Forecasting Hotel Pricing

Group 19: Adam Mills, Trisha Nguyen

MATH 4322 — Fall 2025

Introduction

Description of Data

The global hospitality industry increasingly relies on data-driven insights to refine pricing strategies and improve customer satisfaction. Forecasting room prices is essential not only for effective revenue management but also for helping customers plan affordable stays. Motivated by the challenge of price volatility, our project develops predictive models using the "Hotel Booking" dataset from Kaggle, which contains over 36,000 observations and 19 variables. The dataset contains booking details like guest count, room type, and month of stay. Using *avg_room_price* as the response variable, we evaluate how booking behavior and timing influence hotel pricing.

We used two predictive modeling approaches. Linear Regression will help us explore and quantify linear relationships between the predictors and the average room price, providing interpretable estimates of how each factor influences pricing. Decision Trees, on the other hand, provide a non-linear approach that can capture complex interactions and threshold effects that may not be visible in a purely linear framework. Model performance and reliability are assessed using K-Fold Cross-Validation, which will allow us to evaluate and compare both models across multiple training and validation splits. This approach supports a more robust estimate of predictive accuracy and helps identify the model that generalizes best to unseen data.

The work on our project is structured collaboratively but with clearly defined responsibilities. Adam is responsible for implementing and testing the Linear Regression model, while Trisha is focused on developing and evaluating the Decision Tree model. Both team members will contribute to data cleaning and preparation, cross-validation, and the final analysis report. By integrating statistical and machine learning methods, this project aims to provide practical insights into hotel pricing dynamics and assist customers in making informed booking decisions.

Description of Variables

- *ID:* Unique identifier of each booking (exclude)
- *n_adults:* Number of adults
- *n_children:* Number of Children
- *weekend_nights:* Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- *week_nights:* Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- *meal_plan:* Type of meal plan booked by the customer
- *car_parking_space:* Does the customer require a car parking space? (0 - No, 1- Yes)

- *room_type:* 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
- *year:* Year of arrival date
- *month:* Month of arrival date
- *date:* Date of the month
- *market_segment:* Market segment designation.
- *repeated_guest:* Is the customer a repeated guest? (0 - No, 1- Yes)
- *previous_cancellations:* Number of previous bookings that were canceled by the customer prior to the current booking
- *previous_bookings_not_canceled:* Number of previous bookings not canceled by the customer prior to the current booking
- *avg_room_price:* Average price per day of the reservation; prices of the rooms are dynamic. (in euros)
- *special_requests:* Total number of special requests made by the customer (e.g., high floor, view from the room, etc.)
- *status:* Flag indicating if the booking was canceled or not.

Main Question

How do both time-related factors and customer characteristics, such as month, year, and number of guests, influence predicted room prices?

Methods and Results

Linear Regression Model (Adam Mills)

We chose Linear Regression to explore and quantify linear relationships between the features and the average room price, providing interpretable estimates of how each factor influences pricing and enabling forecasts that generalize to new booking data. The advantages of this method are its simplicity and transparency, making it easy to implement and understand. A limitation of this approach is its reliance on a strictly linear relationship between the features and the outcome, which may oversimplify the complexity of hotel pricing patterns. The outliers and multicollinearity can also influence the model, leading to instability or overfitting. Our strategy for overcoming these issues involves narrowing down features with stepwise selection, managing unusual data points, and using cross-validation to test how well the model generalizes.

Our objective is to forecast the average room price using booking and time-related features. In this section, we apply a Linear Regression model to establish a baseline for prediction and to

identify which variables most strongly influence hotel pricing. We begin by fitting an initial regression model to the dataset, comparing all selected features against the *avg_room_price*.

Model Formula:

$$avg\_room\_price = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \ldots + \beta_{17} X_{17} + \varepsilon$$

In this model, *avg_room_price* is expressed as a linear combination of all relevant booking features. Each term represents a predictor of the room price, while β_0 is the intercept and ε captures unexplained variation. We exclude *ID* from the model because it is a unique identifier and provides no predictive value. The predictors enter the model in the following order: X_1 corresponds to *n_adults*; X_2 represents *n_children*; X_3 and X_4 capture *weekend_nights* and *week_nights*; X_5 reflects *meal_plan*; X_6 indicates *car_parking_space*; X_7 represents *room_type*; X_8 is *lead_time*; X_9, X_{10}, and X_{11} correspond to *year*, *month*, and *date* of arrival; X_{12} captures *market_segment*; X_{13} indicates *repeated_guest*; X_{14} and X_{15} represent *previous_cancellations* and *previous_bookings_not_canceled*; X_{16} reflects *special_requests*; and X_{17} captures *status*.

```
hotel_data.lm <- lm(avg_room_price ~ ., data = hotel_data)
summary(hotel_data.lm)
```

```
Call:
lm(formula = avg_room_price ~ ., data = hotel_data)

Residuals:
    Min      1Q  Median      3Q     Max
-198.68  -12.27   -1.28   11.41  449.47

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -4.221e+04  7.271e+02 -58.053  < 2e-16 ***
n_adults                     9.030e+00  2.571e-01  35.121  < 2e-16 ***
n_children                   1.063e+01  4.047e-01  26.267  < 2e-16 ***
weekend_nights              -2.565e+00  1.370e-01 -18.717  < 2e-16 ***
week_nights                 -5.471e-01  8.605e-02  -6.358 2.07e-10 ***
meal_planMeal Plan 2         2.245e+01  4.389e-01  51.151  < 2e-16 ***
meal_planMeal Plan 3        -1.641e+01  9.920e+00  -1.654   0.0981 .
meal_planNot Selected       -1.294e+01  3.692e-01 -35.041  < 2e-16 ***
car_parking_space            9.312e+00  6.795e-01  13.704  < 2e-16 ***
room_typeRoom_Type 2        -9.197e+00  8.817e-01 -10.431  < 2e-16 ***
room_typeRoom_Type 3         1.628e+01  8.318e+00   1.957   0.0504 .
room_typeRoom_Type 4         1.561e+01  3.525e-01  44.298  < 2e-16 ***
room_typeRoom_Type 5         2.918e+01  1.368e+00  21.334  < 2e-16 ***
room_typeRoom_Type 6         5.437e+01  9.773e-01  55.632  < 2e-16 ***
room_typeRoom_Type 7         5.934e+01  1.820e+00  32.599  < 2e-16 ***
lead_time                   -1.036e-01  1.750e-03 -59.219  < 2e-16 ***
year                         2.095e+01  3.602e-01  58.151  < 2e-16 ***
```

```
month2                           7.412e-02  8.785e-01   0.084   0.9328
month3                           8.107e+00  8.353e-01   9.706  < 2e-16 ***
month4                           1.728e+01  8.214e-01  21.033  < 2e-16 ***
month5                           3.243e+01  8.286e-01  39.141  < 2e-16 ***
month6                           3.408e+01  8.086e-01  42.145  < 2e-16 ***
month7                           2.995e+01  8.296e-01  36.097  < 2e-16 ***
month8                           3.180e+01  8.086e-01  39.322  < 2e-16 ***
month9                           4.265e+01  7.952e-01  53.637  < 2e-16 ***
month10                          3.388e+01  7.884e-01  42.967  < 2e-16 ***
month11                          1.895e+01  8.208e-01  23.089  < 2e-16 ***
month12                          1.612e+01  8.212e-01  19.630  < 2e-16 ***
date                            -6.819e-04  1.330e-02  -0.051   0.9591
market_segmentComplementary     -9.255e+01  2.302e+00 -40.203  < 2e-16 ***
market_segmentCorporate         -3.458e-02  2.053e+00  -0.017   0.9866
market_segmentOffline            2.434e+00  2.016e+00   1.208   0.2272
market_segmentOnline             1.415e+01  2.006e+00   7.055 1.75e-12 ***
repeated_guest1                 -1.087e+01  9.686e-01 -11.217  < 2e-16 ***
previous_cancellations           7.747e-01  3.660e-01   2.116   0.0343 *
previous_bookings_not_canceled -5.558e-01  8.368e-02  -6.642 3.14e-11 ***
special_requests                 2.191e+00  1.751e-01  12.514  < 2e-16 ***
statusNot_Canceled              -8.058e+00  3.040e-01 -26.512  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 21.98 on 36237 degrees of freedom
Multiple R-squared:  0.608,    Adjusted R-squared:  0.6076
F-statistic:  1519 on 37 and 36237 DF,  p-value: < 2.2e-16
```

```
BIC(hotel_data.lm)
```

```
[1] 327509.2
```

Feature Selection:

The strongest predictors of *avg_room_price* are *n_adults*, *n_children*, *meal_plan*, *room_type*, *lead_time*, *year*, and *month*, all of which show high t-values and p-values < 2e-16. These features indicate that booking timing, seasonal patterns, room category, and party size are major drivers of hotel pricing. Several *month* categories (March through December) are also statistically significant, suggesting meaningful seasonal variation in room prices.

Within the categorical variables, most *room_type* levels are significant (5 out of 6), indicating that room category influences price. Similarly, *meal_plan* shows strong significance for two of its three levels. In contrast, only one of the *market_segment* categories (Online) is significant, while the Corporate and Offline segments do not affect price. The variable *date* is not predictive because its coefficient is essentially zero and its p-value is extremely high (p = 0.9591), indicating that the specific day of the month has no measurable effect on room price once month,

seasonality, and other booking characteristics are accounted for. This conclusion is further supported by running a backward stepwise selection procedure, which identified *date* as the only variable whose removal lowered AIC, confirming that it does not contribute to the model and should be excluded.

The full model yields a BIC of 327509.2, which provides a baseline for comparison.

The reduced model formula is: *avg_room_price* = *n_adults* + *n_children* + *weekend_nights* + *week_nights* + *meal_plan* + *car_parking_space* + *room_type* + *lead_time* + *year* + *month* + *market_segment* + *repeated_guest* + *previous_cancellations* + *previous_bookings_not_canceled* + *special_requests* + *status*

Results:

The reduced Linear Regression model was fit using all the features except *date*, which was removed due to its lack of statistical relevance.

```
reduced_model.lm <- lm(avg_room_price~. -date, data=hotel_data)
summary(reduced_model.lm)
```

```
Call:
lm(formula = avg_room_price ~ . - date, data = hotel_data)

Residuals:
    Min      1Q  Median      3Q     Max
-198.68  -12.27   -1.28   11.41  449.47

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -4.221e+04  7.271e+02 -58.054  < 2e-16 ***
n_adults                     9.030e+00  2.571e-01  35.123  < 2e-16 ***
n_children                   1.063e+01  4.046e-01  26.268  < 2e-16 ***
weekend_nights              -2.565e+00  1.370e-01 -18.723  < 2e-16 ***
week_nights                 -5.470e-01  8.604e-02  -6.358 2.07e-10 ***
meal_planMeal Plan 2         2.245e+01  4.388e-01  51.166  < 2e-16 ***
meal_planMeal Plan 3        -1.641e+01  9.920e+00  -1.654   0.0980 .
meal_planNot Selected       -1.294e+01  3.692e-01 -35.043  < 2e-16 ***
car_parking_space            9.312e+00  6.794e-01  13.705  < 2e-16 ***
room_typeRoom_Type 2        -9.198e+00  8.816e-01 -10.433  < 2e-16 ***
room_typeRoom_Type 3         1.628e+01  8.318e+00   1.957   0.0504 .
room_typeRoom_Type 4         1.561e+01  3.524e-01  44.310  < 2e-16 ***
room_typeRoom_Type 5         2.918e+01  1.368e+00  21.335  < 2e-16 ***
room_typeRoom_Type 6         5.437e+01  9.773e-01  55.634  < 2e-16 ***
room_typeRoom_Type 7         5.934e+01  1.820e+00  32.600  < 2e-16 ***
lead_time                   -1.036e-01  1.750e-03 -59.224  < 2e-16 ***
year                         2.095e+01  3.602e-01  58.152  < 2e-16 ***
```

```
month2                           7.408e-02  8.785e-01   0.084   0.9328
month3                           8.107e+00  8.352e-01   9.707  < 2e-16 ***
month4                           1.728e+01  8.212e-01  21.039  < 2e-16 ***
month5                           3.243e+01  8.284e-01  39.152  < 2e-16 ***
month6                           3.408e+01  8.083e-01  42.166  < 2e-16 ***
month7                           2.995e+01  8.292e-01  36.116  < 2e-16 ***
month8                           3.180e+01  8.084e-01  39.333  < 2e-16 ***
month9                           4.265e+01  7.950e-01  53.646  < 2e-16 ***
month10                          3.388e+01  7.879e-01  42.995  < 2e-16 ***
month11                          1.895e+01  8.195e-01  23.128  < 2e-16 ***
month12                          1.612e+01  8.212e-01  19.630  < 2e-16 ***
market_segmentComplementary     -9.255e+01  2.302e+00 -40.204  < 2e-16 ***
market_segmentCorporate         -3.477e-02  2.053e+00  -0.017   0.9865
market_segmentOffline            2.435e+00  2.016e+00   1.208   0.2271
market_segmentOnline             1.415e+01  2.006e+00   7.056 1.75e-12 ***
repeated_guest1                 -1.086e+01  9.685e-01 -11.218  < 2e-16 ***
previous_cancellations           7.749e-01  3.660e-01   2.117   0.0343 *
previous_bookings_not_canceled -5.558e-01  8.367e-02  -6.643 3.13e-11 ***
special_requests                 2.191e+00  1.751e-01  12.516  < 2e-16 ***
statusNot_Canceled              -8.058e+00  3.039e-01 -26.514  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 21.98 on 36238 degrees of freedom
Multiple R-squared:  0.608,    Adjusted R-squared:  0.6076
F-statistic:  1561 on 36 and 36238 DF,  p-value: < 2.2e-16
```

```
BIC(reduced_model.lm)
```

```
[1] 327498.7
```

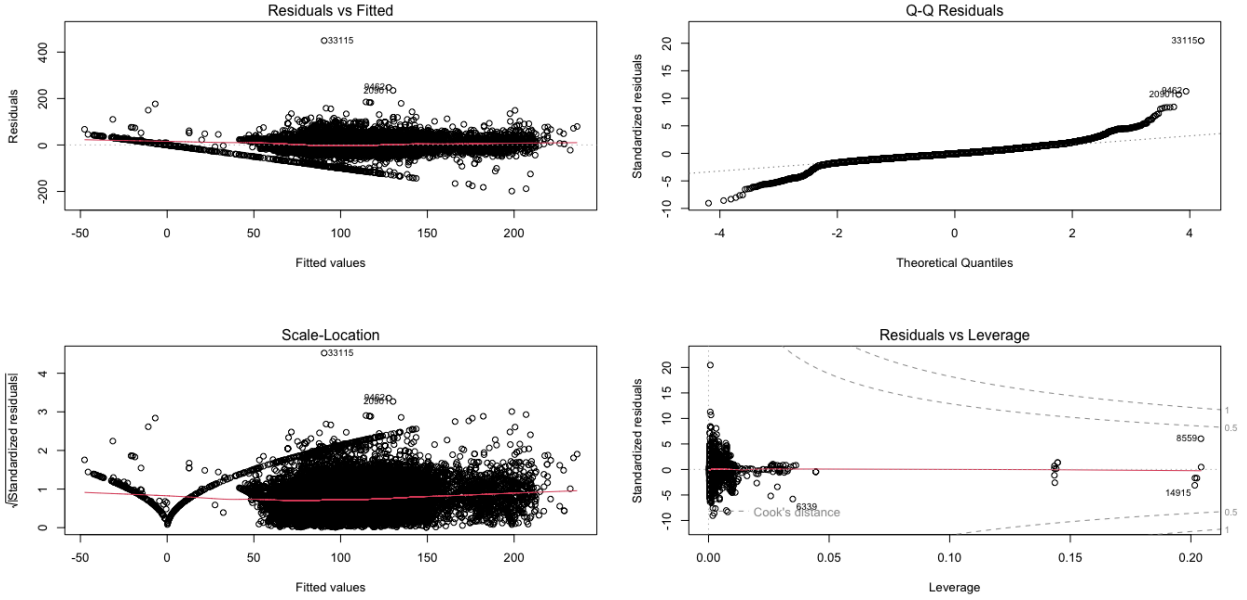
Here, the BIC value for the reduced model (327498.7) is slightly lower than the full model's BIC of 327509.2, indicating that removing the non-informative *date* feature results in a marginally better balance between model fit and complexity. The Multiple R-squared value of 0.608 means that about 60.8% of the variation in average room price is explained by the features in the reduced model.

The model results show that both time-related factors and customer characteristics play substantial roles in determining predicted room prices. Among the time-related variables, *month* and *year* are highly significant, indicating strong seasonal and annual pricing patterns. Prices increase sharply in months 3 through 12 (March through December), reflecting higher demand during most of the year, while the positive year coefficient suggests that room prices have risen over time. The *lead_time* feature also has a meaningful effect: longer booking lead times are associated with lower predicted prices, consistent with early-booking discounts or lower demand far in advance.

Customer characteristics also influence predicted room prices. The number of guests, represented by *n_adults* and *n_children*, is strongly significant, and each additional guest increases the expected room price. Room-related choices such as *room_type*, *meal_plan*, and *special_requests* also contribute substantially, reflecting higher pricing for upgraded rooms, added services, and additional amenities. Behavioral variables, including *repeated_guest*, *previous_cancellations*, and *previous_bookings_not_canceled*, show smaller but statistically meaningful effects, suggesting that guest history influences pricing through loyalty or risk-related adjustments.

To further assess the validity of the reduced model, I examined the standard diagnostic plots.

```
par(mfrow = c(2,2))
plot(reduced_model.lm)
```



The diagnostic plots for the reduced model show generally acceptable behavior. The Residuals vs Fitted plot suggests a mostly linear relationship, though slight curvature and variance spread indicate mild heteroscedasticity. The Q-Q plot shows that residuals are approximately normal, with minor deviations at the tails. The Scale-Location plot confirms that variance increases slightly with fitted values, supporting the presence of mild heteroscedasticity. The Residuals vs Leverage plot identifies a few influential observations, but none appear to dominate the model. Overall, the assumptions of linearity, normality, and constant variance are reasonably satisfied.

To evaluate predictive performance, we randomly subdivided the data into 80% training and 20% testing sets. This process was repeated 10 times, using set.seed(i) to ensure reproducibility across

iterations. For each split, the model was trained on the training data and used to predict the average room price on the held-out test set.

```
MSE = rep(0,10)
for (i in 1:10){
  set.seed(i)

  train_index = sample(1:nrow(hotel_data),.8*nrow(hotel_data))
  train = hotel_data[train_index, -c(2,4)]
  test = hotel_data[-train_index, -c(2,4)]

  hotel_data.lm=lm(avg_room_price~., data=train)
  yhat=predict(hotel_data.lm, newdata=test)

  MSE[i]= mean((yhat-test$avg_room_price)^2)}

MSE
```

```
[1] 481.9246 508.9345 503.0250 504.2433 454.6783 486.1004 481.7311 514.3730
484.4369 509.9358
```

```
mean_testMSE <- mean(MSE)
mean_testMSE
```

```
[1] 492.9383
```

```
rmse <- sqrt(mean_testMSE)
rmse
```

```
[1] 22.20221
```

The mean test MSE across all 10 runs is 492.9383. This corresponds to an average root mean squared error (RMSE) of 22.20221, which is very close to the model's training RMSE of approximately 21.98. This similarity indicates that the model generalizes well to unseen data and does not show signs of overfitting. Overall, the repeated train/test evaluation confirms that the reduced model provides stable and reliable predictions of average room price.

To further validate the model's stability, a 10-fold cross-validation procedure was performed. The cross-validated MSE was 484.3507, yielding an RMSE of 22.00797. Practically, this means that the model's predicted room prices are typically within 22 euros of the actual price. This value aligns closely with both the training RMSE and the repeated test-set RMSE, demonstrating that the reduced model maintains consistent predictive accuracy across multiple resampling strategies. The agreement among these error metrics provides strong evidence that the model generalizes reliably and is not overly sensitive to how the data is partitioned.

In summary, we used a structured linear regression process to understand how time-related factors and customer characteristics influence hotel room prices. We began by preparing the data, removing irrelevant identifiers, converting categorical variables, and checking for missing values. We then fit an initial model using all available features and evaluated its performance using statistical measures such as p-values, t-values, and BIC. To refine the model, we examined a correlation heatmap to identify redundant numeric features and applied backward stepwise selection, which revealed that *date* did not contribute significantly and could be removed. After fitting the reduced model, we assessed its assumptions through diagnostic plots and evaluated its predictive accuracy using repeated train/test splits and 10-fold cross-validation. Across all validation methods, the model consistently predicted room prices with an average error of about 22 euros, demonstrating strong stability and generalizability.

Decision Trees (Trisha Nguyen)

We chose Decision Trees because they capture non-linear relationships and interactions among predictors when forecasting the average room price. Unlike Linear Regression, which assumes additive effects, Decision Trees split the data into rules that reveal complex pricing patterns and highlight the most important variables. This structure allows the model to uncover threshold effects (e.g., certain room types or months where prices change sharply) and combinations of factors that drive price differences. An additional advantage is interpretability: the final tree can be visualized and translated.

Model Formula:

avg_room_price ~ month + room+type + market_segment

```
Call:
rpart(formula = avg_room_price ~ month + room_type + market_segment,
    data = seasonal_data, method = "anova")
  n= 36275

          CP nsplit rel error    xerror        xstd
1 0.18664296      0 1.0000000 1.0000734 0.011919439
2 0.06308598      1 0.8133570 0.8134922 0.010548029
3 0.05143742      3 0.6871851 0.6873807 0.009148781
4 0.02852785      4 0.6357477 0.6359393 0.008649008
5 0.02380324      5 0.6072198 0.6074431 0.008735198
6 0.02195916      6 0.5834166 0.5813358 0.008683640
7 0.01656339      7 0.5614574 0.5616744 0.008494522
8 0.01000000      8 0.5448940 0.5451329 0.008575796

Variable importance
     room_type market_segment          month
            56             33             11
```

In this decision tree model, *avg_room_price* is the response variable and the predictors are:

- *month*: the month in which the booking occurs (time-related factor)
- *room_type*: categorical indicator of the room category
- *market_segment*: information on customer type and booking channel

This specification reflects our focus on understanding how temporal factors, product differentiation, and customer/booking channel characteristics jointly shape hotel pricing.

Results:

To evaluate the decision tree model and reduce the chance that our results were driven by random variation, a repeated train–test procedure with 10 iterations was implemented.

```
# Perform 10 iterations with different train-test splits
for (i in 1:10) {
  set.seed(i * 42)  # Ensures reproducibility with a unique seed for each iteration

  # Randomly select 80% of the data for training
  train_index <- sample(1:nrow(seasonal_data), size = 0.8 * nrow(seasonal_data))
  train_data  <- seasonal_data[train_index, ]
  test_data   <- seasonal_data[-train_index, ]

  # Build the decision tree model
  seasonal_price_tree <- rpart(
    avg_room_price ~ month + room_type + market_segment,
    data   = train_data,
    method = "anova"
  )

  plot(seasonal_price_tree, main = paste("Decision Tree - Iteration", i), cex = 0.7)
  text(seasonal_price_tree, pretty = 0, cex = 0.7)

  # Predict avg_room_price on the test data and calculate MSE
  predictions <- predict(seasonal_price_tree, newdata = test_data)
  MSE[i] <- mean((test_data$avg_room_price - predictions)^2)
}
```

In each iteration, 80% of the data was randomly selected as the training set, and the remaining 20% was reserved as the test set. The training data was used to fit and refine the model, while the test data provided an unbiased estimate of its predictive performance. This design allowed us to assess whether the model was overfitting (performing well on the training data but poorly on new data) or underfitting (failing to capture meaningful structure in the data). Across these iterations, the mean squared error (MSE) on the test sets was 658.6132. Because this value indicated relatively high prediction error, we proceeded to prune the tree in order to obtain a more accurate and better-generalizing model.

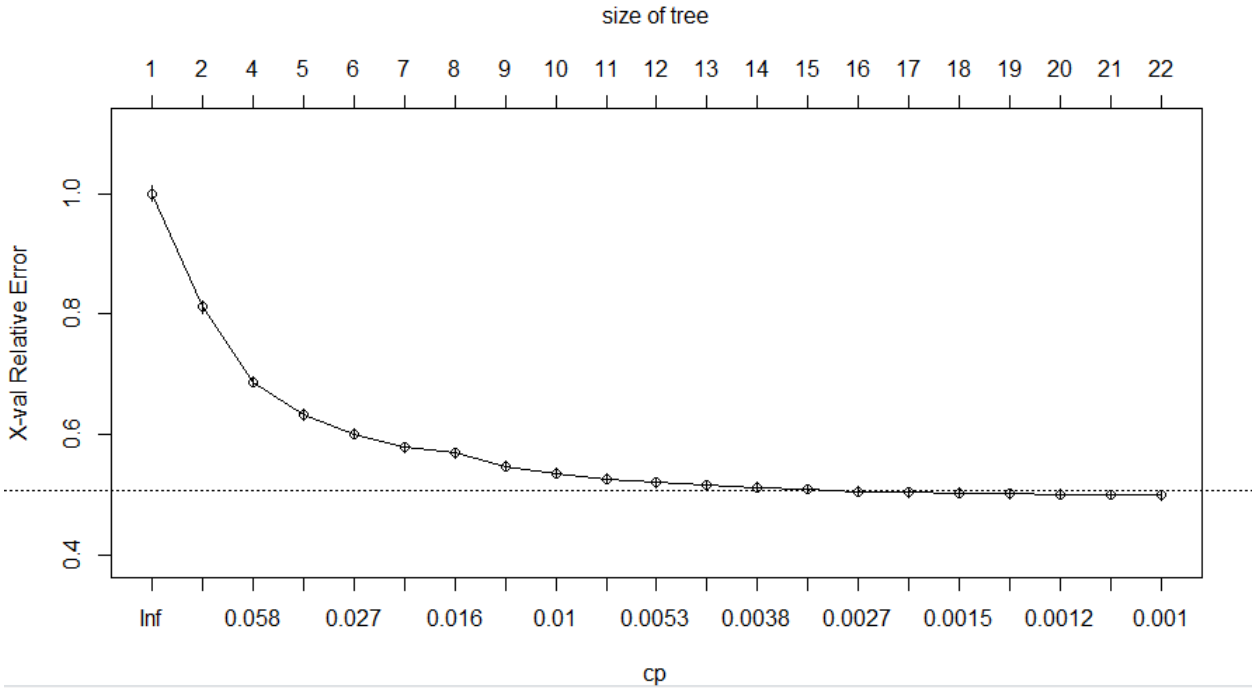
As shown by the fitted decision tree, *room_type* emerges as the primary determinant of average room price, with the first major splits separating room types 1, 2, 3, and all remaining categories. Conditional on *room_type*, *market_segment* assumes a critical secondary role, further differentiating the nightly rates paid by guests and exerting a substantial influence on overall pricing.

Since single Decision Trees tend to overfit when unconstrained, a key part of our modeling strategy was to control complexity through pruning. Overfitting occurs when the model learns noise or idiosyncratic patterns in the training set, leading to weakened performance on new observations. To mitigate this, pruning needs to be applied. Pruning addresses this by simplifying the tree and removing splits that contribute little to predictive accuracy. The goal is to balance model complexity and performance so that the tree generalizes well to unseen data.

The tree was pruned using a procedure parallel to the initial decision tree analysis. For each of the 10 iterations, after splitting the data into training and testing sets, cross-validation was performed on the training data to identify the optimal tree size. The tree was then pruned to an optimal size, achieving a compromise between parsimony and predictive power.

```r
# Perform 10 iterations with different train-test splits
for (i in 1:10) {
  set.seed(i * 42)  # Ensures reproducibility with a unique seed for each iteration

  # Randomly select 80% of the data for training
  train_index <- sample(1:nrow(seasonal_data), size = 0.8 * nrow(seasonal_data))
  train_data <- seasonal_data[train_index, ]
  test_data <- seasonal_data[-train_index, ]

  # Build the decision tree model with cross-validation for optimal cp
  seasonal_price_tree <- rpart(avg_room_price ~ month + room_type + market_segment,
                               data = train_data, method = "anova",
                               control = rpart.control(cp = 0.001))  # Start with a low cp

  # Cross-validation to find the optimal tree size
  cv_tree <- printcp(seasonal_price_tree)  # Displays cp table and cross-validation error

  # Plot the complexity parameter against the model's error
  plotcp(seasonal_price_tree)  # This plots the cross-validation results

  # Prune the tree to the optimal size based on cross-validation results
  best_cp <- seasonal_price_tree$cptable[which.min(seasonal_price_tree$cptable[,"xerror"]), "CP"]
  pruned_tree <- prune(seasonal_price_tree, cp = best_cp)

  # Adjust margins and plot the pruned decision tree
  par(xpd = NA, mar = c(4, 4, 4, 4))  # Adjust margins to prevent text cutoff
  plot(pruned_tree, main = paste("Pruned Decision Tree - Iteration", i),
       uniform = TRUE,  # Uniform vertical spacing
       margin = 0.2,    # Add extra space around the plot
       cex = 0.7)       # Adjust node text size

  # Add text labels to the tree
  text(pruned_tree, pretty = 0, cex = 0.7)  # Ensure all labels are visible

  # Predict avg_room_price on the test data and calculate MSE
  predictions <- predict(pruned_tree, newdata = test_data)
  MSE[i] <- mean((test_data$avg_room_price - predictions)^2)
}
```

Although the initial R implementation produced a relatively complex tree, the pruned version yielded more accurate and stable predictions. In the final tree, *room_type* remains the primary determinant of average room price, with additional splits on *market_segment* and *month*, highlighting their substantial influence on pricing. After pruning, the mean squared error (MSE) improved to 611.8217, underscoring the effectiveness of pruning in enhancing predictive performance. Beyond improving accuracy and reducing overfitting, the pruning process also clarified the most influential variables, particularly *room_type* and *market_segment*, as key drivers of hotel pricing patterns.



In conclusion, the Decision Tree analysis yielded substantive insight into the determinants of hotel room pricing. Pruning acted as an effective regularization step, balancing model complexity and predictive accuracy and reducing the risk of overfitting. *Room_type* consistently emerged as the dominant predictor, with *market_segment* and *month* exerting additional, meaningful influence on price variation. The refined tree thus provides an interpretable,

rule-based representation of pricing dynamics and serves as a useful tool for supporting data-driven pricing and revenue management decisions in the hotel context.

Conclusion

In this project, our objective was to analyze and predict hotel room prices using a set of booking-related predictors, including *month*, *lead_time*, *room_type*, and other contextual variables. To address this goal, we implemented two complementary modeling frameworks: Linear Regression and Decision Trees. Together, these approaches enabled us to examine both linear and non-linear relationships between the predictors and the response variable, *avg_room_price,* and to identify the key factors shaping hotel pricing behavior.

The Linear Regression model offered a transparent and easily interpretable baseline. It indicated that variables such as *lead_time*, *room_type*, and *special_requests* exert a statistically significant influence on average room price. With an R-squared of approximately 0.608, the model explained a substantial portion of the variability in *avg_room_price*. However, residual diagnostics revealed departures from linear model assumptions, including evidence of non-linearity and heteroscedasticity, suggesting that additional refinements could further improve model fit.

In contrast, the Decision Tree model highlighted the strengths of non-linear, rule-based methods. It captured more complex structures in the data, particularly interactions between *room_type* and *market_segment*, that were only partially represented in the linear model. The hierarchical structure of the tree handled categorical predictors naturally and exposed deeper patterns, such as threshold effects and conditional relationships. At the same time, the Decision Tree exhibited a greater tendency to overfit, which we addressed through pruning and cross-validation to balance model complexity and generalization performance.

Both models were evaluated on identical training and testing splits, using metrics such as Mean Squared Error (MSE) and R-squared for comparison. The Linear Regression model achieved a slightly lower prediction error, indicating marginally better predictive accuracy, whereas the Decision Tree remained superior in terms of interpretability and ease of communication to non-technical stakeholders.

Ultimately, the preferred model depends on the analytical objective. When the primary goal is to obtain clear, interpretable estimates of how individual factors affect hotel pricing, Linear Regression is highly suitable. When the focus shifts to uncovering complex, non-linear relationships, the Decision Tree provides a more flexible and expressive alternative. By juxtaposing these two approaches, we developed a more comprehensive understanding of hotel room pricing dynamics: Linear Regression as a strong, interpretable baseline and Decision Trees as a tool for deeper exploration of interaction effects. Future work could extend this analysis by

employing ensemble methods such as Random Forests or Gradient Boosting, which may deliver further gains in predictive accuracy and enhance the value of these models for pricing optimization in the hotel industry.

Bibliography

Nasef, A. W. (n.d.). Hotel Booking. Kaggle. Retrieved November 29, 2025, from https://www.kaggle.com/datasets/ahmedwaelnasef/hotel-booking

Source Code

Linear Regression

```
#------------ LOAD DATA & LIBRARIES ------------#
# Load libraries
library(corrplot)
library(boot)

# Load data
hotel_data <- read.csv("~/Downloads/Hotel.csv") # replace with your file name

#------------ FEATURE ENGINEERING ------------#
str(hotel_data) # Check structure
summary(hotel_data)
colSums(is.na(hotel_data))

# Remove ID (not a predictor)
hotel_data$ID <- NULL

# Drop rows with missing avg_room_price
hotel_data <- hotel_data[!is.na(hotel_data$avg_room_price), ]

# Convert categorical variables to factors
hotel_data$month <- as.factor(hotel_data$month)
hotel_data$room_type <- as.factor(hotel_data$room_type)
hotel_data$market_segment <- as.factor(hotel_data$market_segment)
hotel_data$meal_plan <- as.factor(hotel_data$meal_plan)
hotel_data$repeated_guest <- as.factor(hotel_data$repeated_guest)
hotel_data$status <- as.factor(hotel_data$status)

# Check final structure
str(hotel_data)

#------------ LINEAR REGRESSION MODEL ------------#
hotel_data.lm <- lm(avg_room_price~., data = hotel_data)
summary(hotel_data.lm)
```

```
# BIC
BIC(hotel_data.lm) # 327509.2


#------------ FEATURE SELECTION ------------#
#--- CORRELATION PLOT ---#
numeric_features <- hotel_data[sapply(hotel_data, is.numeric)]

# Compute correlation matrix
cor_matrix <- cor(numeric_features, use = "complete.obs")

# Plot correlation heatmap
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black",
         tl.cex = 0.8, addCoef.col = "black", number.cex = 0.6)

#--- STEPWISE FUNCTION ---#
step_backward <- step(hotel_data.lm, direction = "backward")

#------------ NEW LINEAR REGRESSION MODEL ------------#
#  New model with important features
reduced_model.lm <- lm(avg_room_price~. -date, data=hotel_data)
summary(reduced_model.lm)

# BIC
BIC(reduced_model.lm) # 327498.7

#------------ PLOTS ------------#
# create diagnostic plot
par(mfrow = c(2,2))
plot(reduced_model.lm)

#------------ CALCULATIONS ------------#
#--- SPLIT DATA ---#
MSE = rep(0,10)
for (i in 1:10){
  set.seed(i)

  train_index = sample(1:nrow(hotel_data),.8*nrow(hotel_data))
  train = hotel_data[train_index, -c(2,4)]
  test = hotel_data[-train_index, -c(2,4)]

  hotel_data.lm=lm(avg_room_price~., data=train)
  yhat=predict(hotel_data.lm, newdata=test)

  MSE[i]= mean((yhat-test$avg_room_price)^2)}

MSE
```

```
#--- MEAN TEST MSE ---#
mean_testMSE <- mean(MSE)
mean_testMSE # 492.9383

#--- RMSE ---#
rmse <- sqrt(mean_testMSE)
rmse # 22.20221

#--- CROSS VALIDATION ---#
set.seed(1)
K <- 10
folds <- sample(rep(1:K, length.out = nrow(hotel_data)))

cv_mse <- rep(0, K)

for (k in 1:K) {
  train_data <- hotel_data[folds != k, ]
  test_data  <- hotel_data[folds == k, ]

  model_k <- lm(avg_room_price~. - date, data = train_data)

  preds <- predict(model_k, newdata = test_data)

  cv_mse[k] <- mean((preds - test_data$avg_room_price)^2)
}

mean(cv_mse) # 484.3507
sqrt(mean(cv_mse)) # 22.00797
```

Decision Tree

```
#------------ LOAD DATA & LIBRARIES ------------#
library(rpart)
library(dplyr)
library(rpart.plot)

# Load data
hotel_data <- read.csv("~/Downloads/Hotel.csv") # replace with your file name

# Convert relevant columns to factors
hotel_data$month          <- as.factor(hotel_data$month)
hotel_data$room_type      <- as.factor(hotel_data$room_type)
hotel_data$market_segment <- as.factor(hotel_data$market_segment)

# Select relevant columns for the decision tree models
seasonal_data <- hotel_data %>%
  select(avg_room_price, month, room_type, market_segment)
```

```
#------------ SINGLE DECISION TREE (UNPRUNED) ------------#

MSE_single <- rep(0, 10)  # store MSE for each iteration

for (i in 1:10) {
  set.seed(i * 42)  # Ensures reproducibility with a unique seed for each
iteration

  # Randomly select 80% of the data for training
  train_index <- sample(1:nrow(seasonal_data), size = 0.8 *
nrow(seasonal_data))
  train_data  <- seasonal_data[train_index, ]
  test_data   <- seasonal_data[-train_index, ]

  # Build the decision tree model
  seasonal_price_tree <- rpart(
    avg_room_price ~ month + room_type + market_segment,
    data   = train_data,
    method = "anova"
  )

  # Visualize the decision tree
  plot(seasonal_price_tree,
       main = paste("Decision Tree - Iteration", i),
       cex  = 0.7)
  text(seasonal_price_tree, pretty = 0, cex = 0.7)

  # Predict avg_room_price on the test data and calculate MSE
  predictions <- predict(seasonal_price_tree, newdata = test_data)
  MSE_single[i] <- mean((test_data$avg_room_price - predictions)^2)
}

# Average MSE across all iterations (single tree)
mean_MSE_single <- mean(MSE_single)
print(mean_MSE_single)  # 658.6132


#------------ PRUNED DECISION TREE (WITH CROSS-VALIDATION) ------------#

MSE_pruned <- rep(0, 10)  # store MSE for each iteration

for (i in 1:10) {
  set.seed(i * 42)  # Ensures reproducibility with a unique seed for each
iteration

  # Randomly select 80% of the data for training
  train_index <- sample(1:nrow(seasonal_data), size = 0.8 *
nrow(seasonal_data))
  train_data  <- seasonal_data[train_index, ]
```

```
  test_data   <- seasonal_data[-train_index, ]

  # Build the decision tree model with cross-validation for optimal cp
  seasonal_price_tree <- rpart(
    avg_room_price ~ month + room_type + market_segment,
    data    = train_data,
    method  = "anova",
    control = rpart.control(cp = 0.001)  # Start with a low cp
  )

  # Cross-validation to find the optimal tree size
  cv_tree <- printcp(seasonal_price_tree)  # Displays cp table and
cross-validation error

  # Plot the complexity parameter against the model's error
  plotcp(seasonal_price_tree)  # This plots the cross-validation results

  # Prune the tree to the optimal size based on cross-validation results
  best_cp <- seasonal_price_tree$cptable[
    which.min(seasonal_price_tree$cptable[, "xerror"]),
    "CP"
  ]
  pruned_tree <- prune(seasonal_price_tree, cp = best_cp)

  # Adjust margins and plot the pruned decision tree
  par(xpd = NA, mar = c(4, 4, 4, 4))  # Adjust margins to prevent text cutoff
  plot(pruned_tree,
       main   = paste("Pruned Decision Tree - Iteration", i),
       uniform = TRUE,   # Uniform vertical spacing
       margin  = 0.2,    # Add extra space around the plot
       cex     = 0.7)    # Adjust node text size

  # Add text labels to the tree
  text(pruned_tree, pretty = 0, cex = 0.7)  # Ensure all labels are visible

  # Predict avg_room_price on the test data and calculate MSE
  predictions_pruned <- predict(pruned_tree, newdata = test_data)
  MSE_pruned[i] <- mean((test_data$avg_room_price - predictions_pruned)^2)
}

# Average MSE across all iterations (pruned tree)
mean_MSE_pruned <- mean(MSE_pruned)
print(mean_MSE_pruned) # 611.8217
```