The Hotel Reservation

About Contest:

Context

The online hotel reservation channels have dramatically changed booking possibilities and customers' behavior. A significant number of hotel reservations are called-off due to cancellations or no-shows. The typical reasons for cancellations include change of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with.

Can you predict if the customer is going to honor the reservation or cancel it ?

About Dataset:

About this file

The file contains the different attributes of customers' reservation details. The detailed data dictionary is given below.

Data Dictionary

- **Booking_ID**: unique identifier of each booking
- **no_of_adults**: Number of adults
- **no_of_children**: Number of Children
- **no_of_weekend_nights**: Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- **no_of_week_nights**: Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- **type_of_meal_plan**: Type of meal plan booked by the customer:
- **required_car_parking_space**: Does the customer require a car parking space? (0 - No, 1- Yes)
- **room_type_reserved**: Type of room reserved by the customer. The values are ciphered (encoded) by INN Hotels.
- **lead_time**: Number of days between the date of booking and the arrival date
- **arrival_year**: Year of arrival date
- **arrival_month**: Month of arrival date
- **arrival_date**: Date of the month
- **market_segment_type**: Market segment designation.
- **repeated_guest**: Is the customer a repeated guest? (0 - No, 1- Yes)

- **no_of_previous_cancellations**: Number of previous bookings that were canceled by the customer prior to the current booking
- **no_of_previous_bookings_not_canceled**: Number of previous bookings not canceled by the customer prior to the current booking
- **avg_price_per_room**: Average price per day of the reservation; prices of the rooms are dynamic. (in euros)
- **no_of_special_requests**: Total number of special requests made by the customer (e.g. high floor, view from the room, etc)
- **booking_status**: Flag indicating if the booking was canceled or not.

```
columns
```

```
ex(['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')
```

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
INN00001	2	0	1	2	Meal Plan 1	0	Room_Type 1
INN00002	2	0	2	3	Not Selected	0	Room_Type 1
INN00003	1	0	2	1	Meal Plan 1	0	Room_Type 1
INN00004	2	0	0	2	Meal Plan 1	0	Room_Type 1
INN00005	2	0	1	1	Not Selected	0	Room_Type 1

lead_time	arrival_year	arrival_month	arrival_date	market_segment_type	repeated_guest	no_of_previous_cancellations	no_of_previous_bookings_not_canceled
224	2017	10	2	Offline	0	0	0
5	2018	11	6	Online	0	0	0
1	2018	2	28	Online	0	0	0
211	2018	5	20	Online	0	0	0
48	2018	4	11	Online	0	0	0

avg_price_per_room	no_of_special_requests	booking_status
65.00	0	Not_Canceled
106.68	1	Not_Canceled
60.00	0	Canceled
100.00	0	Canceled
94.50	0	Canceled

This is the dataset of hotel reservation system, There are 36275 numbers of rows and 19
Numbers of Features in which booking_status(Flag indicating if the booking was cancelled or not) is your target column you need to convert booking_status into dummy variable by one Hot Encoding in which 1 represent Not_Cancled and 0 represents Cancelled.

Submission Guidelines:

For completing this challenge, the following **two files** needs to be submitted:

- A Jupyter notebook file (i.e., .ipynb file) used for code implementation. Name the file as ***geekschallange_Firstname_Lastname.ipynb***. For example, if the participant's name is John Smith, then the submission file will be named as ***geekschallange_John_Smith.ipynb***

- The trained machine learning model used for the data challenges as a pickle file [follow the appendix for better understanding]. Name the file as ***geekschallange_Firstname_Lastname.pickle***. For example, if the participant's name is John Smith, then the submission file will be named as ***geekschallange_John_Smith.pickle***

  Make sure to follow the following guidelines for naming the datafile. Do not zip these files into one zip archive, submit two independent files.

We will only accept Jupyter Notebook submissions. Please make sure that the code is properly commented, and the jupyter notebook should include the reporting about the following:

- A brief description of the data set and the problem.

- Details of the data exploration and pre-processing steps.

- Analysis of the exploration plots and insights.
- An explanation of the model selection process and the chosen model.
- Final model evaluation metrics and a discussion of model performance.

Evaluation Criteria:

Submissions will be evaluated based on the following criteria:

- Data Pre-processing techniques
- Data exploration techniques and analysis
- Model selection and performance
- Final model evaluation and discussion
- Coding architecture and commenting

Resources:

- A brief description of the data set and the problem:

https://www.geeksforgeeks.org/how-to-explain-a-machine-learning-project-to-an-interviewer/

- Details of the data exploration and pre-processing steps:

https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

- Analysis of the exploration plots and insights:

https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/

- An explanation of the model selection process and the chosen model:

https://www.geeksforgeeks.org/model-selection-in-machine-learning/

- Final model evaluation metrics and a discussion of model performance:

https://www.geeksforgeeks.org/evaluation-metrics-in-machine-learning/

Appendix:

What is a pickle file?

A pickle file is a serialized object that can be used to store the state of a Python object, such as a machine learning model. It is used in machine learning to save a trained model, along with any pre-processing steps that have been applied to the data, so that it can be reloaded later and used to make predictions on new data.

When you train a machine learning model, the resulting model is just a set of weights and biases that have been learned from the training data. To use the model later, you need to save these weights and biases in a file so that they can be loaded into memory and used to make predictions on new data.

Pickle files provide a way to save Python objects to disk, so that they can be reloaded later. This is useful in machine learning because it allows you to save a trained model, along with any pre-processing steps that have been applied to the data, in a single file. This makes it easy to share and distribute trained models, and to use them in production environments.

How to save the machine learning model in a pickle file

Let's suppose you have trained a logistic regression machine learning model on the dataset as show in the figure 2 below:

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()

# Train a logistic regression model on the dataset
X, y = iris.data, iris.target
linear_model = LogisticRegression()
linear_model.fit(X, y)
```

```
import pickle
# Save the trained model to a file
with open('model.pickle', 'wb') as f:
    pickle.dump(linear_model, f)
```

To load the saved model back into memory later, you can use the pickle.load() function, like this:

```
# Load the saved model from a file
with open('model.pickle', 'rb') as f:
    saved_model = pickle.load(f)

# Use the loaded model to make predictions
X_new = [[5.0, 3.6, 1.3, 0.25]]
y_pred = saved_model.predict(X_new)
print(y_pred)
```

Or

```
# Load the saved model from a file
pickle_in = open('model.pkl', 'rb')
saved_model = pickle.load(pickle_in)

# Use the loaded model to make predictions
X_new = [[5.0, 3.6, 1.3, 0.25]]
y_pred = saved_model.predict(X_new)
print(y_pred)
```