CS584 Final Group Project

Adam Sadek, Kamen Petkov, Sakher Yaish

Professor Oleksandr Narykov

https://github.com/adamsadek813/CS584GroupProject.git

Introduction:

The hotel booking dataset we are analyzing contains data from a hotel industry context, detailing various aspects of customer bookings. It encompasses a wide range of information such as the type of hotel (Resort or City), the time between the booking and the actual stay (lead time), arrival details (year, month, day, and week number), stay duration (weekends and weeknights), the number of adults, children, and babies, and other booking details like the type of meal booked, country of origin, market segment, booking changes, repeated guest status, deposit type, and the number of special requests made.

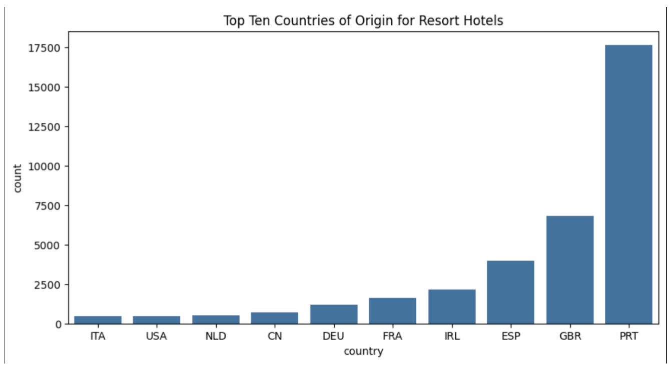
Crucially, it includes a binary 'is_canceled' column indicating whether the booking was canceled, which serves as the target variable for predictive modeling. The dataset is comprehensive, allowing for a multifaceted analysis of booking patterns and serving as a rich resource for building predictive models to forecast cancellations, understand customer behavior, and optimize booking strategies.

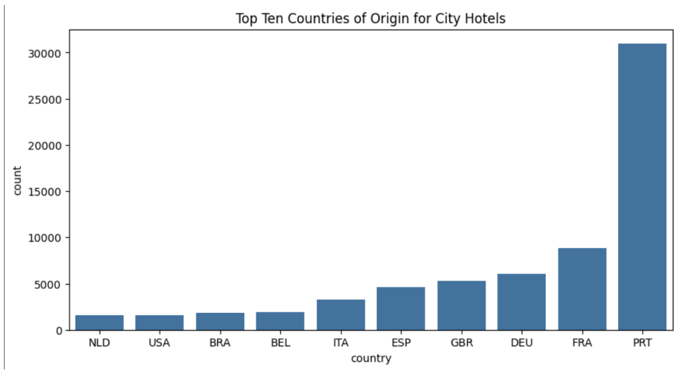
Exploring the Data:

To start, we decided to identify different significant values and metrics, and quantify valuable information. We extracted the following information from the dataset:

- Number of Resort Hotels: 39596
- Number of City Hotels: 79306
- Number of guests who canceled reservation: 44157
- Number of guests who did not cancel the reservation: 74745
- Customer type with the most reservations is Transient with 89174 reservations
- 2 customers required the most number of parking spaces 8
- 111592 customers required the least number of parking spaces 0
- 87.6377% of the people who expressed a room preference during reservation got the room during check-in.

Then, we divided the hotels into their two types: resort and city. We identified that due to the dramatically higher percentage of hotels from PRT (Portugal), the dataset very likely originated from or near Portugal.

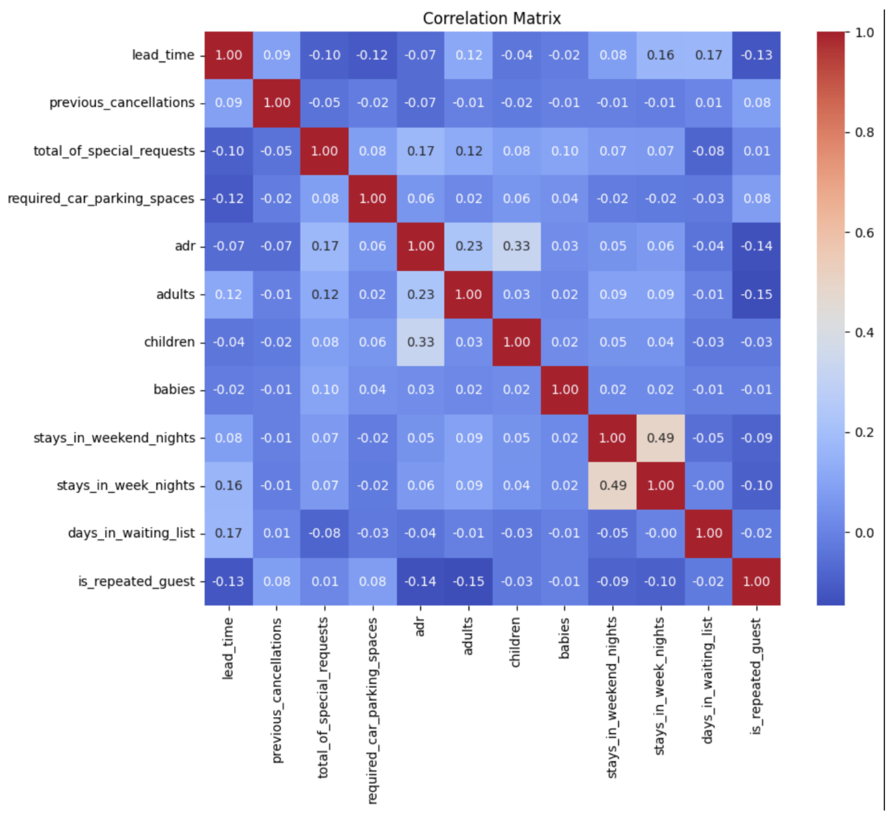




Correlation Matrix:

This is the result of the correlation matrix that was made on the raw data:

High Correlation:



"stays_in_weekend_nights" and "stays_in_week_nights" have the highest positive value of 0.49; it suggests that guests who stay more nights during the week also tend to stay more nights over the weekend.

No or Weak Correlation: Many of the features seem to have no correlation. For instance, "children" and "required_car_parking_spaces" have a value close to 0; it suggests that the number of children does not linearly affect the need for car parking spaces.

Potential Redundancy: If two features have a very high correlation, they may be conveying similar information, which could be redundant. In such cases, you might consider removing or combining these features to simplify the model.

Random Forest Classifier with K-Folds Cross-Validation:

After one-hot encoding the categorical features, we fit the random forest classifier on the data using 10 trees. The k-fold cross-validator used 5 splits and shuffled the data. Our average cross validation accuracy score was 88.53% with a standard deviation of 0.16%. The random forest model worked quite well and was able to determine cancellations with high accuracy.

We then split the data into training (60%), validation (20%), and test (20%) sets. The accuracy of the random forest on the datasets was as followed:

- Training set accuracy: 99.57%
- Validation set accuracy: 88.02%
- Test set accuracy: 87.89%

These are the results of the confusion matrix done on the test set:

Accuracy: 0.8788528657331484

Error: 0.12114713426685164

Balanced Accuracy: 0.8630129575921806

Specificity: 0.9255754647339105

Sensitivity (Recall): 0.8004504504504505

Precision: 0.8650359011804795

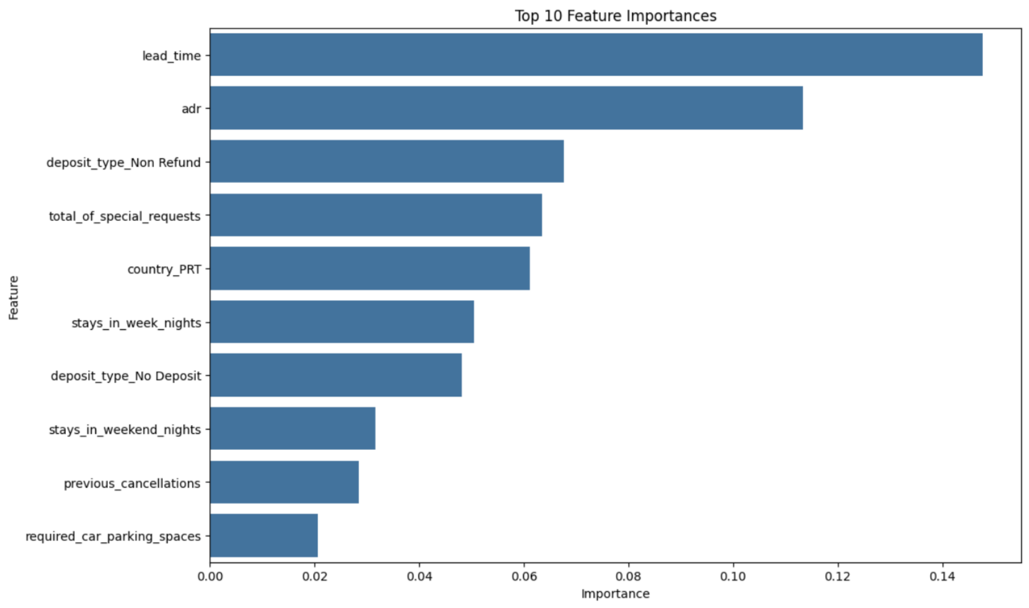
F1 Score: 0.831490904837106

ROC AUC Score: 0.8630129575921804

The balanced accuracy, considering both specificity (92.56%) and sensitivity (80.05%), is at 86.30%, suggesting a good balance in predicting both classes. The F1 score of 83.15% and a ROC AUC score of 86.30% further indicate a strong performance, particularly in terms of the model's ability to balance precision and recall efficiently.

We saw that our model performed well on the training set and on the validation set as well. This means that our model was not overfitting or at least not too much. It appeared that decreasing the dimensionality may benefit the model as some of the features were not important and have strong collinearity with other features.

Using the random forest, we identified the 10 most important features in the dataset. These were the results:



Manual Decrease of Features and Increase of Trees in Random Forest:

Using the list of most important features, we decided to only consider those features and test out the same model. The features in use are: 'lead_time', 'total_of_special_requests', 'adr', 'country', and 'deposit_type'. The results of the k-means cross validation is as follows: Average cross-validation accuracy: 79.37%. Standard deviation of the accuracy scores: 0.50%. These values show a decrease in accuracy which is to be expected. The results are not dramatically different which is a good sign. After splitting the data into training, validation, and test, these were the following accuracies:

- Training set accuracy: 99.95%
- Validation set accuracy: 77.91%
- Test set accuracy: 80.41%

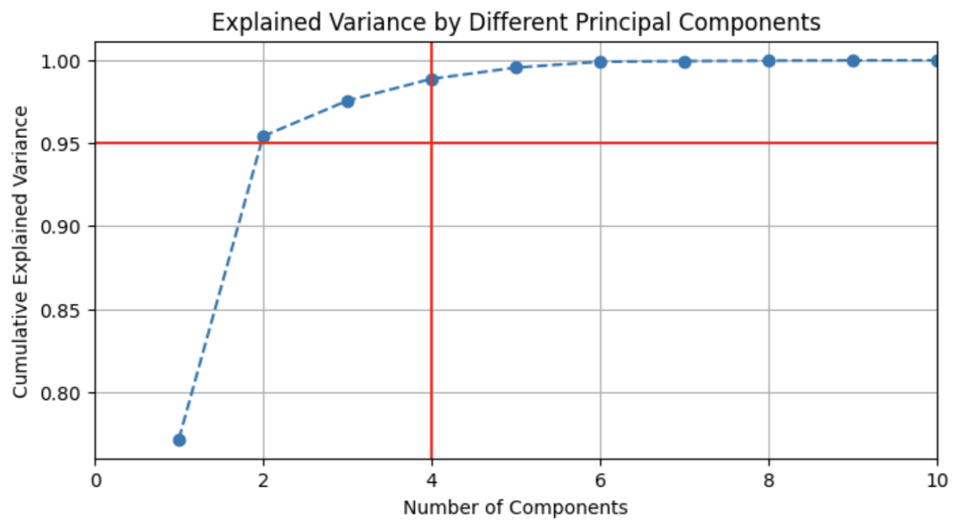
We saw noticeable overfitting as there was a discrepancy between the training set accuracy and validation set accuracy. This indicated that the model needed tuning. To decrease the overfitting, we increased the number of trees from 10 to 50 and decreased the max depth of trees to 3. The results are as follows:

- Training set accuracy: 74.66%
- Validation set accuracy: 74.27%
- Test set accuracy: 74.84%

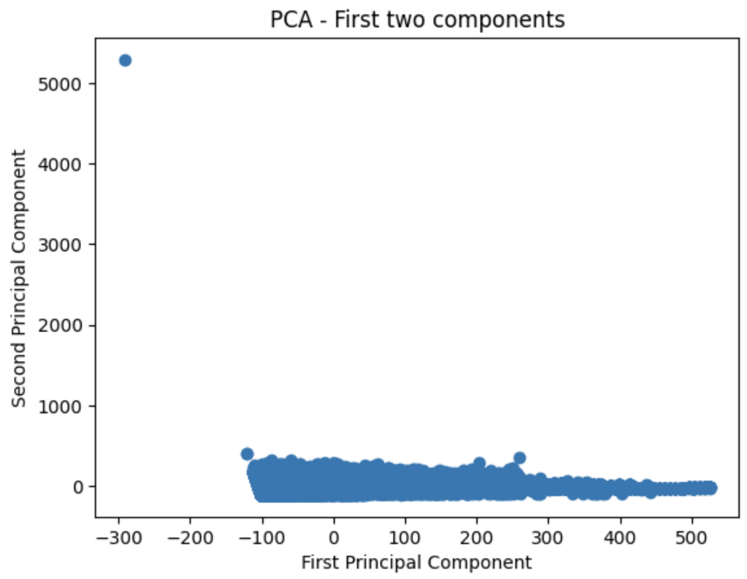
The results show less overfitting but the performance of our model suggests that we used too few features and the model can't learn well

Dimensionality Reduction with PCA:

Using PCA, we calculated the explained variance by different principal components. The results show that the cumulative variance is largely explained by the first 2 components, then as components are added the increase in explained variance plateaus.



We then fit the train, validation, and test data using the PCA model using 2 components. The plot of the training data is as shown:



Using k-fold cross validation on the random forest classifier with the PCA pipeline yielded an average cross-validation accuracy of 80.12% with a standard deviation of 0.19%. This is a high accuracy with a low standard deviation.

This is the confusion matrix on the test set:

Accuracy: 0.7903368235145705

Error: 0.20966317648542954

Balanced Accuracy: 0.7617110768912662

Specificity: 0.8747735051338836

Sensitivity (Recall): 0.6486486486486487

Precision: 0.7553107789142408

F1 Score: 0.697928026172301

ROC AUC Score: 0.7617110768912662

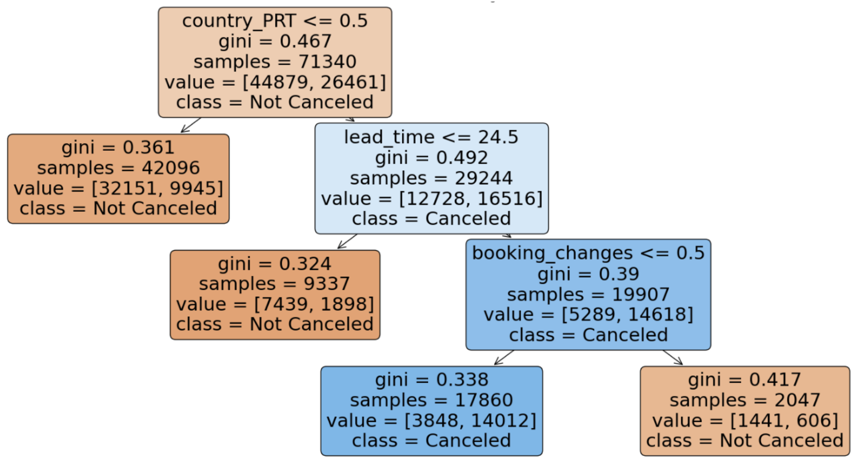
Unfortunately, we noticed overfitting in the model since the training set had an accuracy of 100% while the validation set had an accuracy of 80.88%. The drop in performance on the validation (80.88%) and test sets (80.24%) indicates that the model is not generalizing as well to new, unseen data. This is a common challenge when the dimensionality reduction either retains too much complexity (leading to overfitting) or loses significant information pertinent to prediction (underfitting in a way).

PCA aims to reduce dimensions while retaining as much variance as possible, but this doesn't always equate to retaining predictive accuracy. The key in PCA is to find the right number of components that capture the essential information without holding onto the noise and redundancy. If too many components are retained, the model may overfit; if too few are kept, it may miss critical information for making accurate predictions.

Single Decision Tree Model

We then decided to use a single decision tree model, only using the important features that we decided on. Our decision tree has the following properties:

- Max depth: 5
- Min samples split: 20
- Max leaf nodes: 10
- Min impurity decrease: 0.01



The tree gave us the top 3 important features to be country_PRT, lead_time, and booking_changes. Using cross-validation we got an average cross-validation accuracy of 76.88% with a standard deviation of 0.40%. This was the confusion matrix for the test set:

Accuracy: 0.7664522097472772

Error: 0.23354779025272276

Balanced Accuracy: 0.7163967092721875

Specificity: 0.9140997248506811

Sensitivity (Recall): 0.5186936936936937

Precision: 0.782534828406388

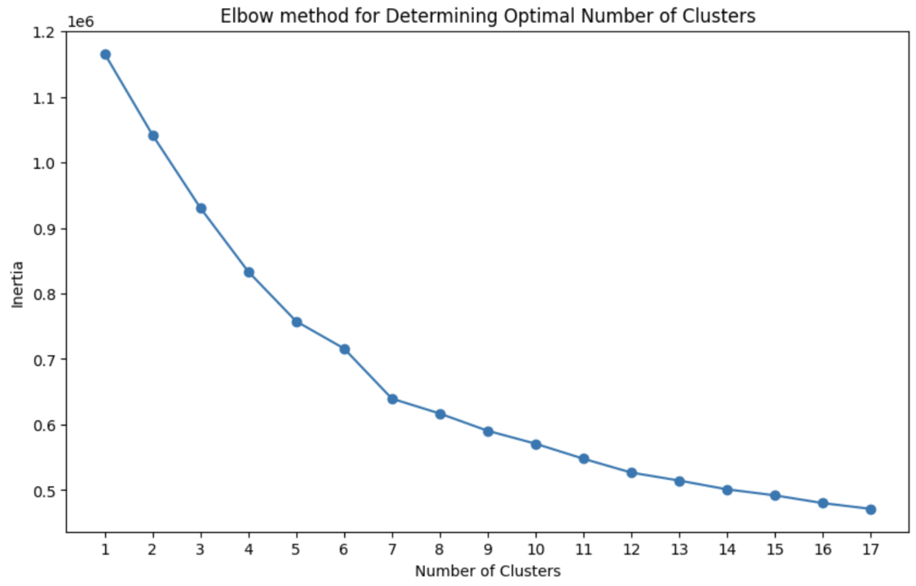
F1 Score: 0.6238656372748205

ROC AUC Score: 0.7241346263718923

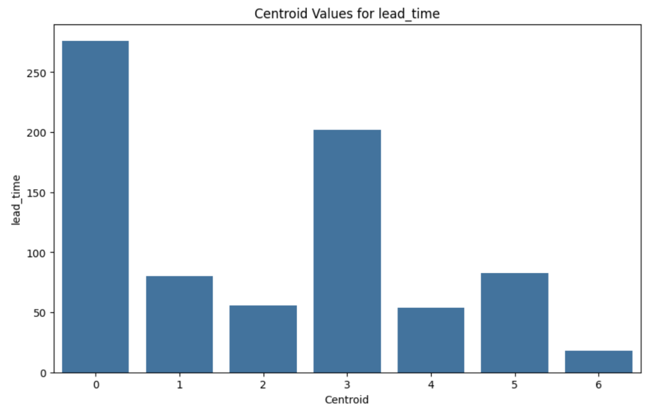
The results indicate that the simple decision tree model demonstrates moderate overall performance with a validation accuracy of 77.42% and a slightly lower overall accuracy of 76.65%. The model shows a high specificity (over 91%), indicating it is effective at correctly identifying negative cases, but a lower sensitivity (around 52%), suggesting it's less adept at correctly identifying positive cases. The balanced accuracy, which is a more representative metric in case of imbalanced datasets, stands at around 72% for both validation and overall, further supporting the model's moderate performance. The F1 scores, which balance precision and recall, are relatively moderate (around 63%), and the similar ROC AUC scores (approximately 72%) indicate a decent but not exceptional ability to discriminate between the classes. This performance profile suggests room for improvement, possibly through model complexity adjustments or feature engineering.

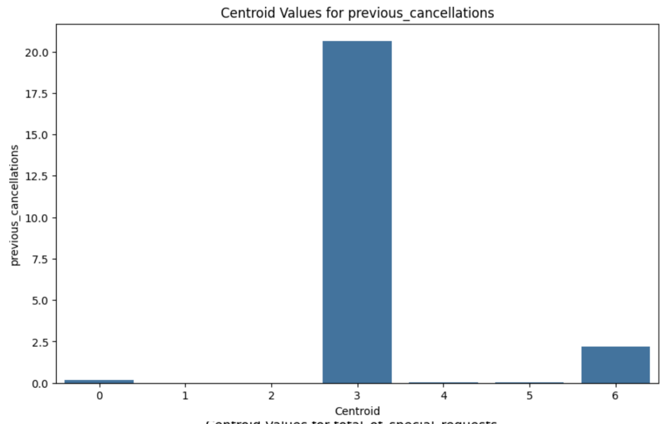
K-means:

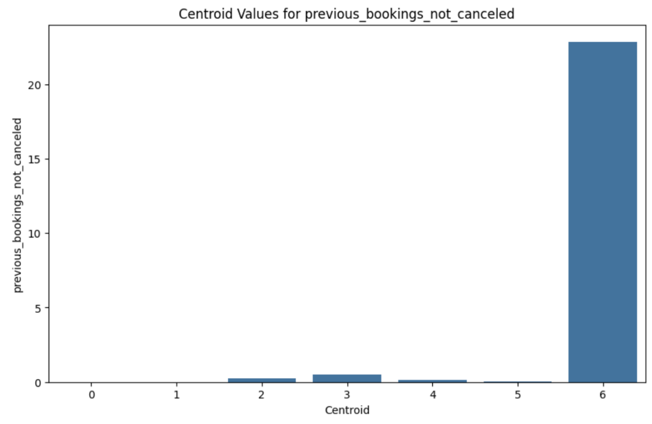
We used k-means to perform customer segmentation on the dataset. We started by converting categorical variables like ‘country’, ‘market_segment’, and ‘customer_type’ into numerical form using encoding techniques. We then scaled the numerical data so that each feature contributed equally to the distance computations in K-Means. We utilized the elbow method to determine the optimal number of clusters. This was the graph we obtained that made us choose 7 as the optimal number of clusters:

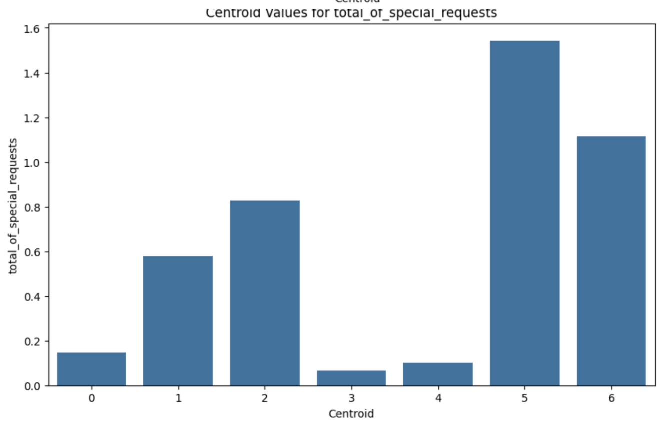


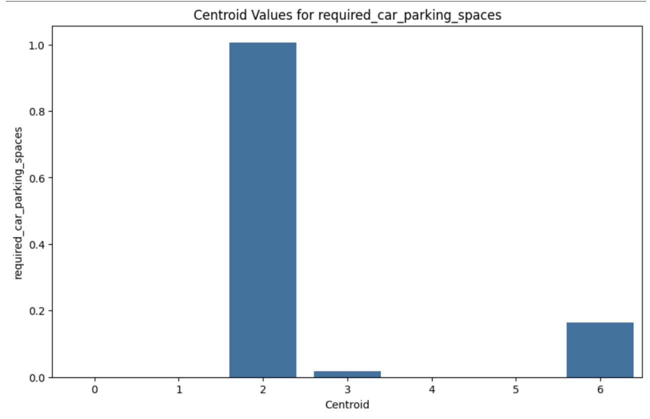
From these 7 clusters, we were able to identify noticeable trends relevant to each feature. Note that each cluster is numbered and in each graph below they keep their same number.

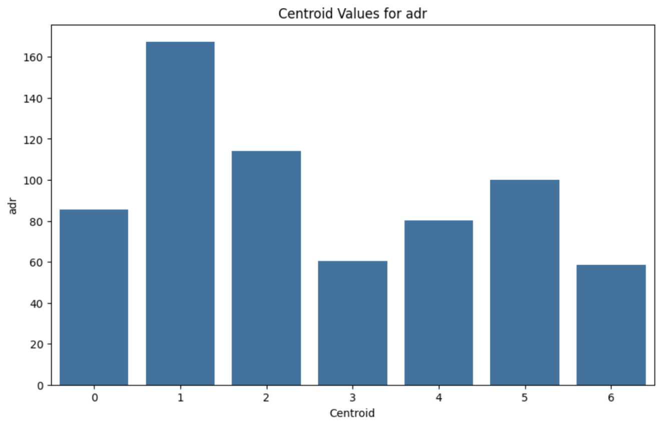












The clusters that have high lead time represent the guests who make reservations ahead of time. The clusters with a high value of previous cancellations, tend to cancel their reservations. Hotels will value the most the guests who book ahead of time and don't cancel. We can see that for the most part, the clusters (of guests) tend to book about 20-100 before showing up and they don't cancel their reservation.
Also, we can notice that customers who had previous reservations are not likely to have previous cancellations. + They tend to reserve 200+ days ahead.

We can also see that the customers who want parking spots, make special requests, and book about 50 days in advance, tend to not cancel their reservations. → This data suggests that families are more responsible and show up when making a reservation (because they are usually the ones having all of those requirements).

<u>Neural Network:</u>

For our neural network model, we first scaled the data then implemented early stopping. We used the keras library for our neural network model. We had 4 layers in our model – 1 input, 2 hidden, and 1 output. All layers used ReLU activation function except for the output layer, which used linear activation function. This is because the neural network had to output a binary prediction for the is_canceled attribute.

After our first neural network implementation, our confusion matrix on the test set was as follows:
Accuracy: 0.6312026913372581
Error: 0.36879730866274185
Balanced Accuracy: 0.5
Specificity: 1.0
Sensitivity (Recall): 0.0
Precision: 0.0
F1 Score: 0.0
ROC AUC Score: 0.5
What stands out is the low accuracy (63%), low sensitivity (0.0), precision (0.0), and F1 score (0.0). This model did not work so well, unfortunately. Thus, we decided to create another–simpler–model using MLPRegressor instead of keras. We also decreased the learning rate from 0.001 to 0.0001. We implemented three versions of this model, varying the activation function each time (tanh, logistic, ReLU). The results were much improved:

<u>Tanh activation function:</u>	<u>Logistic activation function:</u>	<u>ReLU activation function:</u>
Accuracy: 0.9985281749369218	Accuracy: 0.9480656013456686	Accuracy: 0.9976871320437343
Error: 0.0014718250630781915	Error: 0.05193439865433136	Error: 0.0023128679562657295
Balanced Accuracy: 0.998360082256071	Balanced Accuracy: 0.9311301017869501	Balanced Accuracy: 0.9974568455693165
Specificity: 0.9990006662225184	Specificity: 0.9956695536309127	Specificity: 0.9983344437041972

Sensitivity (Recall): 0.9977194982896237	Sensitivity (Recall): 0.8665906499429875	Sensitivity (Recall): 0.9965792474344356
Precision: 0.9982886480319453	Precision: 0.9915198956294846	Precision: 0.9971477467199087
F1 Score: 0.9980039920159681	F1 Score: 0.9248554913294798	F1 Score: 0.9968634160250927
ROC AUC Score: 0.9983600822560711	ROC AUC Score: 0.9311301017869501	ROC AUC Score: 0.9974568455693165

These results are much better. Namely, the accuracy and balanced accuracy are in the mid-to-high 90% range, suggesting that the model performs equally well on both classes. The specificity, sensitivity, and precision are all also high, suggesting that the model does a great job identifying which hotel bookings are canceled and which are not.