# Project: Putting it All Together: Building a Data Mining System, From Data Cleaning to Model Evaluation

Kanika Gaur

# Dataset used: Hotel booking demand (https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand)

**Code:**

import numpy as np

import pandas as pd

import os

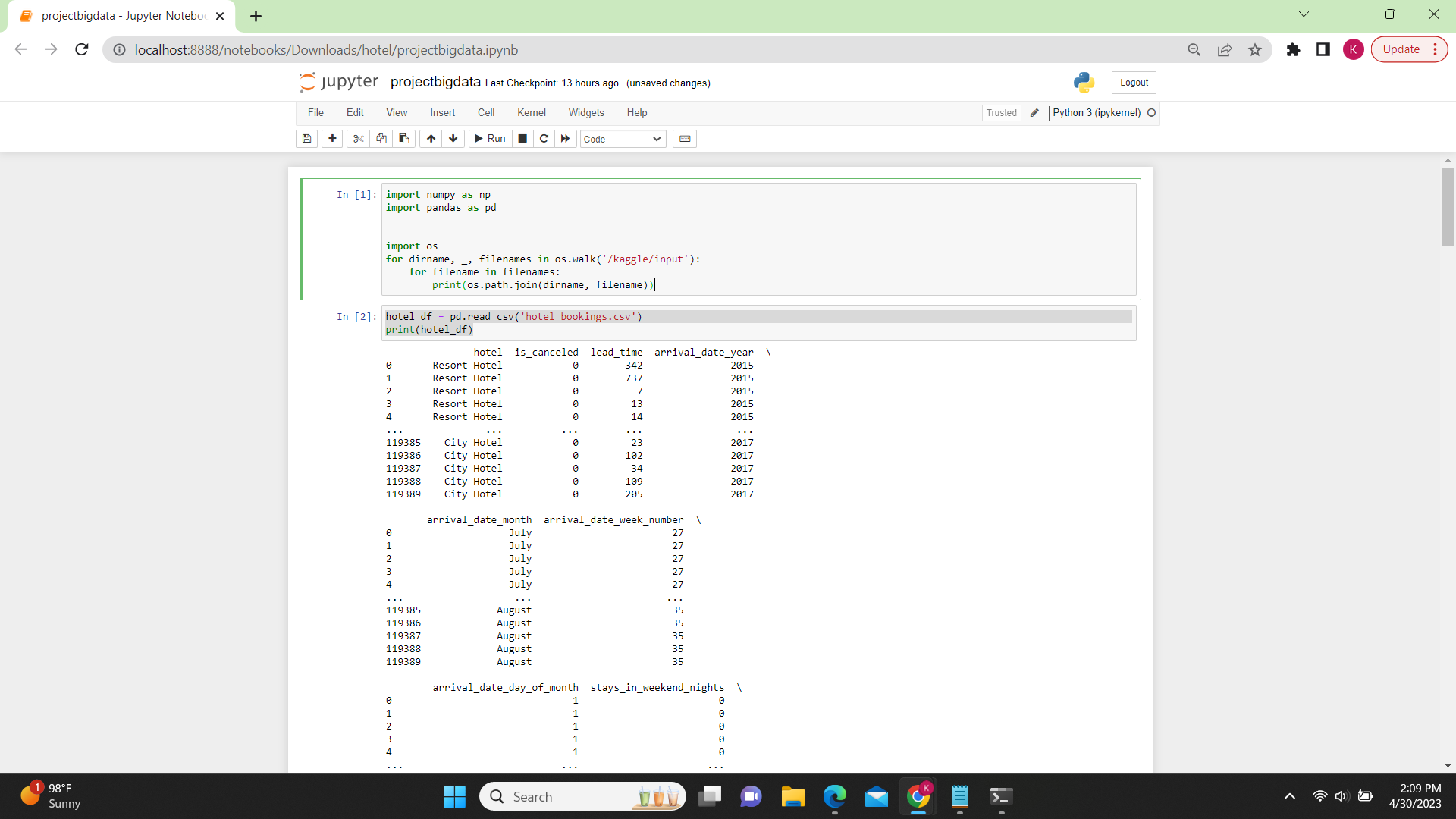
for dirname, \_, filenames in os.walk('/kaggle/input'):

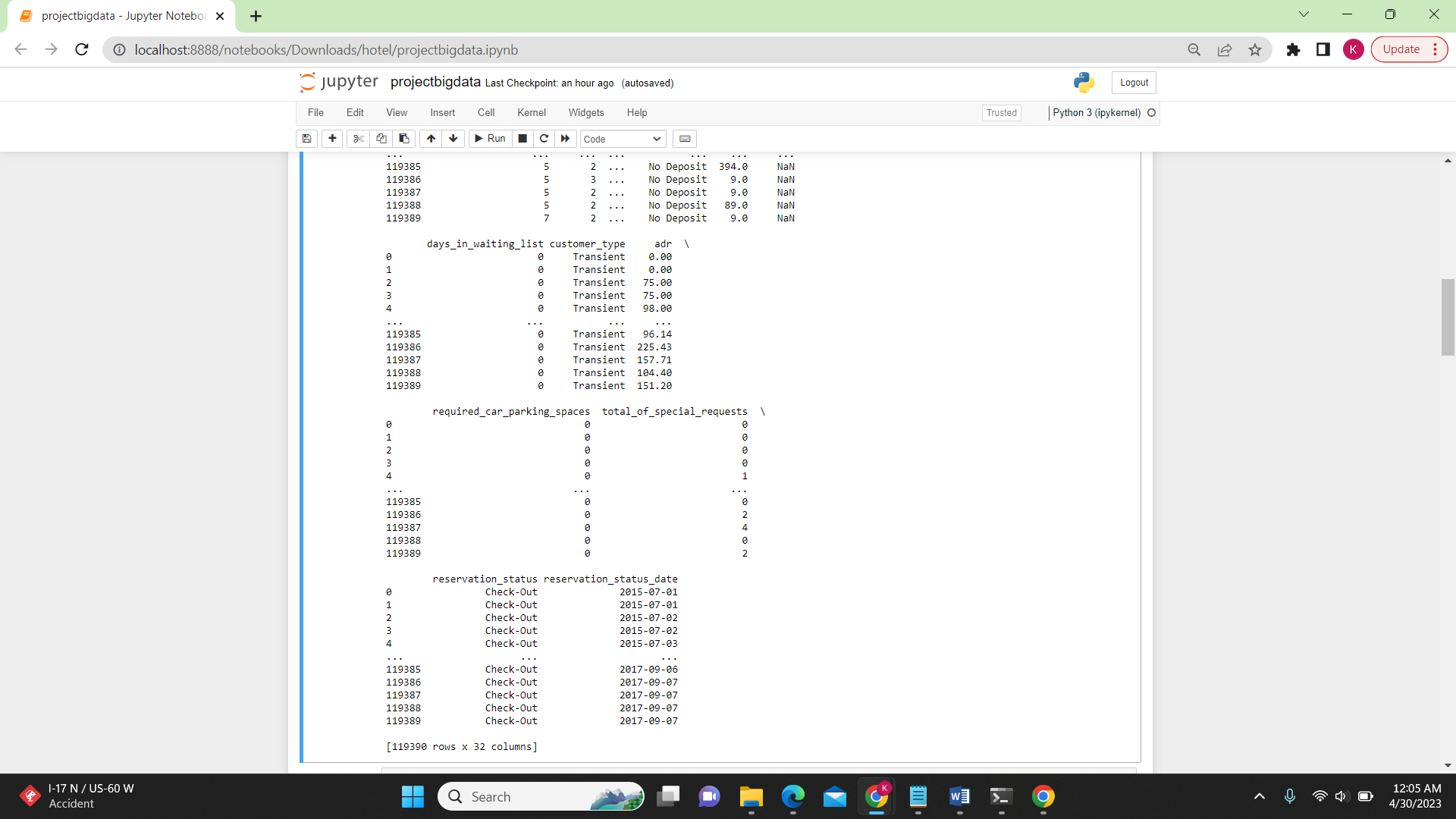
for filename in filenames:

print(os.path.join(dirname, filename))

hotel\_df = pd.read\_csv('hotel\_bookings.csv')

print(hotel\_df)

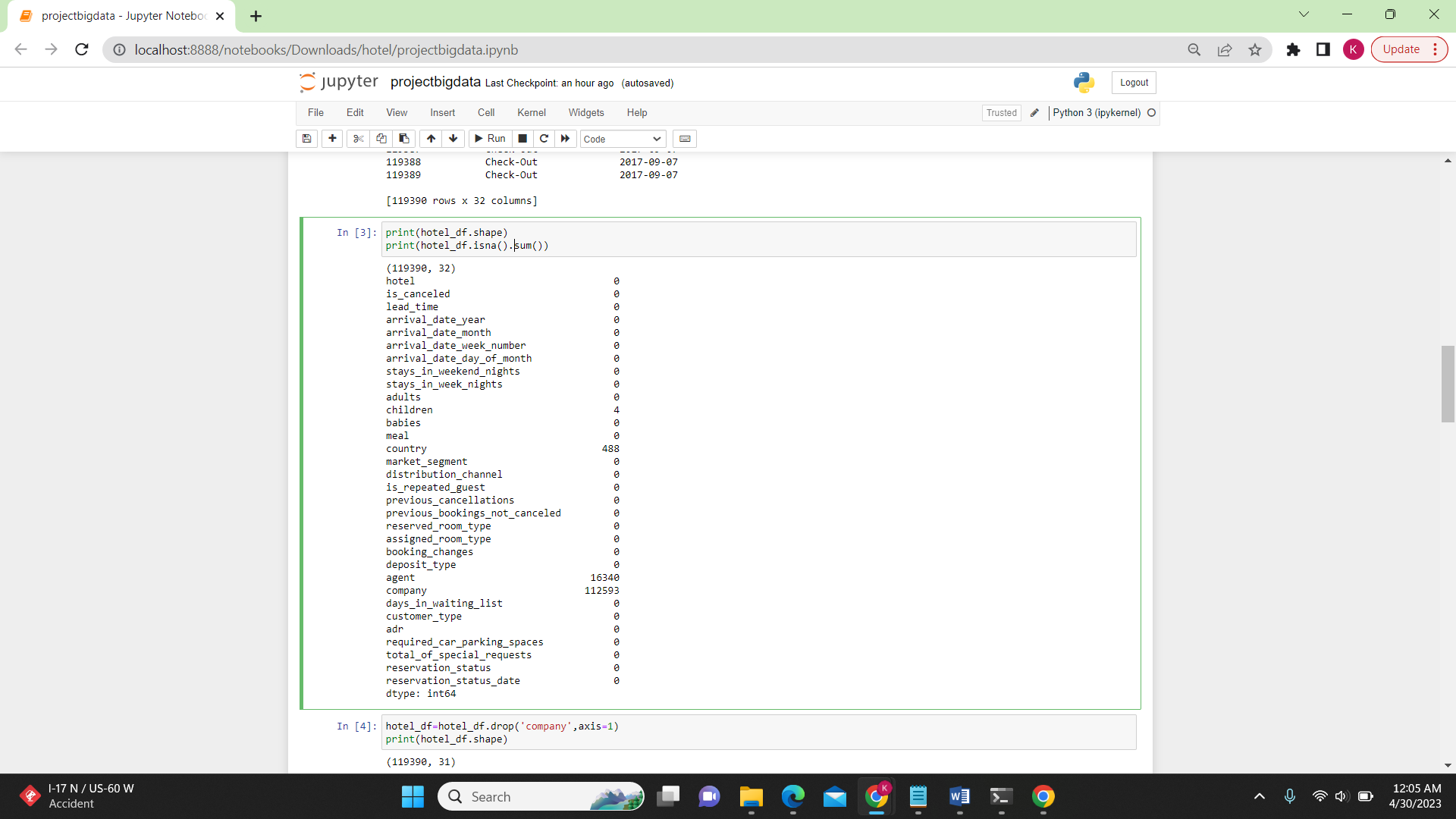




**Number of rows & columns in this dataset**: [119390 rows x 32 columns]

The hotel booking demand dataset available on Kaggle contains information on hotel booking demand for two hotels: a city hotel and a resort hotel. The dataset has a total of 119,390 rows and 32 columns.

Each row in the dataset represents a unique booking made by a customer. The columns contain various pieces of information about each booking, including the hotel type, booking date, length

of stay, number of adults/children, room type, meal plan, and whether or not the booking was ultimately cancelled. 

**Code:**

print(hotel\_df.shape)

print(hotel\_df.isna().sum())

**Each rows represents:**

**Nominal:**

hotel: the type of hotel (city hotel or resort hotel)

meal: the type of meal booked (Undefined/SC – no meal package; BB – Bed & Breakfast; HB – Half board (breakfast and one other meal); FB – Full board (breakfast, lunch and dinner))

market\_segment: the type of customer segment (e.g., Online Travel Agents, Offline TA/TO, Direct, etc.)

distribution\_channel: the distribution channel used to make the booking (e.g., Online Travel Agents, Offline TA/TO, Direct, etc.)

reserved\_room\_type: the type of room reserved (e.g., A, B, C, etc.)

assigned\_room\_type: the type of room assigned at check-in

deposit\_type: the type of deposit made (e.g., No Deposit, Non-refundable, Refundable)

**Ordinal:**

arrival\_date\_month: the month of arrival

arrival\_date\_year: the year of arrival

Interval

arrival\_date\_week\_number: the week number of the year of arrival

arrival\_date\_day\_of\_month: the day of the month of arrival

**Ratio:**

lead\_time: the number of days between the booking date and the arrival date

adults: number of adults in the booking

children: number of children in the booking

babies: number of babies in the booking

previous\_cancellations: number of previous bookings that were cancelled by the same customer

previous\_bookings\_not\_canceled: number of previous bookings not cancelled by the same customer

booking\_changes: number of changes made to the booking before arrival

days\_in\_waiting\_list: number of days the booking was on the waiting list before it was confirmed

adr: average daily rate (total booking revenue divided by number of nights stayed)

required\_car\_parking\_spaces: number of car parking spaces requested by the customer

total\_of\_special\_requests: number of special requests made by the customer (e.g., extra bed, late check-in, etc.)

stays\_in\_weekend\_nights: number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel

stays\_in\_week\_nights: number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel

1. **Dataset Cleaning:**

In order to clean the dataset, we need to remove any rows or columns with missing values. In this case, we will remove rows that have missing values in the 'country', 'children', and 'agent' columns. These missing values may affect the accuracy of the classification model if they are not dealt with properly.

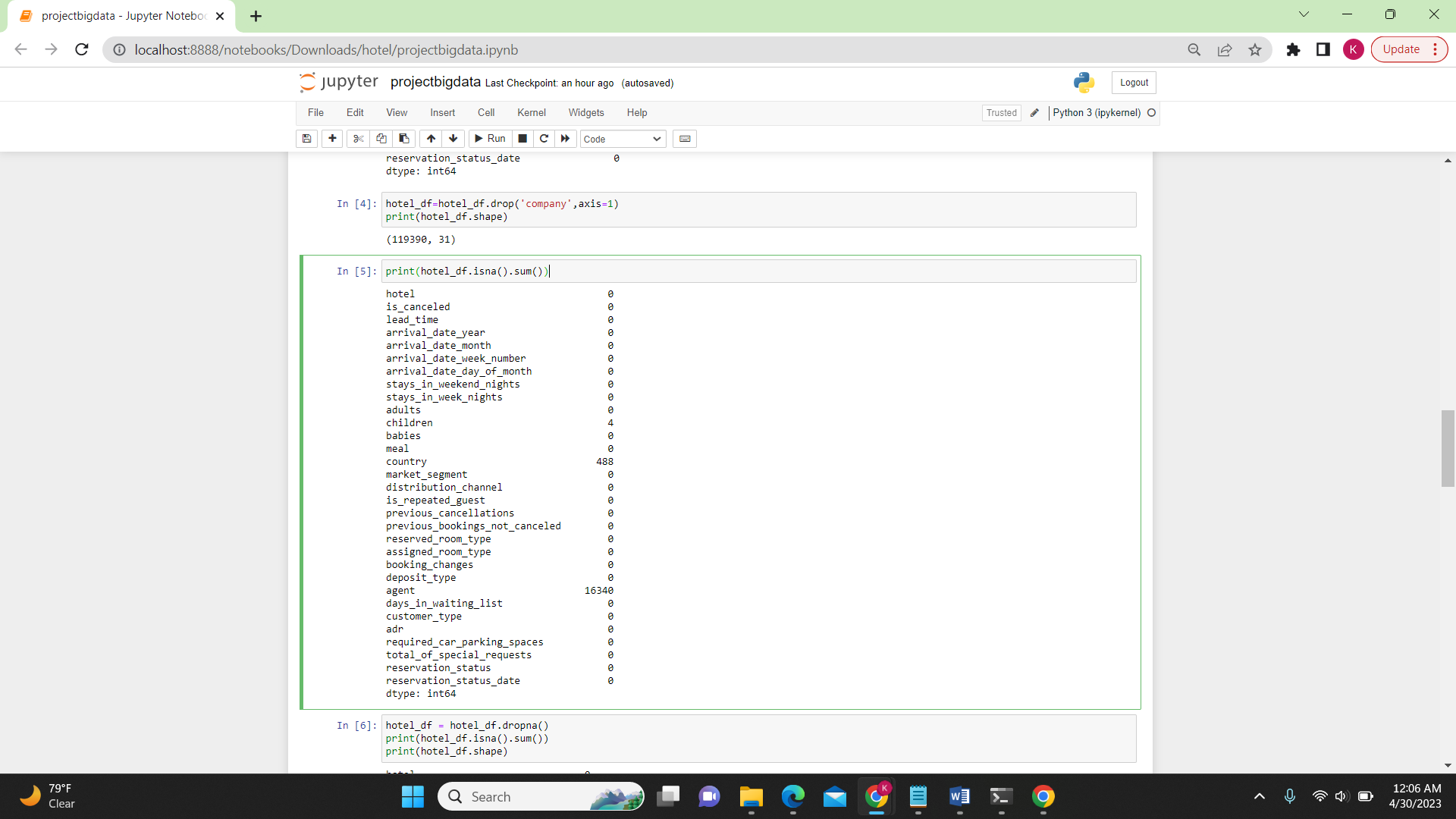
The “Company”, “Country”, and “Agent” columns represent entities external to the hotel and are less likely to have a direct impact on hotel bookings, compared to other features

**Code:**

hotel\_df=hotel\_df.drop('company',axis=1)

print(hotel\_df.shape)

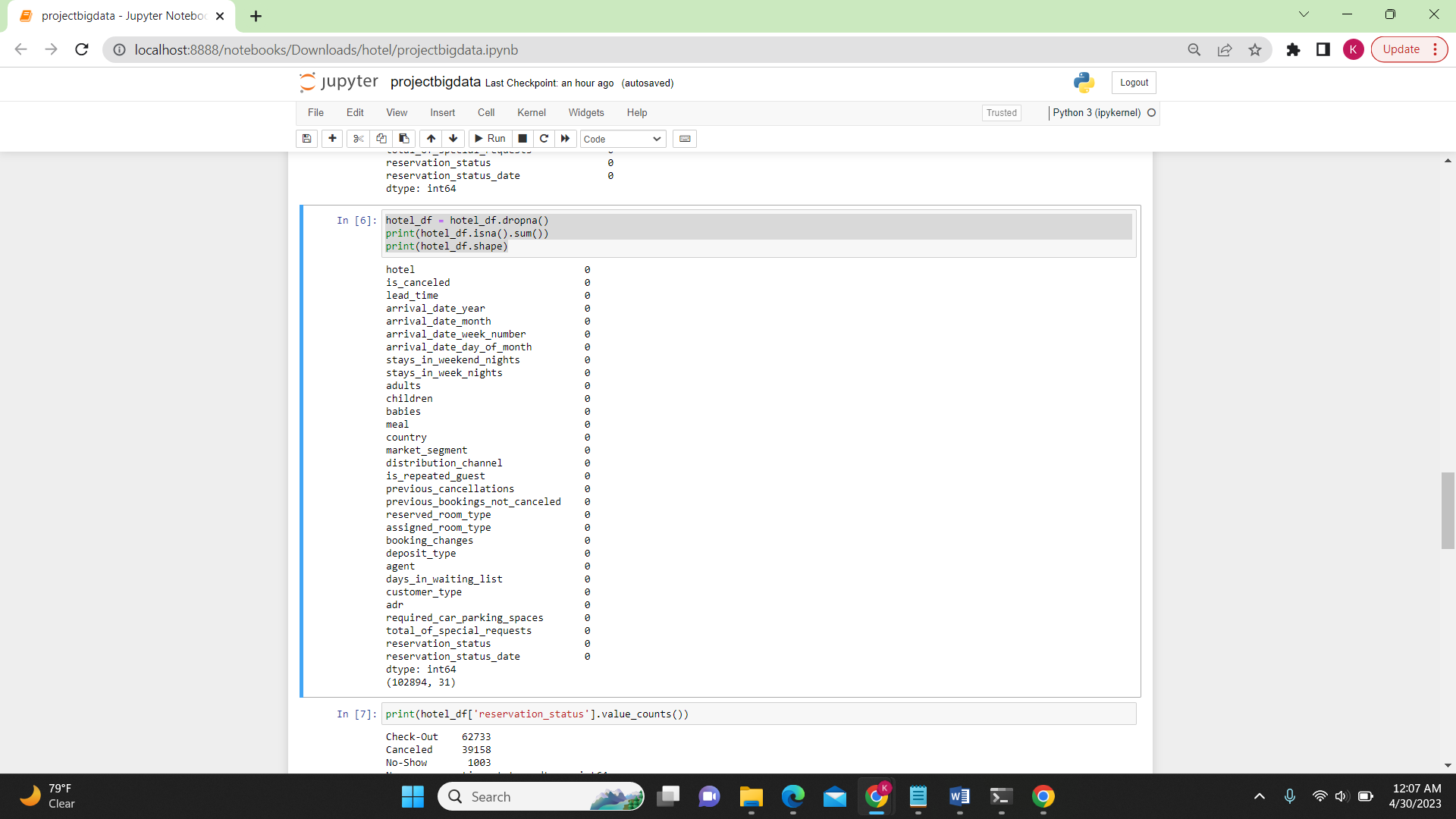
print(hotel\_df.isna().sum())



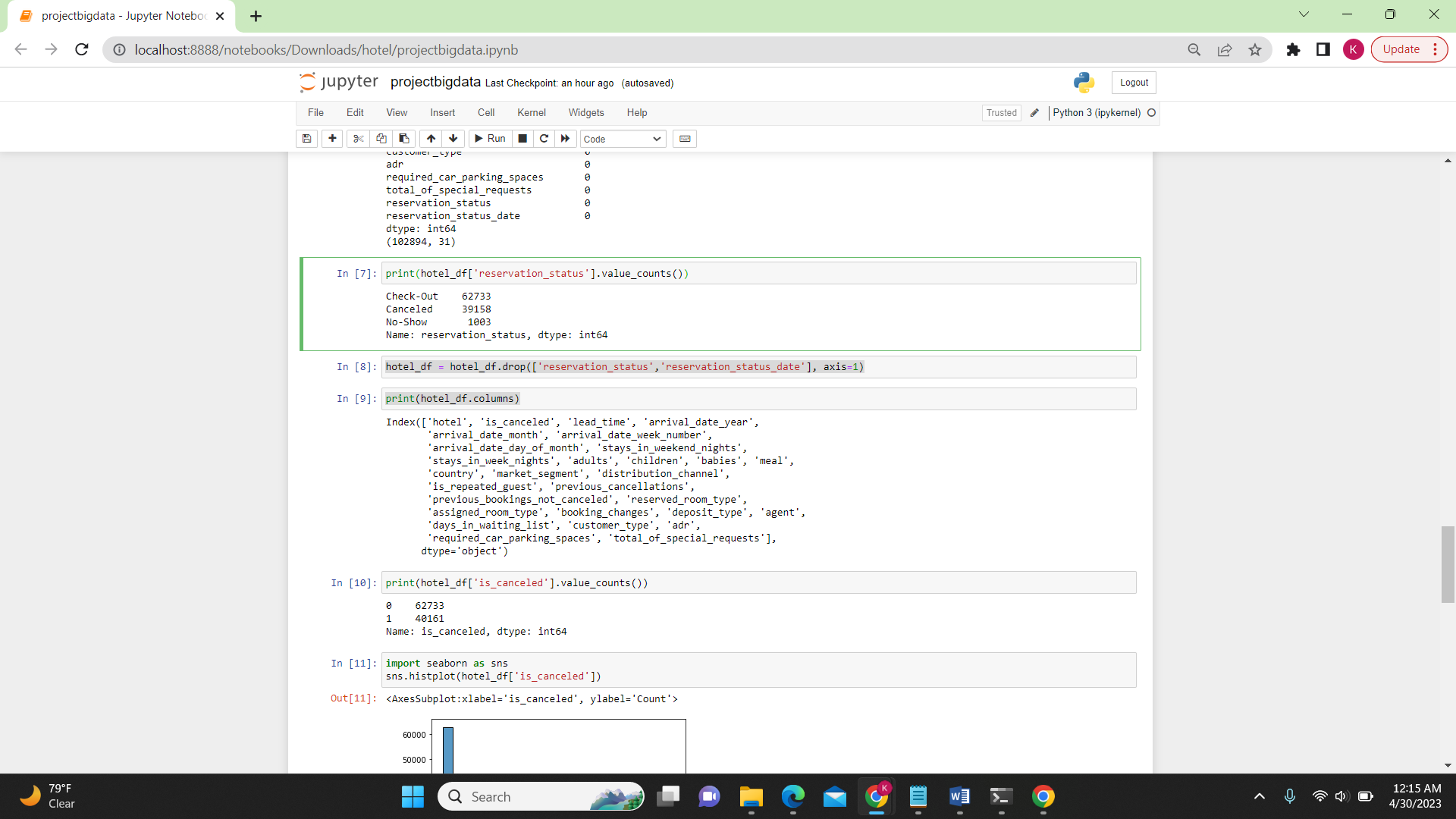
hotel\_df = hotel\_df.dropna()

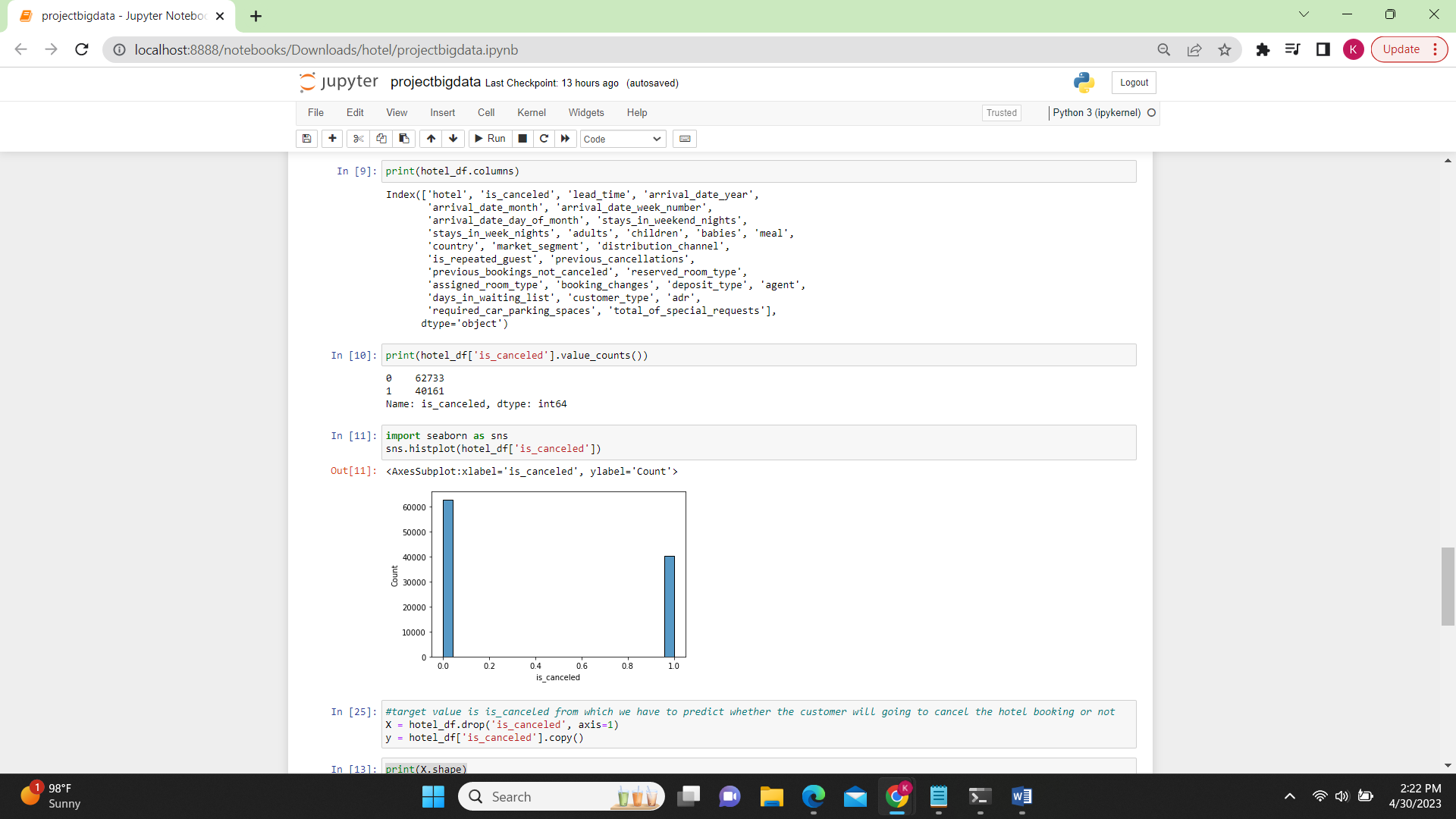
print(hotel\_df.isna().sum())

print(hotel\_df.shape)



hotel\_df = hotel\_df.drop(['reservation\_status','reservation\_status\_date'], axis=1)  
print(hotel\_df.columns)

print(hotel\_df['is\_canceled'].value\_counts())  




**Stratified K Fold:**

**Code:** Experiment with ‘Gini’

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import OrdinalEncoder

from sklearn.model\_selection import StratifiedKFold

X = hotel\_df.drop('is\_canceled', axis=1)

y = hotel\_df['is\_canceled'].copy()

print(X.shape)

print(y.shape)

decision\_tree = DecisionTreeClassifier(criterion = 'gini',max\_depth=290)

numeric\_cols = hotel\_df.select\_dtypes(include=np.number).columns.tolist()

object\_cols = list(set(hotel\_df.columns) - set(numeric\_cols))

enc = OrdinalEncoder()

enc.fit(hotel\_df[object\_cols])

hotel\_df[object\_cols] = enc.fit\_transform(hotel\_df[object\_cols])

skf = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=1)

dt\_acc = []

for train\_index, test\_index in skf.split(X, y):

train = hotel\_df.iloc[train\_index,:]

test = hotel\_df.iloc[test\_index,:]

X\_train = train.drop('is\_canceled', axis=1)

y\_train = train['is\_canceled'].copy()

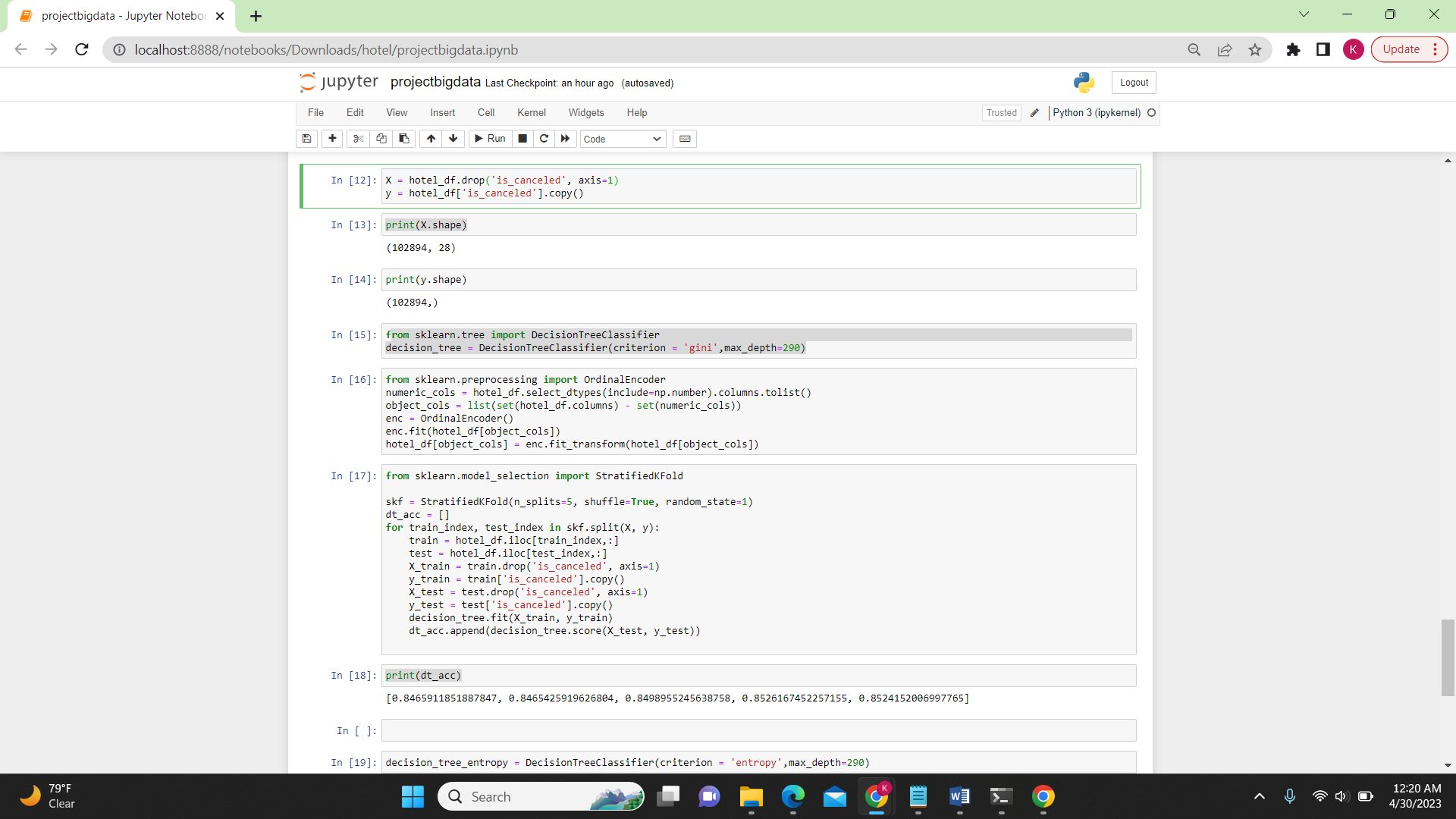
X\_test = test.drop('is\_canceled', axis=1)

y\_test = test['is\_canceled'].copy()

decision\_tree.fit(X\_train, y\_train)

dt\_acc.append(decision\_tree.score(X\_test, y\_test))

print(dt\_acc)



**Code:** Experiment with ‘Entropy’

decision\_tree\_entropy = DecisionTreeClassifier(criterion = 'entropy',max\_depth=290)

from sklearn.model\_selection import StratifiedKFold

skf = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=1)

dt\_acc\_entropy = []

for train\_index, test\_index in skf.split(X, y):

train = hotel\_df.iloc[train\_index,:]

test = hotel\_df.iloc[test\_index,:]

X\_train = train.drop('is\_canceled', axis=1)

y\_train = train['is\_canceled'].copy()

X\_test = test.drop('is\_canceled', axis=1)

y\_test = test['is\_canceled'].copy()

decision\_tree\_entropy.fit(X\_train, y\_train)

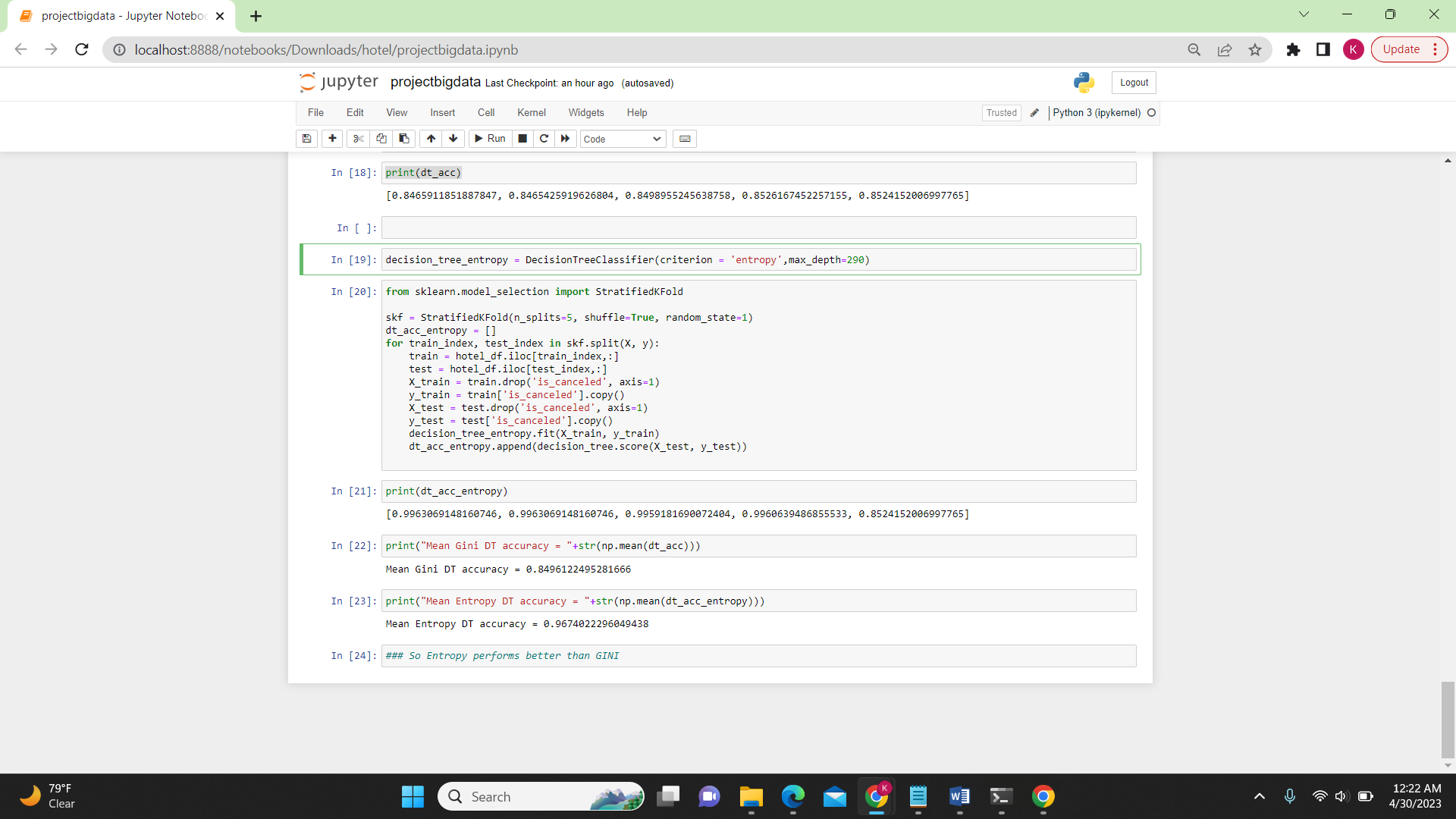
dt\_acc\_entropy.append(decision\_tree.score(X\_test, y\_test))

print(dt\_acc\_entropy)

print("Mean Gini DT accuracy = "+str(np.mean(dt\_acc)))

print("Mean Entropy DT accuracy = "+str(np.mean(dt\_acc\_entropy)))

### So Entropy performs better than GINI



**Parameter that gives the best result is:** Entropy i.e [0.9963069148160746, 0.9963069148160746, 0.9959181690072404, 0.9960639486855533, 0.8524152006997765] and the mean accuracy is 0.9674022296049438 which is higher than Gini parameter.

**References**

* <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html>
* <https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand>