Machine Learning Hotel Booking Cancellation Prediction

SENG474

Junyi Wei V00979016

Siheng Chen V00935153

Yifan Zhu V00977863

## Abstract

Unexpected hotel cancellations can be a significant challenge for the hotel industry. Therefore we believe by utilizing machine learning as a powerful tool for the prediction of cancellation, hotel resource allocation could be significantly optimized. We have made use of various helpful libraries to gain insights and clean/preprocess the data. Many insightful visuals are generated for the analysis and interpretations are also included.

# 

[Abstract 1](#_tyjcwt)

[Introduction 2](#_4d34og8)

[**Data Preparation 3**](#)

[Cleaning Data 3](#_17dp8vu)

[Data Visualization 5](#)

[Adding features 9](#)

[Feature encoding 10](#)

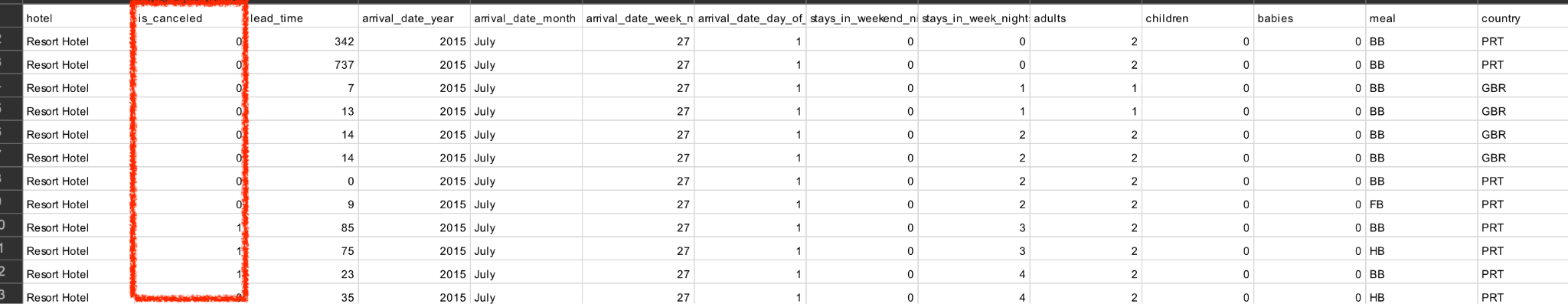
[Data Mining 11](#_26in1rg)

[Evaluation 11](#_lnxbz9)

[Conclusion 11](#_35nkun2)

## Introduction

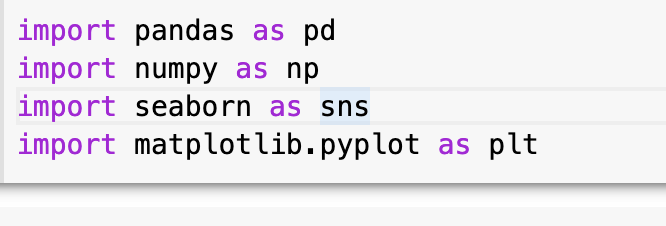
As briefly mentioned in the abstract, hotel cancellations can significantly impact resource allocation and profit for hotel booking platforms. This project aims to develop a machine learning model to predict unexpected cancellations of hotel bookings made by users. By addressing this practical real-world, business-related theme, we seek to provide a valuable tool that optimizes both resource allocation and profit in the hospitality industry. Our focus lies on the target feature "is\_canceled" within a comprehensive .csv dataset containing numerous other features. To build an effective model, a thorough understanding of the data and its insights is crucial. In this report, we present the key highlights and methods employed for the implementation of our model. Through this project, we aim to contribute to the optimization of hotel operations and enhance the efficiency of hotel booking platforms.



*Figure 1 target feature*

## Data Preparation

Since the entire project is extended from data, then it is critical to make sure the data is fit and ready before we begin the data analysis phase. In this case, the dataset is the basis of everything since without the data we have nothing to mine and analyze.

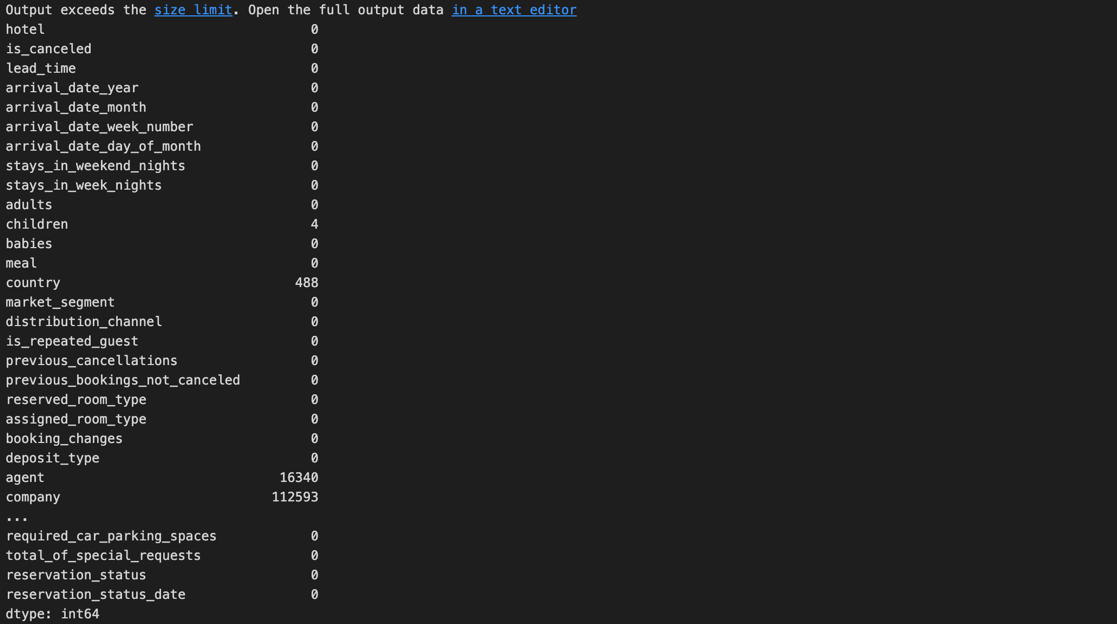


*Figure 2 Libraries used for the phase*

We have imported relevant python modules as shown in Fig2 to help process the data. Pandas are mainly used for data manipulation, numpy for mathematical operations while seaborn and matplotlib.plt provide us with visualization of the data.

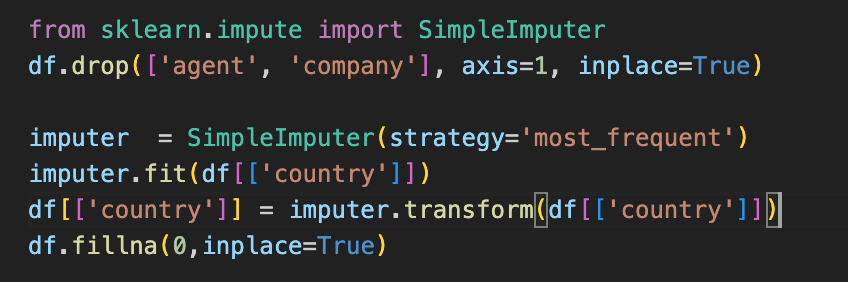
### Cleaning Data

After thoroughly examining the dataset using df.isnull().sum(), it becomes evident that certain features contain missing values. Addressing these missing values is crucial to ensure that our trained model performs optimally. Upon analysis, we identified four features with missing values as depicted in the figure below. The "country" feature has 488 missing values, the “children” column has 4 missing values, while the "agent" and "company" features exhibit 16,340 and 112,593 missing values, respectively.



*Figure 3 Missing values by using df.isnull().sum()*

Considering the impact of missing values, we deliberated on appropriate measures to handle this situation while preserving the integrity of the data and model performance. In this case, we determined that "agent" and "company" are not decisive factors for our analysis, as they merely represent the names of entities coordinating travel affairs. Consequently, we decided to drop these two columns. However, recognizing the potential insights lost through deletion, we opted for imputation for the remaining missing values.



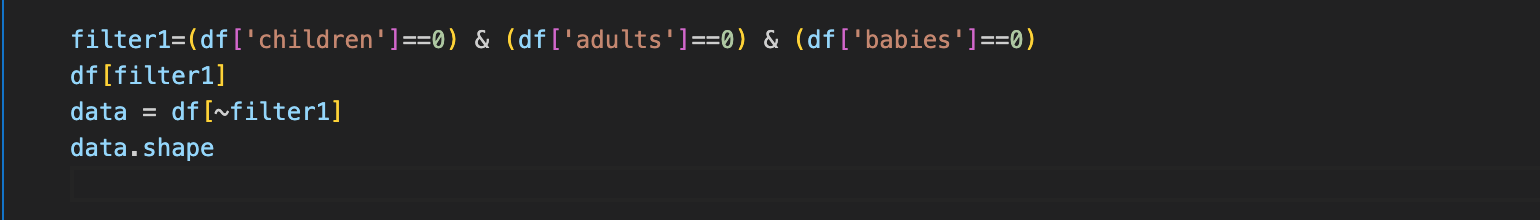
*Figure 4 Sample code for filling missing values*

Here we used SimpleImputer with the strategy “most\_frequent” to replace the 488 missing values in the “country column” with the most frequent value, which is Portugal in this case. After dropping the “agent” and “company” column and replacing the missing values in the “country” column with the most frequent value, we only have 4 missing values left from the “children” column. We simply fill these empty values with 0.



*Figure 5 after drop/impute*

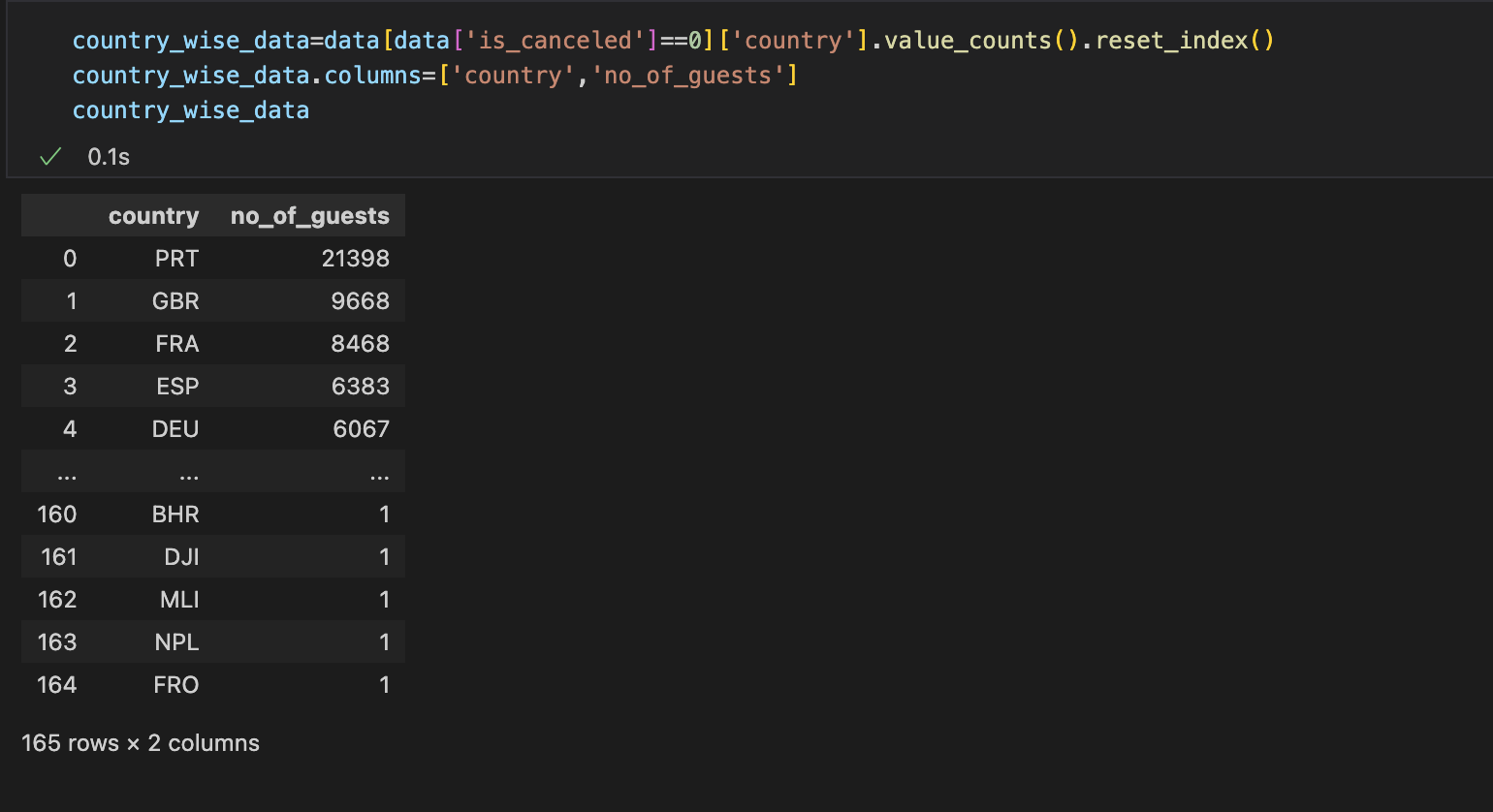
Since we simply replaced the missing values in the “children” column with 0. There is a chance that all three columns of “adult”, “children”, and “babies” are 0. Therefore to avoid that, we set a filter to filter out such cases.



*Figure 6 filter the invalid object*

### Data Visualization

We want to know where the guests come from. Here we selected all non-canceled bookings merged with the “country” column. We get the number of guests from each country.



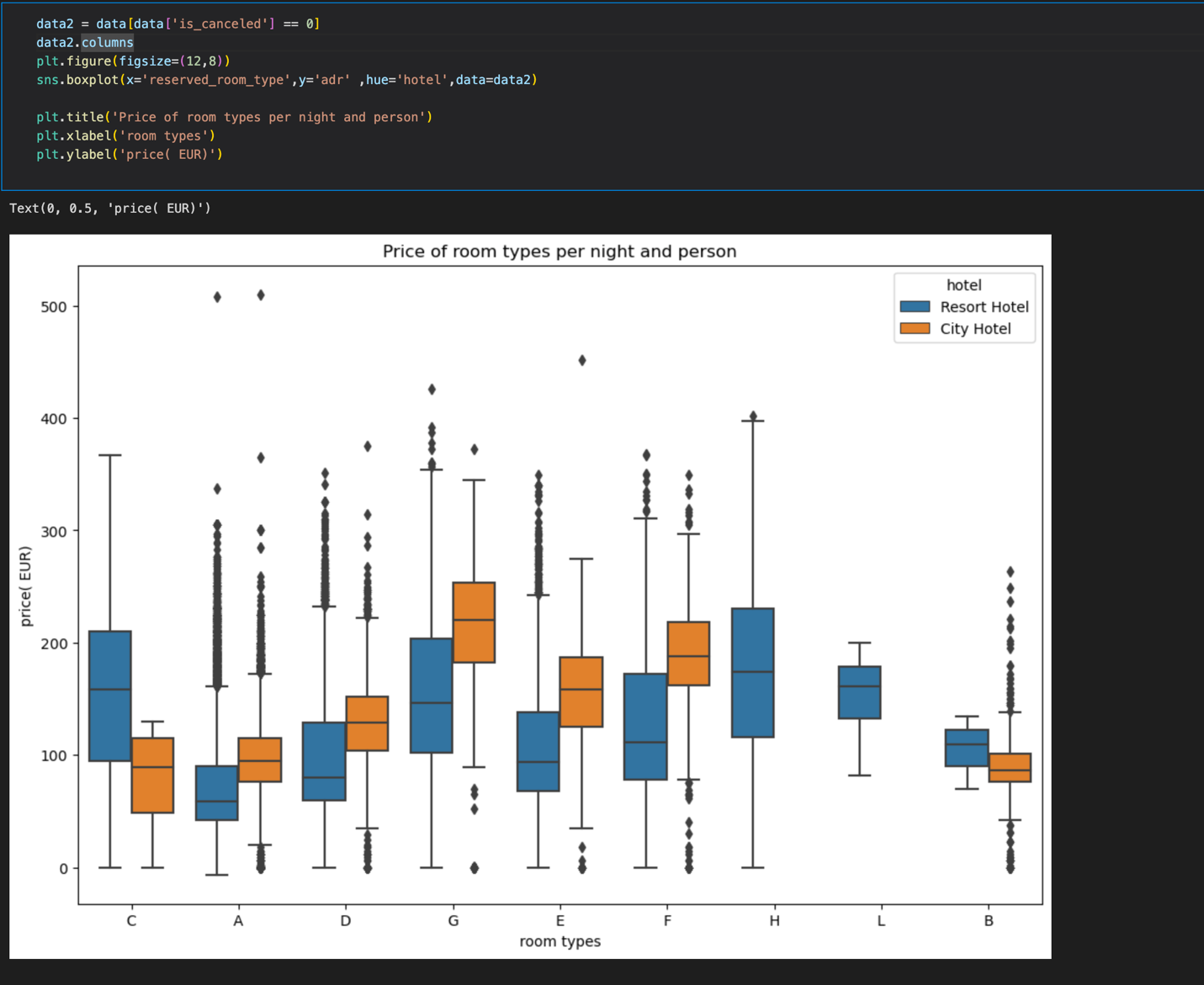
*Figure 7 Number of guests from each country*

Then we visualized these data into the world map to see the distribution of guests around the world. We used the choropleth() function in the plotly package to plot the map. In choropleth(), we set location = country\_wise\_data['country'], color=country\_wise\_data['no\_of\_guests'], hover\_name=country\_wise\_data['country'], title='home country of guests'. Figure 8 shows the code and the map. We can easily see that the country Portugal has the most number of outgoing tourists.



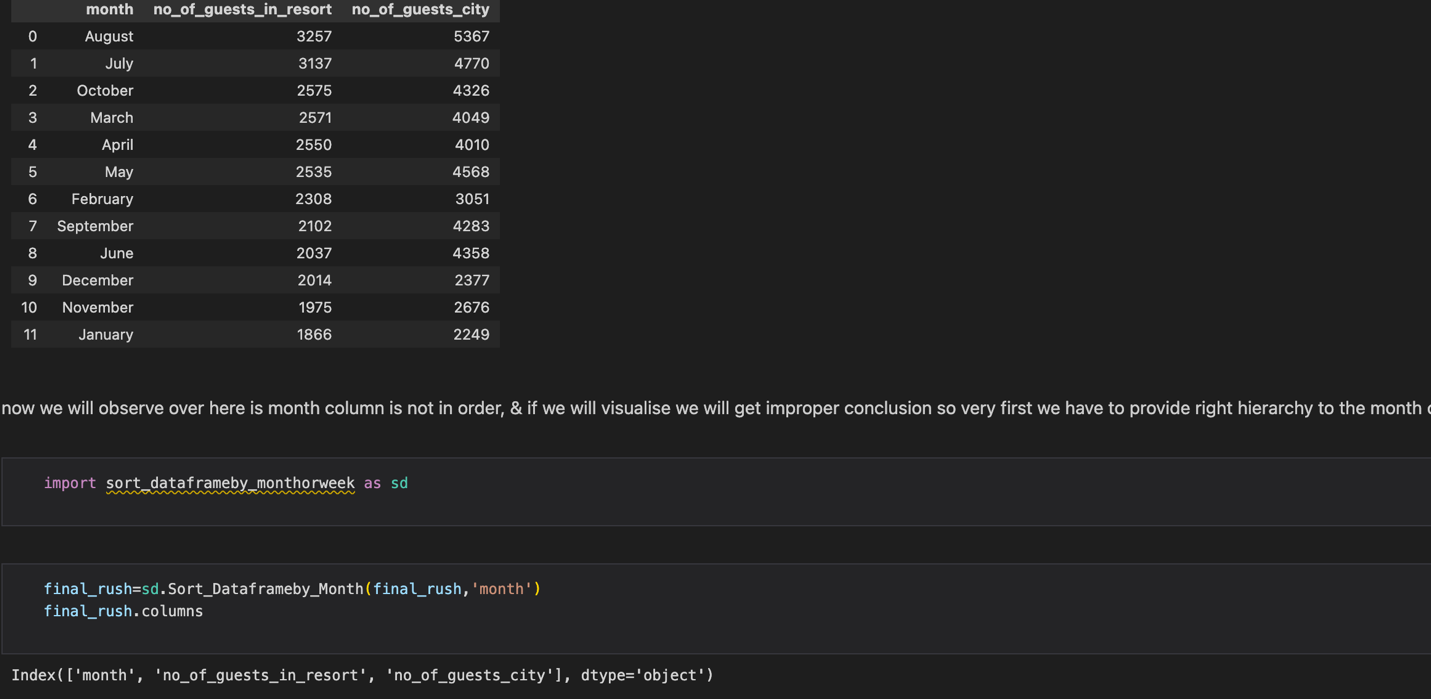
*Figure 8 The distribution of guests over the world*

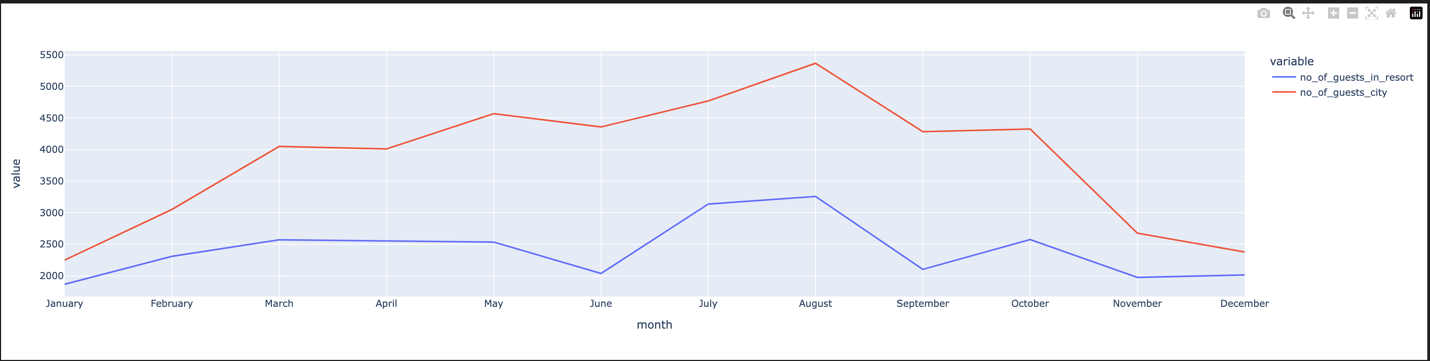
Secondly, we plot the boxplot to indicate the relationship between price distribution and room type. To get the relationship with price. We need to select the non-canceled bookings, then use sns.boxplot() to plot the chart. There is much insightful information we can gain from this chart. As we can observe the boxes of room type A and B for both resort and city hotel are comparatively short, from this phenomenon we can understand that these two room types have a relatively stable floating range of price. Vice versa, boxes that are tall show that the prices vary by a larger range.



*Figure 9 Code and output of the boxplot*

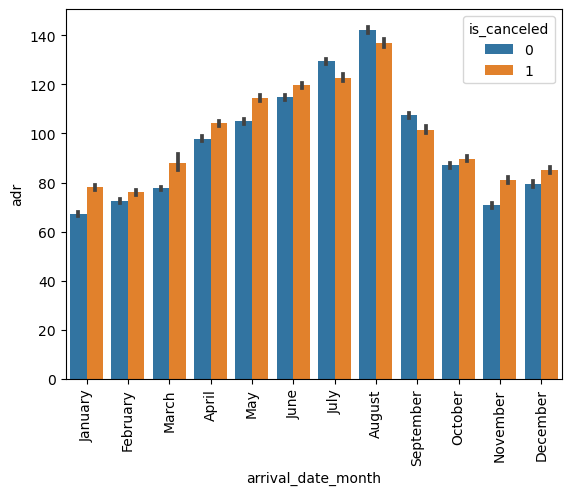
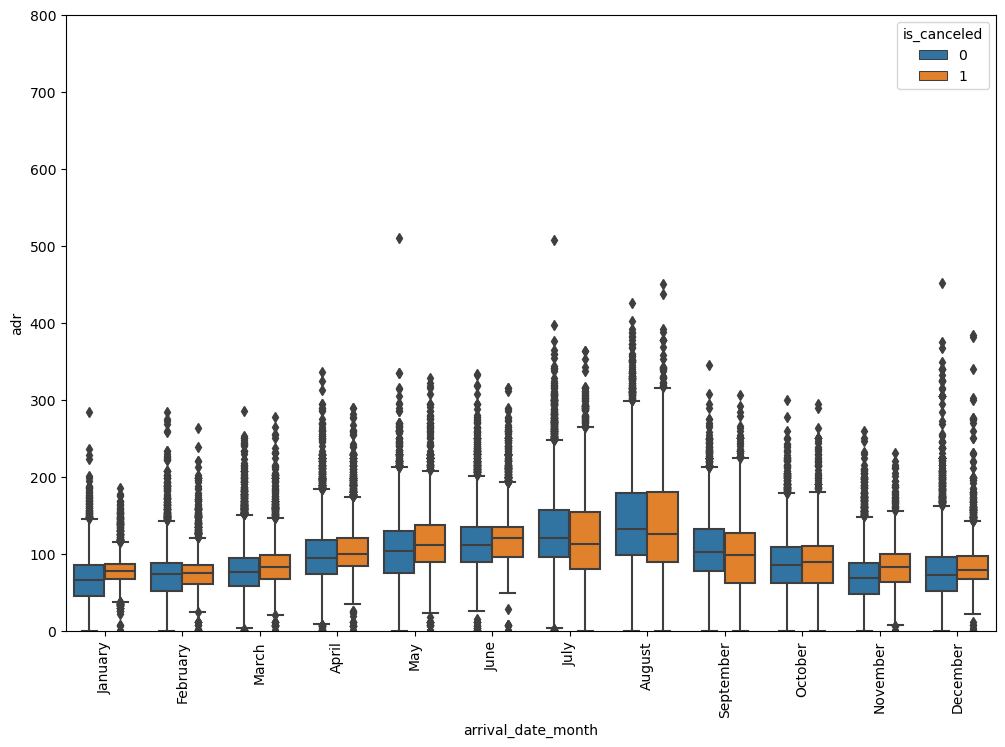
Thirdly, we want to know the number of guests received by different types of hotels in different months. We split data to 2 subsets by resort hotel and city hotel with the condition of being non-canceled bookings. Also we would need to use the Sort\_Dataframeby\_Month(final\_rush,'month') method from the sort\_dataframeby\_monthorweek package to sort the table by months. Finally, we plot the line chart.



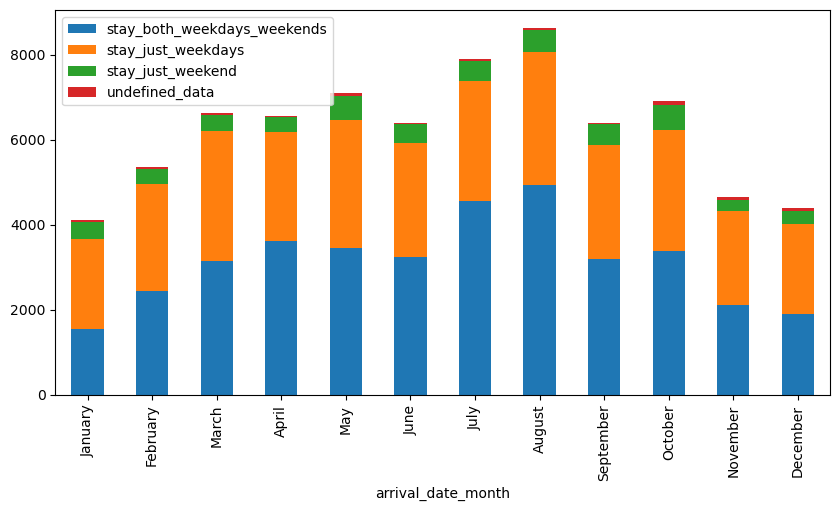


*Figure 10 Number of guests of different hotels in different months*

We have also shown months with adrs in bar chart and boxplot, and bookings were made only for weekdays or for weekends or for both.



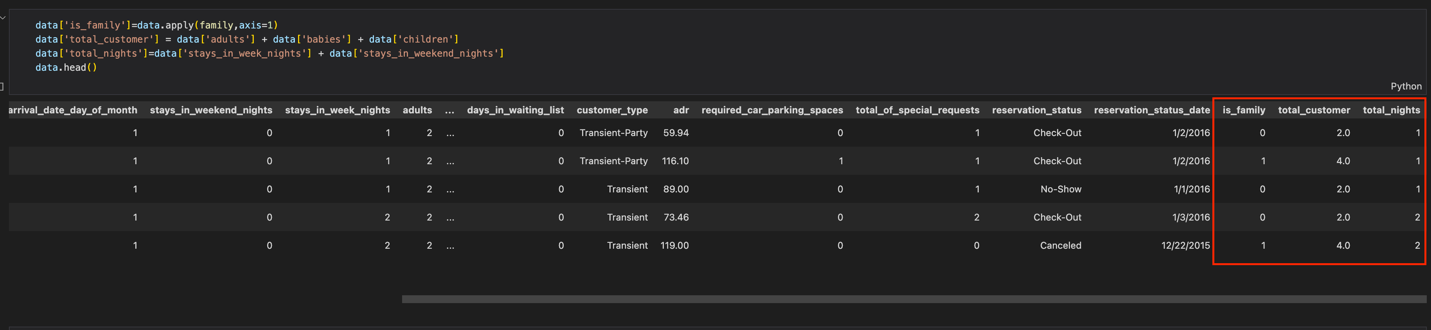
*Figure 11 boxplot and bar chart of price in different months*



*Figure 12 stacked bar chart on days of booking*

### Adding features

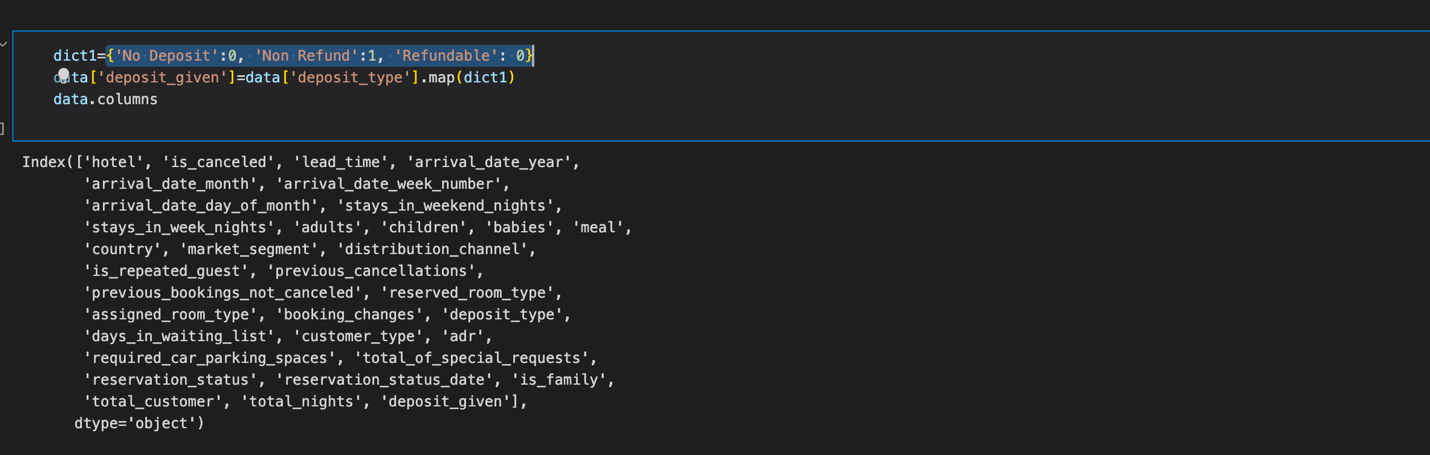
We wrote a method family() to check if guests are family. Then apply this method to the dataset, we add a column named “is\_family” with values of the family() method returned(1/0). We added a feature named “total\_customer” with the value of summation of the number of adults, children, and babies. We also added a “total\_nights” feature by summation of stays\_in\_week\_nights and stays\_in\_weekend\_nights



*Figure 13 after adding features*

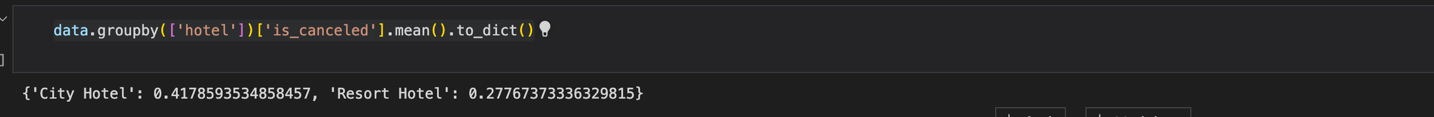
### Feature encoding

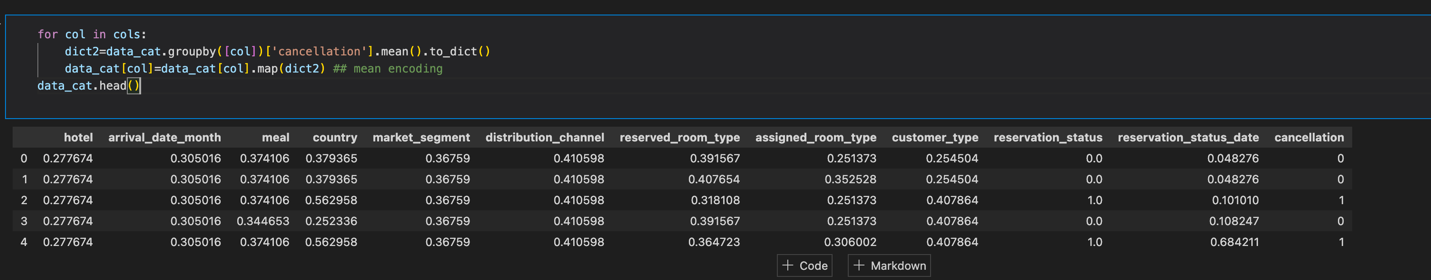
Here we encode for the categorical features. Firstly we look at “deposit\_type” feature. We created a dictionary with keys and values of {'No Deposit':0, 'Non Refund':1, 'Refundable': 0}. We apply this dictionary on “deposit\_type” column, then we can get a numerical column for “deposit\_types”(Non Refund->1/else->0) named “deposit\_given”



*Figure 14 change deposit\_type to deposit\_given*

Then we split the features into cat\_features and num\_features. For cat\_features, we encode each feature by taking the ratio of number of cancellations for each categorical value to the total number of reservations within that categorical value. For example, we encode the “hotel” value by taking the number of cancellations for each type of hotel(resort/city) to the total number of same type hotel reservations(with canceled or not canceled). So the encoded value for each categorical feature and value is the possibility of cancellation for each categorical value.





*Figure 15 Encoded value for categorical features*

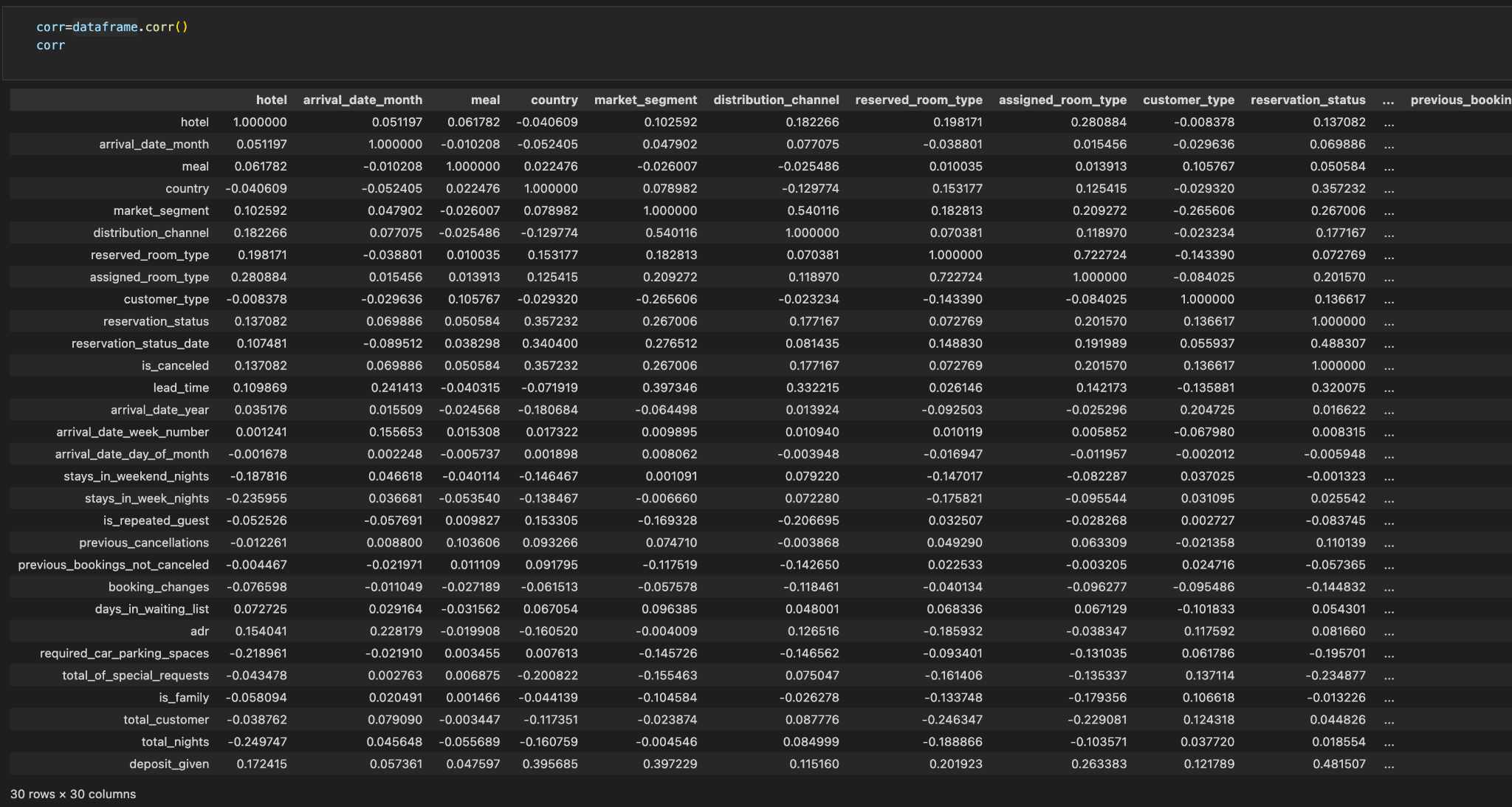
### Feature selection

#### Co-relation

We used corr() method to see the correlation between each field. If the correlation

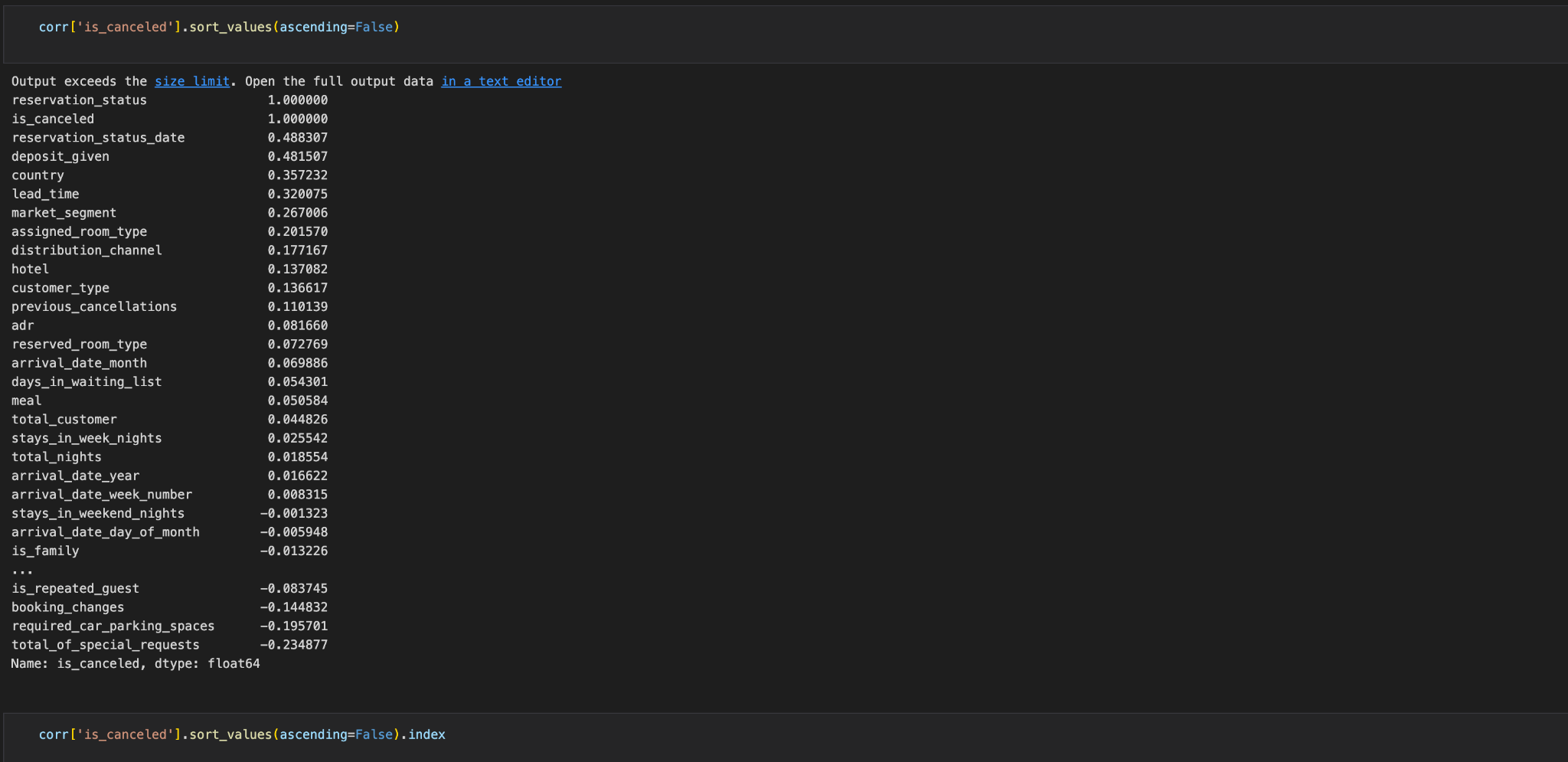
coefficient is near 1, there is a stronger correlation between 2 features. If the coefficient

is near 0, then there is no correlation between features.



*Figure 16 The output matrix of data set*

Then we get the correlation of features and “isCanceled” feature by this:



*Figure 17 The correlation for “is\_canceled” feature*

After reading the data in the table above, we choose to drop the features of

'reservation\_status', 'reservation\_status\_date','arrival\_date\_year',

'arrival\_date\_week\_number', 'stays\_in\_weekend\_nights', and

'arrival\_date\_day\_of\_month' since these features have low correlations with

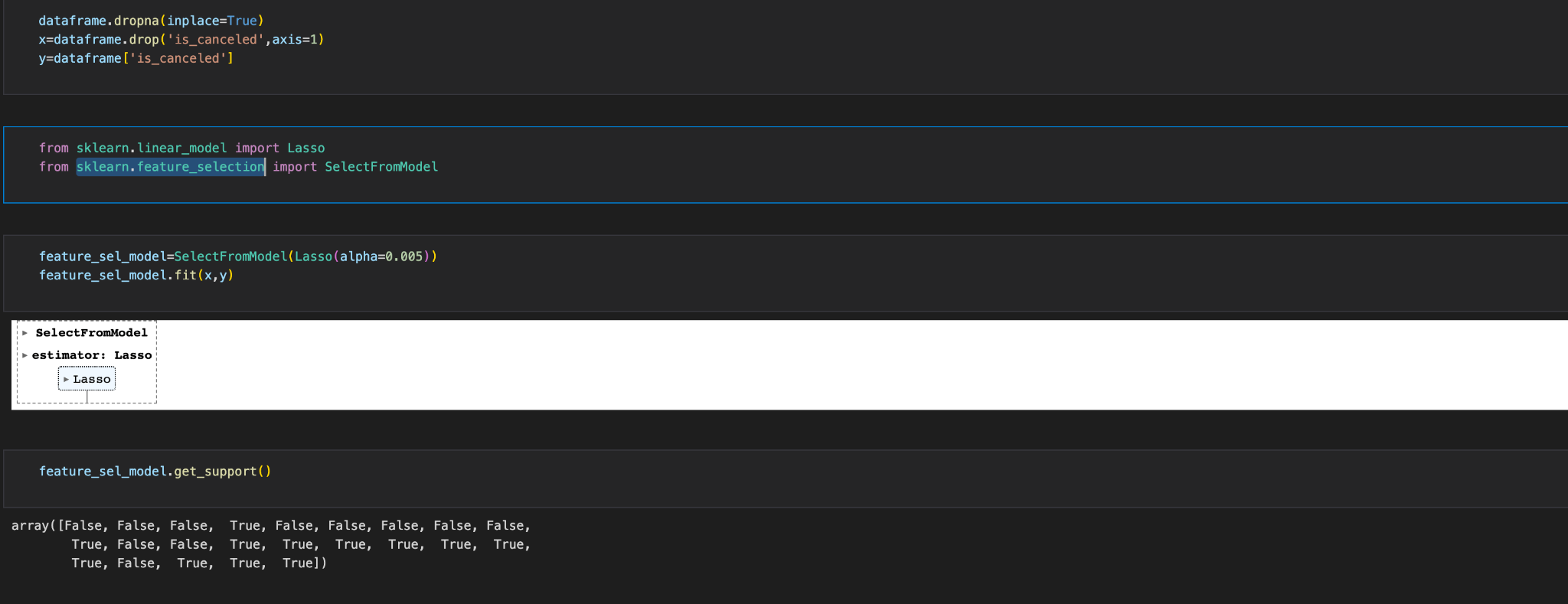
“is\_canceled” feature.

#### Feature selection

Then we split the data set into 2. The first set dropped “is\_cancelled” feature, and the

second set only has the “is\_canceled’ column(dependent set). We used

SelectFromModel() from sklearn.feature\_selection to further select important features.



*Figure 18 Important feature selection*

## Data Mining and Evaluation

While building an ML model, we typically need 2 types of data: training data and test data. Training data is the data that is used by the ML algorithm, it will learn some kind of relationship and pattern. Testing data is used to evaluate how well the ML model is performed, it is unseen to the ML model.

We create the training data by importing train\_test\_split from sklearn.model\_selection

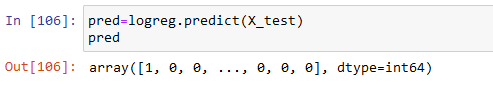


Generally speaking in ML, typically we can categorize into three: regression(continuous), classification(discrete), clustering(categorize). We are going to predict if a hotel booking is going to get canceled or not, there are only 2 outputs: 1/0. It is a typical classification use. There are tons of classification problems like: logistic, random forest, decision tree, KNN and so on.

Now we import logisticRegression from sklearn.linear\_model, then initialize the logistic regression, store it into logreg. The fit function trains a logistic regression model using the training data x\_train and corresponding labels y\_train. The logistic regression algorithm learns the optimal parameter by minimizing the difference between predicted outputs and actual target values. This is achieved through an optimization process such as gradient descent. Once the fit function finishes, the model is trained and can be used to make predictions on new, unseen data. Logistic regression is commonly used for binary classification tasks which is perfect for our purposes.



The predict method applies the learned parameters to the test data and generates the predicted labels. The resulting predictions are stored in the variable pred. This allows us to evaluate the performance of the model on unseen data or use the predictions for further analysis.



Let’s analyze the results by using confusion matrix:

1.True Positives: There are 13,944 instances that were correctly predicted as positive (belonging to the positive class). These are the cases where the model predicted a positive outcome, and the actual label was also positive.

2.True Negatives: There are 5,030 instances that were correctly predicted as negative (belonging to the negative class). These are the cases where the model predicted a negative outcome, and the actual label was also negative.

3.False Positives: There are 1,108 instances that were incorrectly predicted as positive. These are the cases where the model predicted a positive outcome, but the actual label was negative.

4.False Negatives: There are 3,760 instances that were incorrectly predicted as negative. These are the cases where the model predicted a negative outcome, but the actual label was positive.

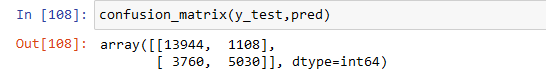
From these values, we can derive several evaluation metrics:

1.Accuracy: The overall accuracy of the model can be calculated as (TP + TN) / Total. In this case, it would be (13,944 + 5,030) / (13,944 + 1,108 + 3,760 + 5,030), which gives the proportion of correct predictions.

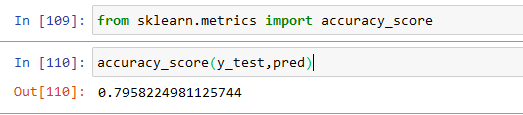
2.Precision: Precision measures the proportion of true positive predictions out of all positive predictions. It is calculated as TP / (TP + FP). In this case, it would be 13,944 / (13,944 + 1,108).

3.Recall (Sensitivity or True Positive Rate): Recall measures the proportion of true positive predictions out of all actual positive instances. It is calculated as TP / (TP + FN). In this case, it would be 13,944 / (13,944 + 3,760).

4.Specificity (True Negative Rate): Specificity measures the proportion of true negative predictions out of all actual negative instances. It is calculated as TN / (TN + FP). In this case, it would be 5,030 / (5,030 + 1,108).



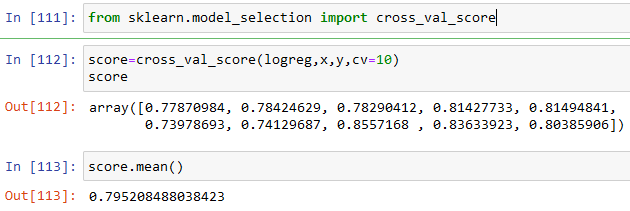
Now let us calculate the accuracy using accuracy\_score from sklearn.metrics:



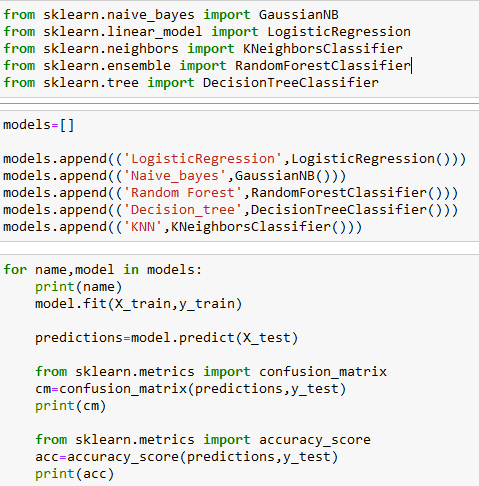
The model has an accuracy of approximately 79.58%.

Next step is to use cross validation for our model. It is a technique that assesses the model’s performance by splitting the data into 10 subsets and evaluating the model on each subset while using the remaining subsets for training.

The cross-validated scores range from approximately 73.98 to 85.57, and the mean is 79.52.



We also used other algorithms from sklearn.

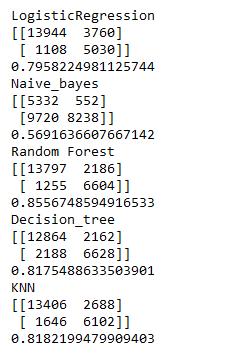


The analysis of the classification models reveals varying performance levels. The Logistic Regression model achieved an accuracy of approximately 79.58%, indicating that it correctly predicted the class labels for around 79.58% of the instances. The corresponding confusion matrix shows a relatively balanced distribution of true positives, true negatives, false positives, and false negatives. In contrast, the Naive Bayes model had a lower accuracy of about 56.92%, suggesting that it struggled to accurately classify the instances. The confusion matrix for Naive Bayes reveals a higher number of false positives and false negatives compared to true positives and true negatives.

On the other hand, the Random Forest model performed well with an accuracy of around 85.57%. The confusion matrix demonstrates a higher number of true positives and true negatives, indicating that the model successfully predicted these instances. Similarly, the Decision Tree model achieved an accuracy of approximately 81.75% with a relatively balanced distribution of true positives, true negatives, false positives, and false negatives.

The K-Nearest Neighbors (KNN) model also performed well, achieving an accuracy of about 81.82%. The confusion matrix suggests a higher number of true positives and true negatives, showing its ability to classify instances accurately.

Comparing the models, Random Forest and KNN demonstrate relatively higher accuracy compared to Logistic Regression, Naive Bayes, and Decision Tree. However, it's important to consider additional evaluation metrics and potentially fine-tune the models' hyperparameters to further optimize their performance.



## Conclusion

In conclusion, this machine learning project focused on developing a model to predict hotel booking cancellations. The aim was to provide a valuable tool for optimizing resource allocation and profit in the hospitality industry. Through the analysis and processing of a comprehensive dataset, we employed various techniques and methodologies to ensure the data was fit for analysis.

The data cleaning process involved addressing missing values in specific features. By dropping columns such as "agent" and "company" that were not crucial for the analysis, and imputing missing values in the "country" and "children" columns, we prepared the dataset for further analysis.

Data visualization played a crucial role in understanding the insights and patterns within the dataset. We explored the distribution of guests from different countries, analyzed the relationship between price and room type using box plots, and examined the number of guests received by different types of hotels in different months using line charts. These visualizations provided valuable insights into the data.

Additional features were added to enhance the model's predictive capabilities. Features such as "is\_family" indicating whether guests are a family, "total\_customer" representing the total number of guests, and "total\_nights" capturing the total duration of stays were incorporated into the dataset.

Feature encoding was performed to convert categorical features into numerical values. This encoding process involved calculating the probability of cancellation for each categorical value, thereby enriching the dataset.

Feature selection was conducted using correlation analysis and SelectFromModel. Features with low correlations with the target feature "is\_canceled" were dropped, and important features were selected for model training.

The machine learning phase involved splitting the dataset into training and testing data. Logistic Regression was implemented as the classification algorithm to predict hotel booking cancellations. Evaluation metrics such as accuracy, precision, recall, and specificity were calculated, demonstrating the model's performance. Cross-validation was also employed to assess the model's performance across different subsets of the data.

Among the models evaluated, Random Forest and K-Nearest Neighbors (KNN) demonstrated higher accuracy compared to Logistic Regression, Naive Bayes, and Decision Tree. However, further optimization and fine-tuning of the models' hyperparameters may be necessary to enhance their performance.

Overall, this project aimed to contribute to the optimization of hotel operations and the efficiency of hotel booking platforms by developing a machine learning model for predicting hotel booking cancellations. The findings and insights obtained from this project can assist hotel industry professionals in resource allocation and decision-making processes.