Load the Dataset

```
import pandas as pd
df = pd.read_excel('/content/drive/MyDrive/ML Projects/Hotel Reservations.xlsx')
df
```

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_ni
0	INN00001	2	0	1	
1	INN00002	2	0	2	
2	INN00003	1	0	2	
3	INN00004	2	0	0	
4	INN00005	2	0	1	
...	...	...	...	...	
36270	INN36271	3	0	2	
36271	INN36272	2	0	1	
36272	INN36273	2	0	2	
36273	INN36274	2	0	0	
36274	INN36275	2	0	1	

36275 rows × 19 columns

Data Preprocessing

Preprocessing

```
df.shape
```

```
(36275, 19)
```

```
df.columns
```

```
Index(['Booking_ID', 'no_of_adults', 'no_of_children', 'no_of_weekend_nights',
       'no_of_week_nights', 'type_of_meal_plan', 'required_car_parking_space',
       'room_type_reserved', 'lead_time', 'arrival_year', 'arrival_month',
       'arrival_date', 'market_segment_type', 'repeated_guest',
       'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled',
       'avg_price_per_room', 'no_of_special_requests', 'booking_status'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Booking_ID                            36275 non-null  object 
 1   no_of_adults                          36275 non-null  int64  
 2   no_of_children                        36275 non-null  int64  
 3   no_of_weekend_nights                  36275 non-null  int64  
 4   no_of_week_nights                     36275 non-null  int64  
 5   type_of_meal_plan                     36275 non-null  object 
 6   required_car_parking_space            36275 non-null  int64  
 7   room_type_reserved                    36275 non-null  object 
 8   lead_time                             36275 non-null  int64  
 9   arrival_year                          36275 non-null  int64  
 10  arrival_month                         36275 non-null  int64  
 11  arrival_date                          36275 non-null  int64  
 12  market_segment_type                   36275 non-null  object 
 13  repeated_guest                        36275 non-null  int64  
 14  no_of_previous_cancellations          36275 non-null  int64  
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64  
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64  
 18  booking_status                        36275 non-null  object 
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

```
df.describe()
```

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	requir
count	36275.000000	36275.000000	36275.000000	36275.000000	
mean	1.844962	0.105279	0.810724	2.204300	
std	0.518715	0.402648	0.870644	1.410905	
min	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	
50%	2.000000	0.000000	1.000000	2.000000	
75%	2.000000	0.000000	2.000000	3.000000	
max	4.000000	10.000000	7.000000	17.000000	

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

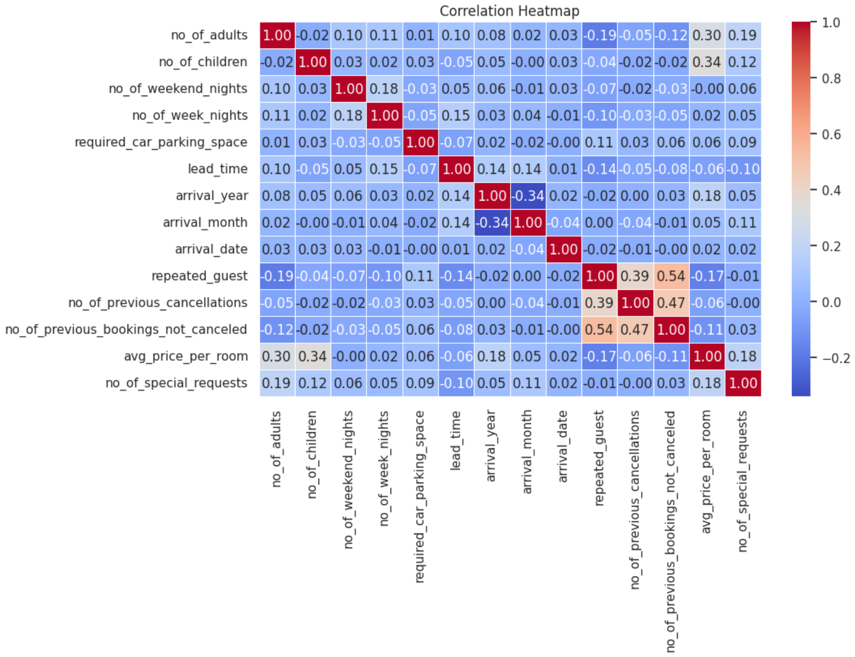
```
Booking_ID                              0
no_of_adults                            0
no_of_children                          0
no_of_weekend_nights                    0
no_of_week_nights                       0
type_of_meal_plan                       0
required_car_parking_space              0
room_type_reserved                      0
lead_time                               0
arrival_year                            0
arrival_month                           0
arrival_date                            0
market_segment_type                     0
repeated_guest                          0
no_of_previous_cancellations            0
no_of_previous_bookings_not_canceled    0
avg_price_per_room                      0
no_of_special_requests                  0
booking_status                          0
dtype: int64
```

```
df.nunique()
```

```
Booking_ID                              36275
no_of_adults                                5
no_of_children                              6
no_of_weekend_nights                        8
no_of_week_nights                          18
type_of_meal_plan                           4
required_car_parking_space                  2
room_type_reserved                          7
lead_time                                 352
arrival_year                                2
arrival_month                              12
arrival_date                               31
market_segment_type                         5
repeated_guest                              2
no_of_previous_cancellations                9
no_of_previous_bookings_not_canceled       59
avg_price_per_room                       3930
no_of_special_requests                      6
booking_status                              2
dtype: int64
```

Heatmap Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style = "white")
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only = True) , annot=True , cmap='coolwarm' , fmt='.2f' , linewidth=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Data Encoding

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

df['Booking_ID'] = label_encoder.fit_transform(df['Booking_ID'])
df['type_of_meal_plan'] = label_encoder.fit_transform(df['type_of_meal_plan'])
df['market_segment_type'] = label_encoder.fit_transform(df['market_segment_type'])
df['room_type_reserved'] = label_encoder.fit_transform(df['room_type_reserved'])
df['booking_status'] = label_encoder.fit_transform(df['booking_status'])
```

Data Scaling

```
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
df = standard_scaler.fit_transform(df)
df
```

```
     array([[-1.73200306,  0.29889263, -0.26147045, ..., -1.09503276,
             -0.78813999,  0.69806151],
            [-1.73190756,  0.29889263, -0.26147045, ...,  0.09280591,
              0.48376045,  0.69806151],
            [-1.73181207, -1.62897546, -0.26147045, ..., -1.2375278 ,
             -0.78813999, -1.43253851],
            ...,
            [ 1.73181207,  0.29889263, -0.26147045, ..., -0.14345087,
              1.75566089,  0.69806151],
            [ 1.73190756,  0.29889263, -0.26147045, ..., -0.25431201,
             -0.78813999, -1.43253851],
            [ 1.73200306,  0.29889263, -0.26147045, ...,  1.65996637,
             -0.78813999,  0.69806151]])
```

Data Seperation of x(features) and y(target)

```
y = df['booking_status']
x = df.drop(columns=['booking_status'])
```

Data Spliting into x_train , x_test , y_train , y_test

```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size=0.2 , random_state = 42)
```

```
x_train.shape
```

```
(29020, 18)
```

```
x_test.shape
```

```
(7255, 18)
```

Converting the Categorical features into numerical for train and test datasets

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

x_train['Booking_ID'] = label_encoder.fit_transform(x_train['Booking_ID'])
x_test['Booking_ID'] = label_encoder.fit_transform(x_test['Booking_ID'])

x_train['type_of_meal_plan'] = label_encoder.fit_transform(x_train['type_of_meal_plan'])
x_test['type_of_meal_plan'] = label_encoder.fit_transform(x_test['type_of_meal_plan'])

x_train['market_segment_type'] = label_encoder.fit_transform(x_train['market_segment_type'])
x_test['market_segment_type'] = label_encoder.fit_transform(x_test['market_segment_type'])

x_train['room_type_reserved'] = label_encoder.fit_transform(x_train['room_type_reserved'])
x_test['room_type_reserved'] = label_encoder.fit_transform(x_test['room_type_reserved'])
```

Model Phase

Building a Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression()
```

Train the Model

```
lgr.fit(x_train , y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Conve
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

Apply the Model to make predictions

```
y_lgr_test_pred = lgr.predict(x_test)
```

Evaluating the Model

```
from sklearn.metrics import confusion_matrix
confusion_matrix_results = confusion_matrix(y_test , y_lgr_test_pred)
confusion_matrix_results
```

```
array([[1172, 1244],
       [ 448, 4391]])
```

```
def accuracy_percentage_from_confusion_matrix(confusion_matrix):
    #Calculate the accuracy from the confusion matrix
    true_positive, false_positive = confusion_matrix[0]
    false_negative, true_negative = confusion_matrix[1]
    accuracy = (true_positive + true_negative) / (true_positive + false_positive + false_negative + true_negative)

    #Convert accuracy to a percentage
    accuracy_percentage_lgr = accuracy * 100
    return accuracy_percentage_lgr

confusion_matrix = ([[1172, 1244],
                    [448, 4391]])

#Calculate accuracy as a percentage
accuracy_percentage_lgr = accuracy_percentage_from_confusion_matrix(confusion_matrix)
print(f"Accuracy: {accuracy_percentage_lgr:.2f}%")
```

```
Accuracy: 76.68%
```

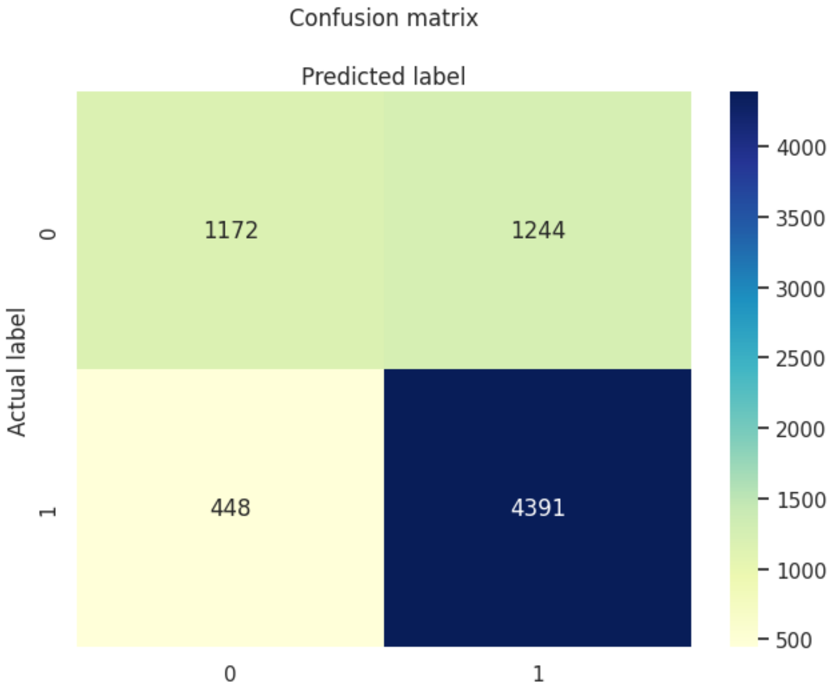
Visualizing Confusion Matrix

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

confusion_matrix = np.array([[1172, 1244],
                            [448, 4391]])

#create heatmap
sns.heatmap(pd.DataFrame(confusion_matrix_results), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



Using Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

```
rfc.fit(x_train , y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
µy_rfc_test_pred = rfc.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
model_result = accuracy_score(y_test , y_rfc_test_pred)
accuracy_percentage_rfc = pd.DataFrame(["Random Forest" , model_result]).transpose()
accuracy_percentage_rfc.columns = ["Method" , "accuracy_percentage"]
accuracy_percentage_rfc
```

	Method	accuracy_percentage
0	Random Forest	0.894693

Conclusion

-Logistic Regression model has 76.68% accuracy at predicting the hotel's reservation cancellation

-Random Forest model has 89.46% accuracy at predicting the hotel's reservation cancellation

->the Random Forest has a higher precision and better prediciton performance