HOTEL BOOKING CANCELLATION

MACHINE LEARNING

I.SCENARIO

II.DATASET OVERVIEW

III. PREPROCESSING DATA

IV.ANALYSIS AND FINDINGS

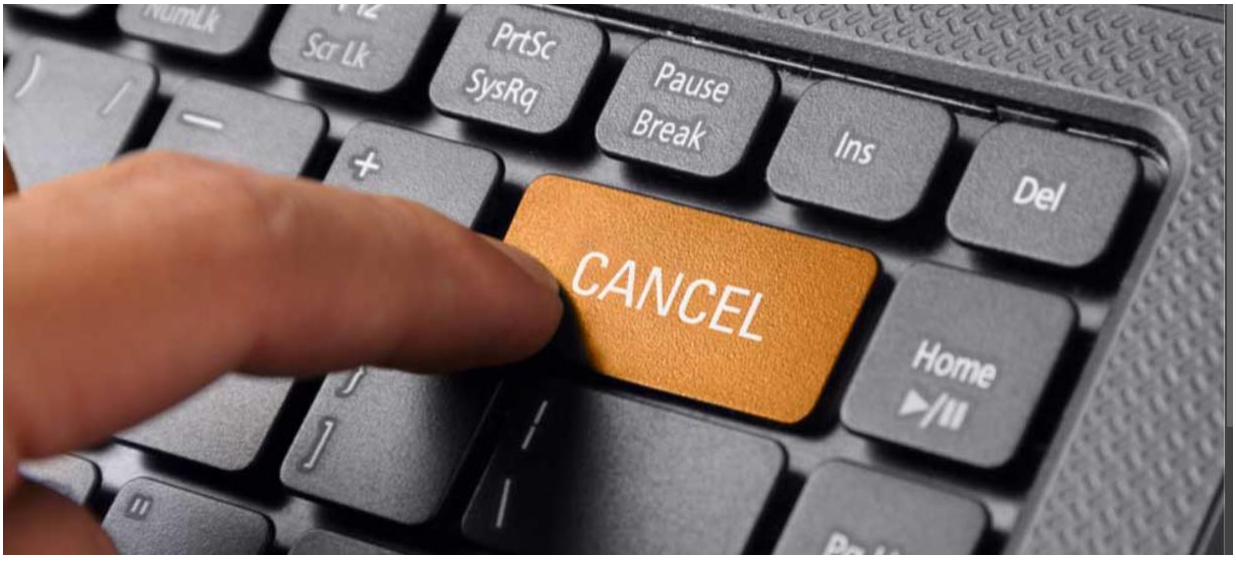
V.MACHINE LEARNING

IV. SUGGESTIONS

I.SCENARIO

1. Business Problem

You are a Data Analyst at the famous city hotel named The Continental. The company is presently engaged in the business of operating City Hotel and Resort Hotel. But in the recent year, The hotel has witnessed high cancellation rates. Each of the branch hotels is now dealing with a number of issues as a result, including a significant decline in revenues and less than ideal hotel room use. As a result, lowering the cancellation rates is the highest priority of the branch hotels to increase their efficiency in generating revenue and for us to offer thorough business advice to address this problem.



2. Assumptions

a) No unusual occurrences between 2015 and 2017 will substantially impact the data used.
b) The information is still current and can be used to analyze a hotel's possible plans.
c) There are no unanticipated negatives to the hotel.
d) The most significant factor affecting the effectiveness of earning income is booking cancellations.
e) The hotels are not currently using any of the suggested solutions.

3. Research Question

a) What are the factors that affect hotel reservation cancellations?

b) How can we make hotel reservation cancellations better?

c) How will hotels be assisted in making pricing and promotional decisions?

II.DATASET OVERVIEW

1. Introduction

The data set source :*https://www.kaggle.com/jessemostipak/hotel-booking-demand*

This data set consists of ***119,390*** observations and holds booking data for a city hotel and a resort hotel from 2015 to 2017. It has 32 variables which include *reservation and arrival date, length of stay, canceled or not, the number of adults, children, or babies, the number of available parking spaces, how many special guests, companies, and agents pushed the reservation*, etc.

2. Details of Data

hotel :(H1 = Resort Hotel or H2 = City Hotel).	**reserved_room_type**: Code of room type reserved. Code is presented instead of designation for anonymit reasons.
is_canceled Value: showing if the booking hac been cancelled (1) or not (0).	**assigned_room_typ**e: Code for the type of room assigned to the booking. Code is presented instead of designation for anonymity reasons.
lead_time: Number of days that elapsed betwee the entering date of the booking into the PMS and the arrival date.	**booking_changes**: How many times did booking changes happen
arrival_date_year: Year of arrival date.	**deposit_type**: Indication on if the customer deposited something to confirm the booking.
arrival_date_month: The months in which guests are coming.	**agent:** If the booking happens through agents or not.
arrival_date_week_number: Week number of year for arrival date.	**company**: If the booking happens through companies, the company ID that made the booking or responsible for paying the booking.
arrival_date_day_of_month: Which day of th months guest is arriving.	**days_in_waiting_list**: Number of days the booking was on the waiting list before the confirmation to the customer.

stays_in_weekend_nights: Number of weeken stay at night (Saturday or Sunday) the guest stayed or booked to stay at the hotel.	**customer_type**: Booking type like Transient – Transient-Party – Contract – Group.
stays_in_week_nights: Number of weekdays stay at night (Monday to Friday) in the hotel.	**adr**: Average Daily Rates that described via way of means of dividing the sum of all accommodations transactions using entire numbers of staying nights.
adults: Number of adults.	**required_car_parking_spaces**: How many parking areas are necessary for the customers.
children: Number of children.	**total_of_special_requests**: Total unique requests fron consumers.
babies: Number of babies.	**reservation_status**: The last status of reservation, assuming one of three categories: Canceled – booking was cancelled by the customer; Check-Out
meal: Type of meal booked.	**reservation_status_date**: The last status date.
country: Country of origin.	**is_repeated_guest**: The values indicating if the booking name was from a repeated guest (1) or not (0
market_segment: Through which channel hote were booked.	**previous_cancellations:** Show if the repeated guest has cancelled the booking before.
distribution_channel: Booking distribution channel.	**previous_bookings_not_canceled**: Show if the repeated guest has not cancelled the booking before.

III PREPROCESSING DATA

Data preprocessing is a data mining technique used to transform raw data into a useful and efficient format. The team primarily performs data preprocessing tasks such as filling in missing values, removing duplicate data, handling outliers, data transformation, data splitting, and feature scaling.

1. **Import Libraries**

   After collecting data we have to import the necessary libraries to build machine learning models. Numpy, Pandas, Matploilib, and Seaborn are the known libraries used in the machine learning model.

   + **Numpy:** NumPy is a Numerical Python Library that helps perform mathematical operations.

   + Pandas: Panda is an open-source library that helps understand relational or labelled data.

Importing Libraries

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    import numpy as np
    warnings.filterwarnings('ignore')
```

The data shown below represents the Hotel booking cancellation dataset that is available on Kaggle

```
[2] df = pd.read_csv('/content/hotel_bookings 2.csv')
```

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

2. Data Cleaning

Real-world raw data is often incomplete, inconsistent, and lacking in certain behaviors or trends. The raw data contain many errors. So, once collected, the next step machine learning pipeline is to clean data which refers to removing unwanted data.

Some steps which are used to clean data are:

- Remove missing values, outliers, and unnecessary rows/ columns.
- Check and impute null values.
- Check Imbalanced data.
- Re-indexing and reformatting our data.

First, have to check if our data holds duplicate values.

Pandas **duplicated()** method helps to return duplicate values only, and any() method returns a boolean series that is True only if the conditions are true.

```
#Check if data holds duplicate values.
df.duplicated().any()
```

```
True
```

The dataset holds the duplicate entries.

Pandas **drop_duplicate()** method helps to remove duplicate entries. The keep parameter is set to False so that only Unique values are taken and the duplicate values are removed from the data.

```
#Drop Duplicate entries
df.drop_duplicates(inplace= False)
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	
...	...	...	...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	35	30	2	
119386	City Hotel	0	102	2017	August	35	31	2	

Secondly, we should check the number of missing values in each column and the percentage of missing values in the columns. Pandas **isna()** method helps to find out null values and also convert the result to the percentage for review purpose

```
df.isnull().sum()
```

```
null =100 *(df.isna().sum() /df.shape[0])
null
```

hotel	0
is_canceled	0
lead_time	0
arrival_date_year	0
arrival_date_month	0
arrival_date_week_number	0
arrival_date_day_of_month	0
stays_in_weekend_nights	0
stays_in_week_nights	0
adults	0
children	4
babies	0
meal	0
country	488
market_segment	0
distribution_channel	0
is_repeated_guest	0
previous_cancellations	0
previous_bookings_not_canceled	0
reserved_room_type	0
assigned_room_type	0
booking_changes	0
deposit_type	0
agent	16340
company	112593
days_in_waiting_list	0
customer_type	0
adr	0
required_car_parking_spaces	0
total_of_special_requests	0
reservation_status	0
reservation_status_date	0

hotel	0.000000
is_canceled	0.000000
lead_time	0.000000
arrival_date_year	0.000000
arrival_date_month	0.000000
arrival_date_week_number	0.000000
arrival_date_day_of_month	0.000000
stays_in_weekend_nights	0.000000
stays_in_week_nights	0.000000
adults	0.000000
children	0.003350
babies	0.000000
meal	0.000000
country	0.408744
market_segment	0.000000
distribution_channel	0.000000
is_repeated_guest	0.000000
previous_cancellations	0.000000
previous_bookings_not_canceled	0.000000
reserved_room_type	0.000000
assigned_room_type	0.000000
booking_changes	0.000000
deposit_type	0.000000
agent	13.686238
company	94.306893
days_in_waiting_list	0.000000
customer_type	0.000000
adr	0.000000
required_car_parking_spaces	0.000000
total_of_special_requests	0.000000
reservation_status	0.000000
reservation_status_date	0.000000

Column company holds 94% missing data, so we can drop that column.

```
df.drop(['company', 'agent'], axis = 1, inplace = True)
df.dropna(inplace = True)
```

Now we check the result after dropped

```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        0
babies                          0
meal                            0
country                         0
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
deposit_type                    0
days_in_waiting_list            0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
reservation_status_date         0
```

Finally, before analysts make a prediction, the data should have been encoded and categorized.

Dropping columns that are not useful :

```
# Dropping columns that are not useful
useless_col = ['days_in_waiting_list','arrival_date_year','assigned_room_type','booking_changes',
               'reservation_status','country', 'days_in_waiting_list']
df.drop(useless_col,axis =1 , inplace = True )
```

Conducting categorization by following the codes :

```
# Getting all the categorical columns from the dataframe
cat_cols = [col for col in df.columns if df[col].dtype == 'O']

# Displaying the categorical columns
print(cat_cols)

# Creating a new dataframe with only the categorical columns
cat_df = df[cat_cols]

# Displaying the head of the new dataframe
cat_df.head()
```

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

	hotel	arrival_date_month	meal	market_segment	distribution_channel	reserved_room_type	deposit_type	customer_type
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

City Hotel and Resort Hotel are string types, so we need to encode categorical variables

The results after encoding:

```
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})
```

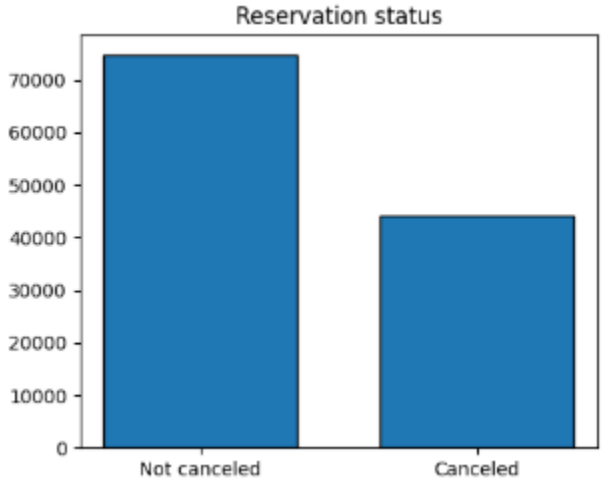
```
cat_df.head()
```

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

The Dataset is ready to Analyst and Predict.

IV ANALYSIS AND FINDINGS

1. Reservation status



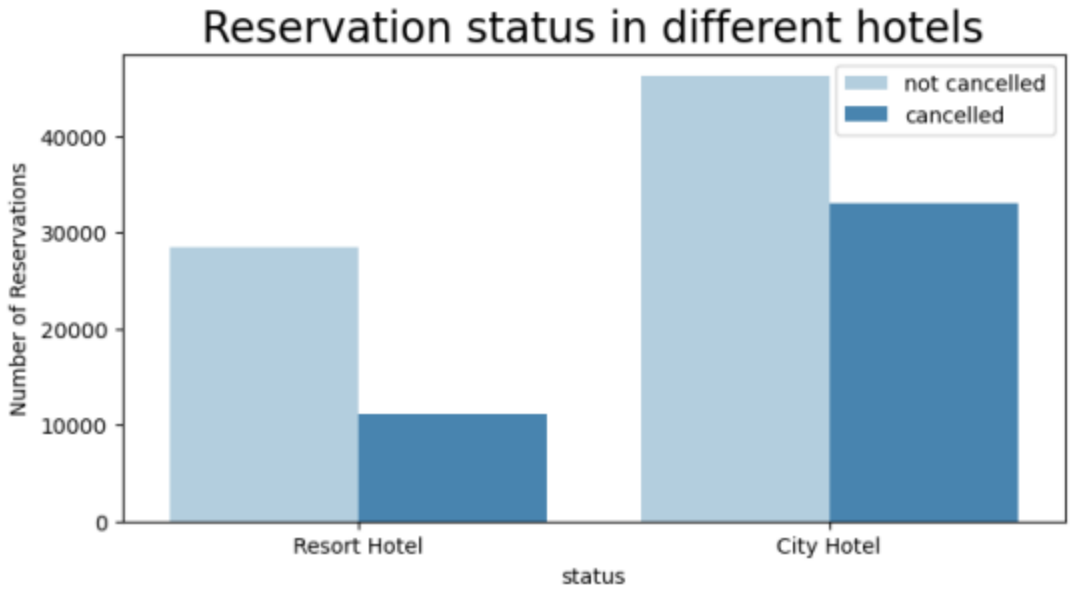

```
0    0.628653
1    0.371347
```

0 : not canceled

1 : canceled

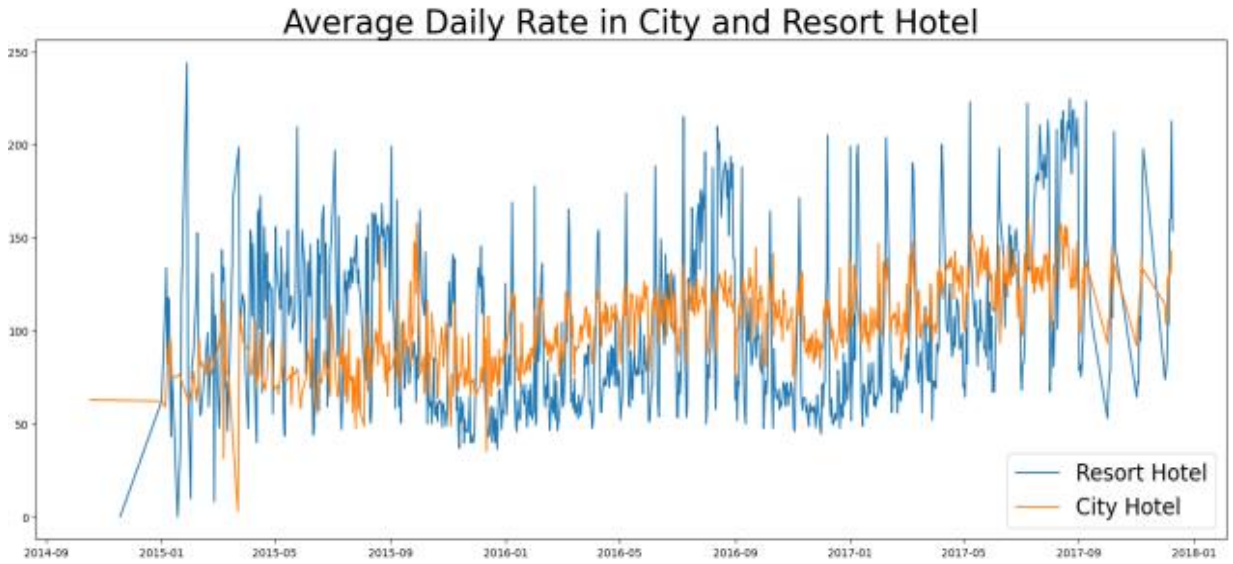
The accompanying bar graph shows the percentage of reservations that are canceled and those that are not. It is obvious that there are still a significant number of reservations that have not been canceled. There are still **37% of clients who canceled** their reservations, which has a significant impact on the hotels' earnings.

2. Reservation status in different hotels



Compared to the Resort Hotel, it's evident that the City Hotel has a higher volume of bookings, which consequently leads to a larger number of cancellations. Consequently, there has been a substantial decrease in booking revenue, particularly from the City Hotel, where the number of cancellations has exceeded 30,000 times.

3. Average Daily Rate in City and Resort Hotel



adr: Average Daily Rates that described via way of means of dividing the sum of all accommodations transactions using entire numbers of staying nights

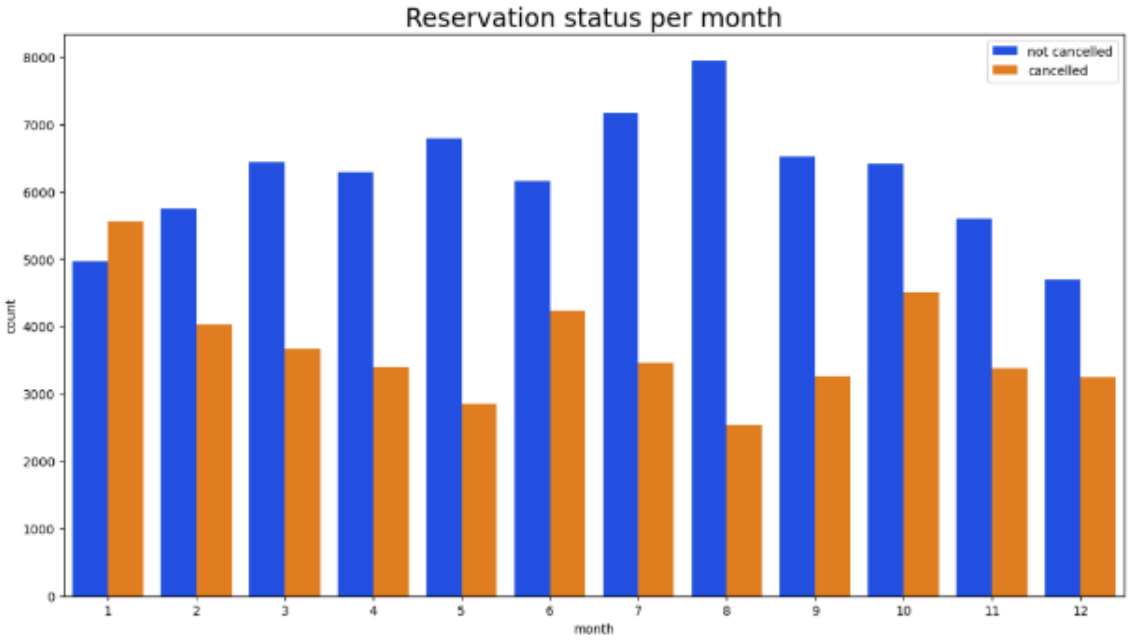
To calculate the ADR, you would add up the total revenue generated from all the room bookings and then divide it by the total number of nights that guests stayed in those rooms. This gives you an average cost per room per day. It's a useful metric because it helps hotel managers and analysts understand the ***average pricing of their rooms and track trends in pricing over time.***

$$ADR = \frac{Total\ Room\ Revenue}{Total\ number\ of\ staying\ nights}$$

By calculating the ADR, hotels can assess their pricing strategies, monitor changes in customer demand, and make informed decisions about their room rates to maximize revenue and profitability.

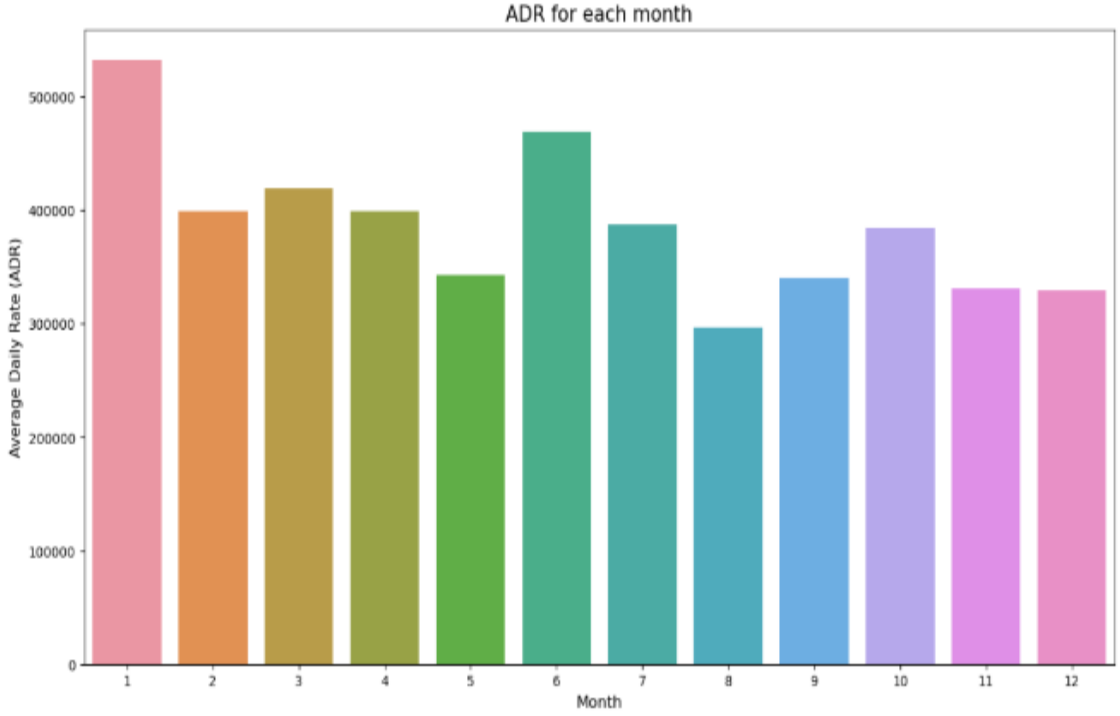
According to the line chart above that illustrates the daily rate for a City hotel is occasionally lower than Resort Hotel. Furthermore, there are circumstances where the City Hotel's even lower nearly reached **0%** Average Daily Rate around in March and April 2015. It is evident that weekends and holidays could witness an increase in the rates for Resort Hotel.

4. Reservation per month



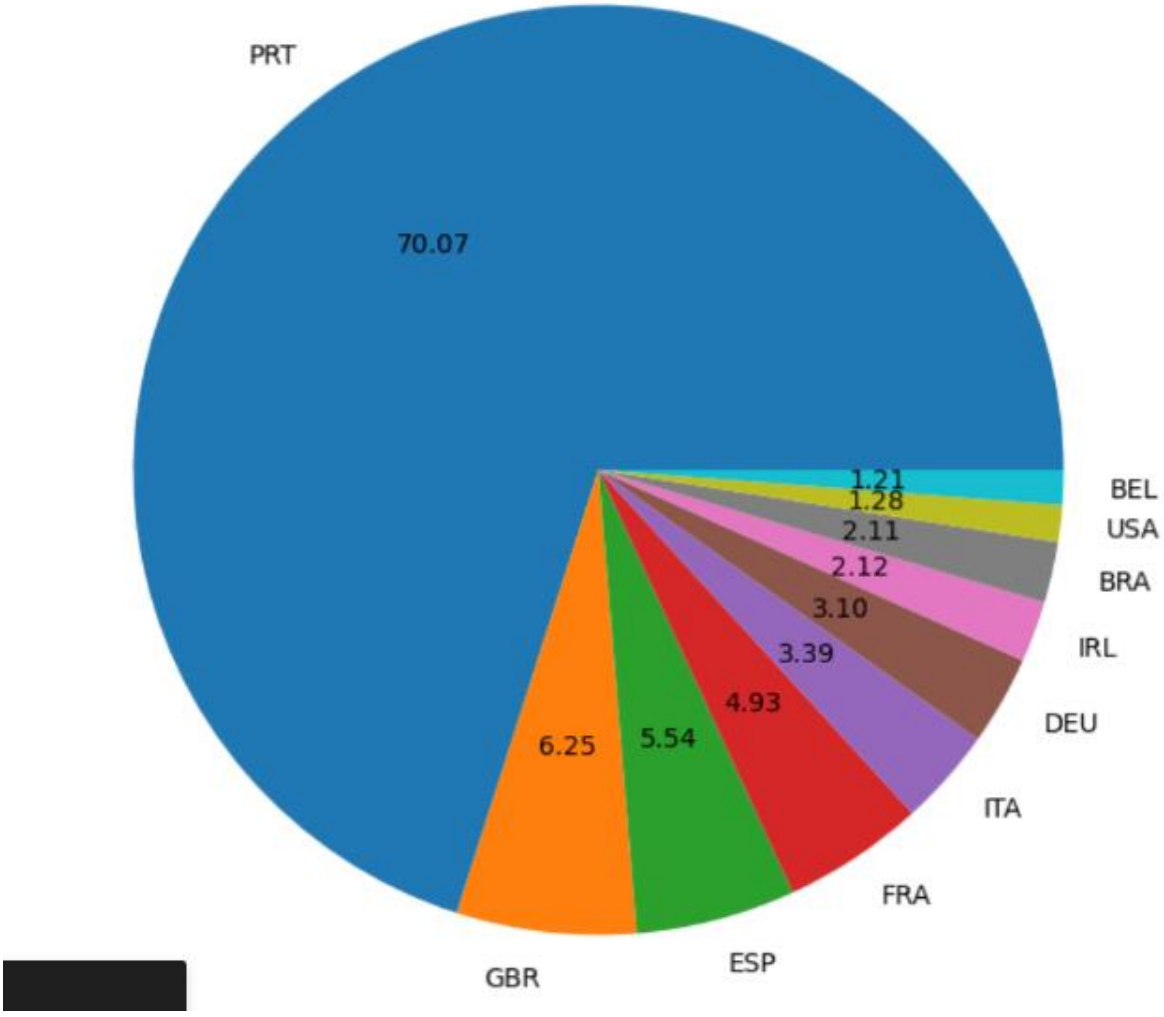
The grouped bar graph examines the greatest and lowest reservation levels by status of reservations for each month. The data illustrated that the month of August exhibits the highest figures, reached at the top with **8000** for confirmed bookings and the lowest for cancellations. Conversely, it is notable that the month of January stands out due to having the highest count of cancellations of bookings with nearly **6000 cancellations.**

5. ADR for each month



The depicted bar graph provides clear evidence that instances of cancellations are observed when **accommodation prices reach their highest points**, **while occurrences of cancellations diminish notably when prices are at their lowest**. Hence, it can be deduced that the sole determinant for these cancellations is the **cost associated** with the accommodations.

6. Top 10 countries with reservations cancelled



Based on the information presented by the provided pie chart, it is discernible that Portugal emerges as the leading country, contributing to **70%** of the total canceled reservations.

7. Market segment

```
df['market_segment'].value_counts()
```

```
Online TA         56402
Offline TA/TO     24159
Groups            19806
Direct            12448
Corporate          5111
Complementary       734
Aviation            237
Name: market_segment, dtype: int64
```

```
[28] df['market_segment'].value_counts(normalize = True)
```

```
Online TA         0.474377
Offline TA/TO     0.203193
Groups            0.166581
Direct            0.104696
Corporate         0.042987
Complementary     0.006173
Aviation          0.001993
Name: market_segment, dtype: float64
```

The area from where guests visit the hotels and make reservations.

Is it coming from Direct or Groups, Online or Offline Travel Agents? The given dataset shows that **20%** of customers come from traditional travel agents and **47%** from online travel agencies respectively come from **offline travel agencies**. whereas 33% of customers are from untapped markets.

V MACHINE LEARNING

1. Model Building

The 4 models were applied in my research to forecast and assess the cancellation.

+ **KNN**

+ **Decision Tree**

+ **Random Forest**

+ **Gradient Boosting**

Import necessary libraries:

```
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 GradientBoostingClassifier
```

```
X = pd.concat([cat_df, num_df], axis=1)
y = df['is_canceled']
```

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

a) KNeighbors regression

```
# KNeighbors
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_knn= confusion_matrix(y_test, y_pred_knn)
clf_report = classification_report(y_test, y_pred_knn)



print(f"Accuracy Score of KNN is : {acc_knn}")
print(f"Confusion Matrix : n{conf}")
print(f"Classification Report : n{clf_report}")
```

```
Accuracy Score of KNN is : 0.8831916956596133
Confusion Matrix : n[[9748  632]
 [1416 5737]]
Classification Report : n              precision    recall  f1-score   support

           0       0.87      0.94      0.90     10380
           1       0.90      0.80      0.85      7153

    accuracy                           0.88     17533
   macro avg       0.89      0.87      0.88     17533
weighted avg       0.88      0.88      0.88     17533
```

b) Decision Tree

```
# Decision Tree
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)
clf_report = classification_report(y_test, y_pred_dtc)

print(f"Accuracy Score of Decision Tree is : {acc_dtc}")
print(f"Confusion Matrix : n{conf}")
print(f"Classification Report : n{clf_report}")
```

```
Accuracy Score of Decision Tree is : 0.9182684081446415
Confusion Matrix : n[[9668  712]
 [ 721 6432]]
Classification Report : n              precision    recall  f1-score   support

           0       0.93      0.93      0.93     10380
           1       0.90      0.90      0.90      7153

    accuracy                           0.92     17533
   macro avg       0.92      0.92      0.92     17533
weighted avg       0.92      0.92      0.92     17533
```

c) Random Forest

```
# Random Forest
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)
clf_report = classification_report(y_test, y_pred_rd_clf)

print(f"Accuracy Score of Random Forest is : {acc_rd_clf}")
print(f"Confusion Matrix : n{conf}")
print(f"Classification Report : n{clf_report}")
```

```
Accuracy Score of Random Forest is : 0.9624137341014087
Confusion Matrix : n[[10267   113]
 [  546  6607]]
Classification Report : n              precision    recall  f1-score   support

           0       0.95      0.99      0.97     10380
           1       0.98      0.92      0.95      7153

    accuracy                           0.96     17533
   macro avg       0.97      0.96      0.96     17533
weighted avg       0.96      0.96      0.96     17533
```

d) Gradient Boosting

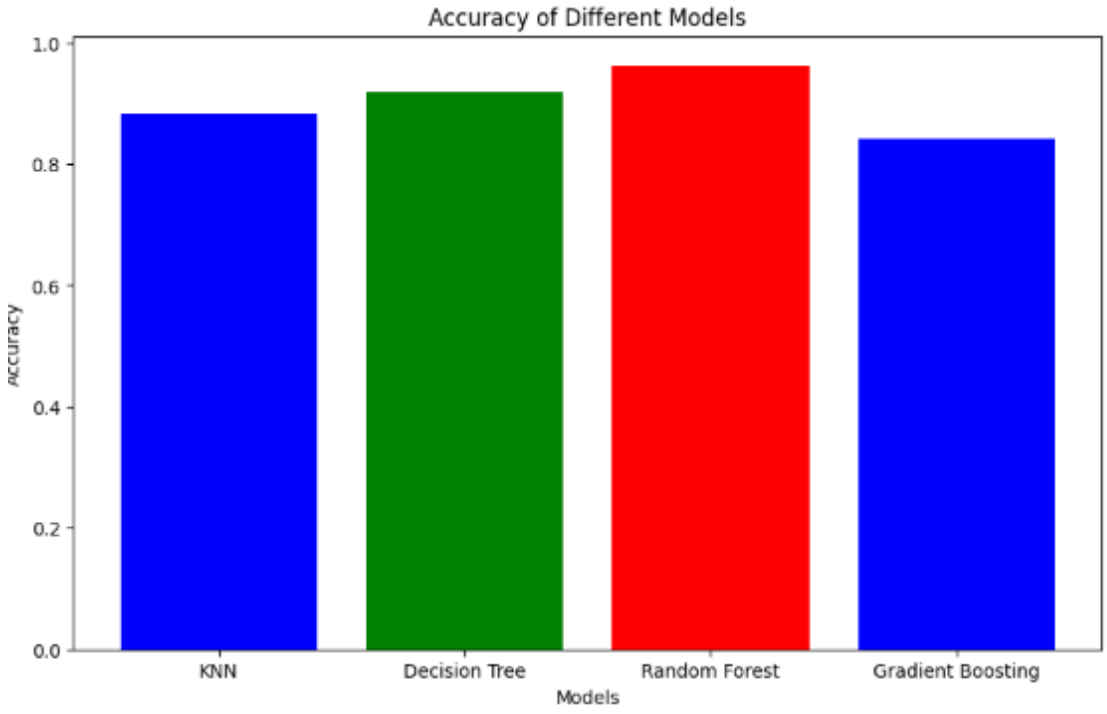
```
# Initialize Gradient Boosting Classifier
gb_clf = GradientBoostingClassifier()

# Train the model
gb_clf.fit(X_train, y_train)

# Predict the test set results
y_pred_gb = gb_clf.predict(X_test)

# Calculate performance metrics
acc_gb = accuracy_score(y_test, y_pred_gb)
conf_gb = confusion_matrix(y_test, y_pred_gb)
clf_report_gb = classification_report(y_test, y_pred_gb)

# Print out the results
print(f"Accuracy Score of Gradient Boosting is : {acc_gb}")
print(f"Confusion Matrix : \n{conf_gb}")
print(f"Classification Report : \n{clf_report_gb}")
```

```
Accuracy Score of Gradient Boosting is : 0.8417270290309702
Confusion Matrix :
[[9560  820]
 [1955 5198]]
Classification Report :
              precision    recall  f1-score   support

           0       0.83      0.92      0.87     10380
           1       0.86      0.73      0.79      7153

    accuracy                           0.84     17533
   macro avg       0.85      0.82      0.83     17533
weighted avg       0.84      0.84      0.84     17533
```

2. Visualizing the results

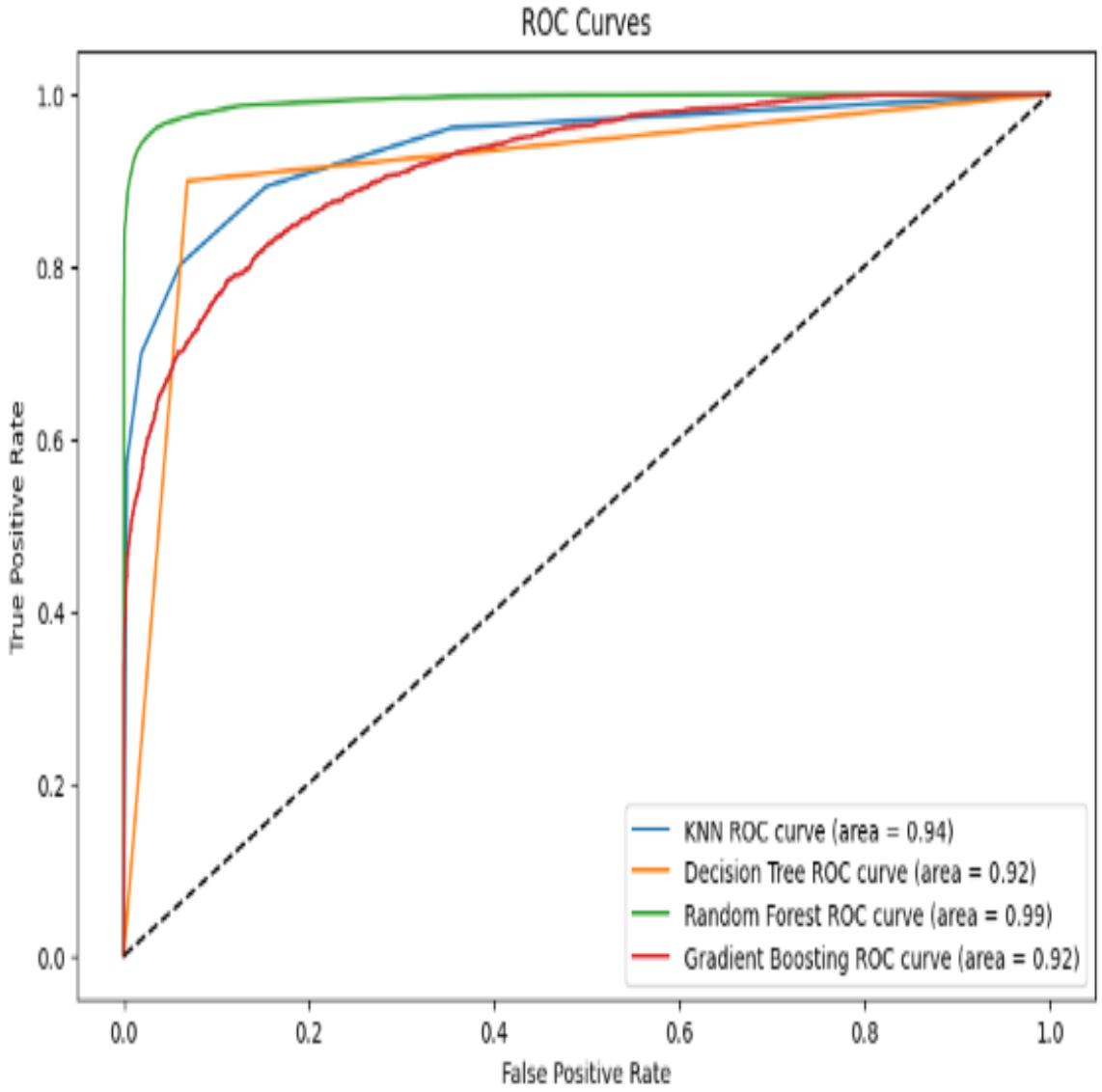
a) Accuracy score

```
# Models' names and their respective accuracy scores
models = ['KNN', 'Decision Tree', 'Random Forest','Gradient Boosting']
accuracy_scores = [acc_knn, acc_dtc, acc_rd_clf,acc_gb]

plt.figure(figsize=(10, 6))
plt.bar(models, accuracy_scores, color=['blue', 'green', 'red'])
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
plt.show()
```



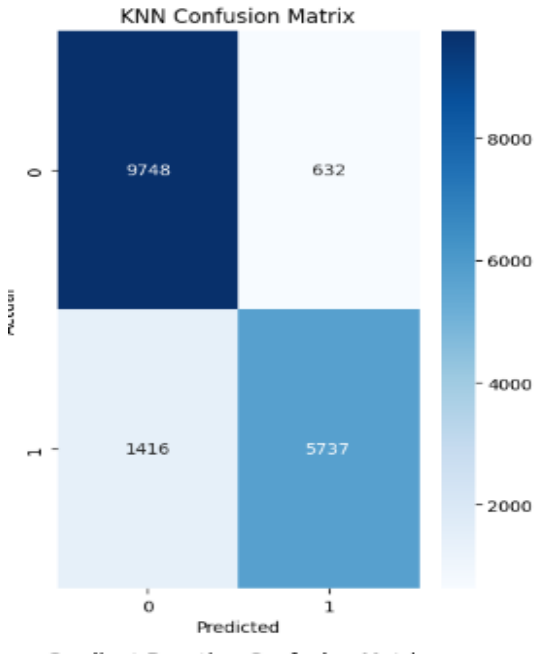
b) ROC Curve

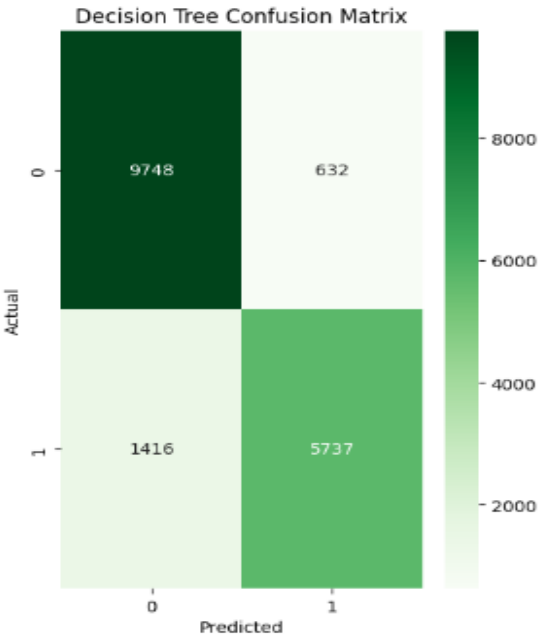
```
from sklearn.metrics import roc_curve, auc

# Getting the predicted probabilities for each model
y_prob_knn = knn.predict_proba(X_test)[:, 1]
y_prob_dtc = dtc.predict_proba(X_test)[:, 1]
y_prob_rd_clf = rd_clf.predict_proba(X_test)[:, 1]
y_prob_gb_clf = gb_clf.predict_proba(X_test)[:, 1]

# Calculating ROC curve and ROC AUC for each model
fpr_knn, tpr_knn, _ = roc_curve(y_test, y_prob_knn)
roc_auc_knn = auc(fpr_knn, tpr_knn)

fpr_dtc, tpr_dtc, _ = roc_curve(y_test, y_prob_dtc)
roc_auc_dtc = auc(fpr_dtc, tpr_dtc)

fpr_rd_clf, tpr_rd_clf, _ = roc_curve(y_test, y_prob_rd_clf)
roc_auc_rd_clf = auc(fpr_rd_clf, tpr_rd_clf)

fpr_gb_clf, tpr_gb_clf, _ = roc_curve(y_test, y_prob_gb_clf)
roc_auc_gb_clf = auc(fpr_gb_clf, tpr_gb_clf)



# Plotting the ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr_knn, tpr_knn, label=f'KNN ROC curve (area = {roc_auc_knn:.2f})')
plt.plot(fpr_dtc, tpr_dtc, label=f'Decision Tree ROC curve (area = {roc_auc_dtc:.2f})')
plt.plot(fpr_rd_clf, tpr_rd_clf, label=f'Random Forest ROC curve (area = {roc_auc_rd_clf:.2f})')
plt.plot(fpr_gb_clf, tpr_gb_clf, label=f'Gradient Boosting ROC curve (area = {roc_auc_gb_clf:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend(loc='lower right')
plt.show()
```

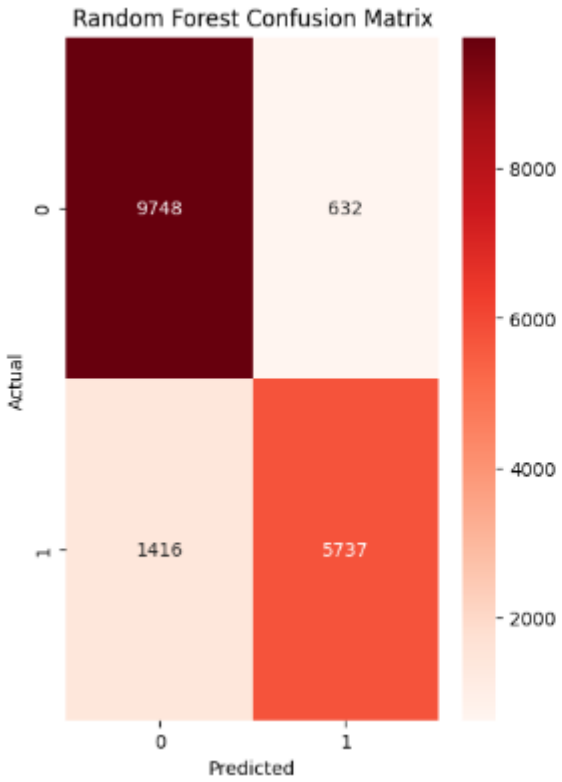

ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
KNN ROC curve (area = 0.94)
Decision Tree ROC curve (area = 0.92)
Random Forest ROC curve (area = 0.99)
Gradient Boosting ROC curve (area = 0.92)

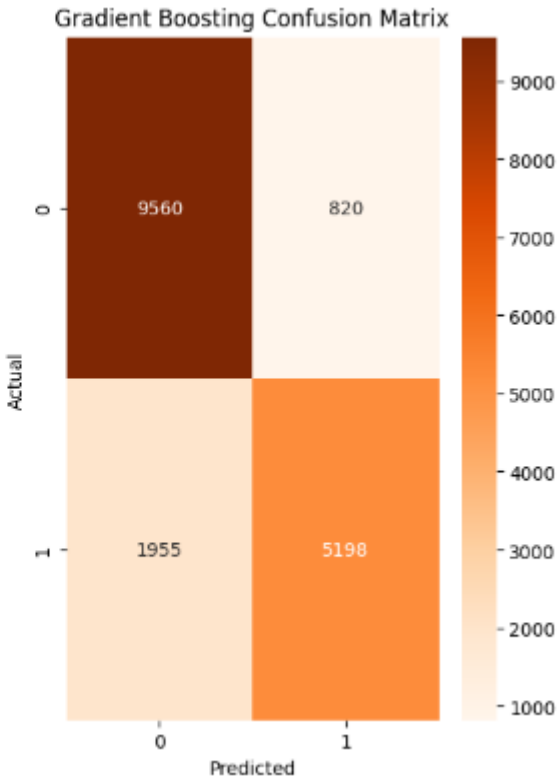
c) Confusion matrix

```
# Visualize confusion matrices
plt.figure(figsize=(13 ,12))

# Logistic Regression Confusion Matrix
plt.subplot(2, 3, 1)
sns.heatmap(conf, annot=True, cmap='Purples', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Logistic Regression Confusion Matrix')

# KNN Confusion Matrix
plt.subplot(2, 3, 2)
sns.heatmap(conf_knn, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('KNN Confusion Matrix')

# Decision Tree Confusion Matrix
plt.subplot(2, 3, 3)
sns.heatmap(conf, annot=True, cmap='Greens', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')

# Random Forest Confusion Matrix
plt.subplot(2, 3, 4)
sns.heatmap(conf, annot=True, cmap='Reds', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')

# Gradient Boosting Confusion Matrix
plt.subplot(2, 3, 5)
sns.heatmap(conf_gb, annot=True, cmap='Oranges', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Gradient Boosting Confusion Matrix')

plt.tight_layout()
plt.show()
```


KNN Confusion Matrix
9748
632
1416
5737
0
1
Predicted
8000
6000
4000
2000


Decision Tree Confusion Matrix
9748
632
1416
5737
Actual
0
1
Predicted
8000
6000
4000
2000


Random Forest Confusion Matrix
9748
632
1416
5737
Actual
0
1
Predicted
8000
6000
4000
2000


Gradient Boosting Confusion Matrix
9560
820
1955
5198
Actual
0
1
Predicted
9000
8000
7000
6000
5000
4000
3000
2000
1000

VI.SUGGESTION

1. Business problems

a) Dynamic Pricing and Discounts

+ **Seasonal Adjustments**: Noticing that cancellations are highest in January due to price increases, the hotels should reconsider their pricing strategy for that month. Given that ADR is lowest in January, it might be beneficial to offer special discounts or promotions during this period to encourage bookings

+ **Peak Pricing:** August has the highest ADR, suggesting high demand. The hotel could employ a dynamic pricing strategy where prices slightly increase during peak times but offer non-refundable rates at discounted prices to reduce cancellations**.**

b) Partnerships with Travel Agencies

+With **47%** bookings coming from online travel agencies and **20%** from offline agencies, strengthening partnerships with these agencies can be beneficial. Consider offering special rates for agency customers, and in return, ask agencies to push for non-refundable bookings.

+Collaborate with these agencies for joint **marketing campaigns** or **bundle offers** (like combining hotel stay with sightseeing tours) to attract more customers

c) Flexible Booking Policies with Incentives:

+ While non-refundable rates can deter cancellations, they might not be appealing to all. Therefore, provide a flexible booking option but with incentives for those who don't cancel. For instance, if a customer doesn't cancel, they might get a free meal, a discount for a room upgrade, or a discount on their next stay.

+ Consider implementing a policy where customers can reschedule (instead of canceling) their stay to a future date without penalty. This ensures that the hotel retains the revenue even if the customer can't make their original dates

2. Model machine learning

+ **Accuracy:**

KNN: 84.85%

Decision Tree: 91.37%

Random Forest: 95.99%

Gradient Boosting: 86.54%

⇨ **The highest accuracy is obtained by the Random Forest model.**

+ Precision:

KNN: 78%

Decision Tree: 84%

Random Forest: 99%

Gradient Boosting: 89% for

⇨ **The highest precision for class 1 is obtained by the Random Forest model.**

+ Recall:

KNN: 62%

Decision Tree: 85%

Random Forest: 87%

Gradient Boosting: 58%

=> **The highest recall for class 1 is obtained by the Random Forest model.**

+ **F1-Score**:

KNN: 69%

Decision Tree: 84%

Random Forest: 92%

Gradient Boosting: 70%

⇨ **The highest F1-Score for class 1 is obtained by the Random Forest** model.

Conclusion: Based on the results, the Random Forest model seems to be the best performer for predicting cancellations. It has the highest accuracy, precision, recall, and F1 score among all the models.

The End