, lead_time, year, arrival_date_week_number, market_segment

Figure 31: Output of SPSS.



Correlations

Correlations

Sample			is_canceled	predicted
.	is_canceled	Pearson Correlation	1	.335
		Sig. (2-tailed)		.463
		N	7	7
	predicted	Pearson Correlation	.335	1
		Sig. (2-tailed)	.463	
		N	7	7
20% Sample	is_canceled	Pearson Correlation	1	.578**
		Sig. (2-tailed)		<.001
		N	23747	23747
	predicted	Pearson Correlation	.578**	1
		Sig. (2-tailed)	<.001	
		N	23747	23747
80% Sample	is_canceled	Pearson Correlation	1	.573**
		Sig. (2-tailed)		<.001
		N	95231	95231
	predicted	Pearson Correlation	.573**	1
		Sig. (2-tailed)	<.001	
		N	95231	95231

**. Correlation is significant at the 0.01 level (2-tailed).

Figure 32: Output of correlations.

8. Concerns:

Even we got a promising results, there are many concerns about this system. The main concern about the data set is it is not capturing all the influencing factors that affect the cancellations of the hotel. There are null values and handling these null values is more complex part. The relationship between the hotel and the cancellations of the hotels by the guests, capturing these are very important and their should be continuous retraining of the models as the data changes time to time. The overfitting of the models is one of the challenging part in this project as it impacts the outliers for the predictions.

9. Conclusion:

This project gives the predictions about the cancellations for the two hotels. Here we have used multiple ML models such as Logistic Regression, catboost, KNN, XGboost, Decision Tree, Random Forest. We have done the exploratory data analysis where we have analysed different sectors which impacts the hotel booking cancellation forecasting, first of all we have removed the null values. We have categories the guest based on different countries. We have determine our guest based on various market sectors and we have also categories the guess who booked our hotels like from online off-line or through other third-party sites etc, we have categories according to the month in which month the cancellations are more. Based on all these factors we have visualised, we have analysed and then we have built a ML models, we have used different algorithms like Logistic Regression, catboost, KNN, XGboost, Decision Tree, Random Forest. Here we got highest accuracy for catboost model with 99%. By all these above EDA and models, we got a valuable insights, which are influencing the cancellations of hotel bookings. We have compared all the models. Here we have also used SPSS for predicting the hotel booking cancellations. The target variable is predicted where it predicts whether the booking is cancelled or not. We are also having the model summary and we have used Anova. We have also got the correlations. Here we have taken sample size as 20% and 80% and we have calculated the correlations. We have also got the unstandardised coefficient and standardised coefficient, standard error significance. This predictive system will be like a framework or a system where we can help hotel management to increase the experience of the hotel. We can optimise the operations in the hotels and how to improve the system and the facilities provided by the hotels.

10. Implementation status report:

10.1. Work completed:

- Dataset understanding - All 4
- Dataset preprocessing - Numitha 50, Ketha 50
- EDA Analsysis - Sheela 50 Yamuna 50
- Understanding the Correlation - All 4 teammates equally
- Model Creation - Logistic Regression - Numitha ketha equally
- KNN - Sheela Yamuna equally
- cat boost - sheela
- XG boost - Numitha
- Decision Tree classifier - Ketha
- Random forest classifier - Yamuna
- Model Comparision and Evaluation - Ketha
- SPSS - Yamuna
- Report - All teammates contribuited

11. References:

[1]. Shuixia Chen et al., ”Prediction of hotel booking cancellations: Integration of machine learning and probability model based on interpretable feature interaction”, 18 May 2023.

[2]. Pranav Kumar, ”Hotel Booking Prediction using Machine Learning”, Volume 10 Issue V May 2022.

[3]. Link- https://www.kaggle.com/code/sanjana08/hotel-booking-cancellation-prediction/input

[4]. https://www.sciencedirect.com/topics/computer-science/logistic-regression: :text=Logistic

[5]. https://www.ibm.com/topics/knn: :text=Resources-,Next

[6]. https://www.ibm.com/topics/knn: :text=Resources-,Next

[7]. Nugroho Adi Putro et al.,”PREDICTION OF HOTEL BOOKING CANCELLATION USING DEEP NEURAL NETWORK AND LOGISTIC REGRESSION ALGORITHM”, March 2021.

12. Github link:

https://github.com/tketha/hotel-booking-emperical-project-group-15