Prediction Challenge Pt.1

—

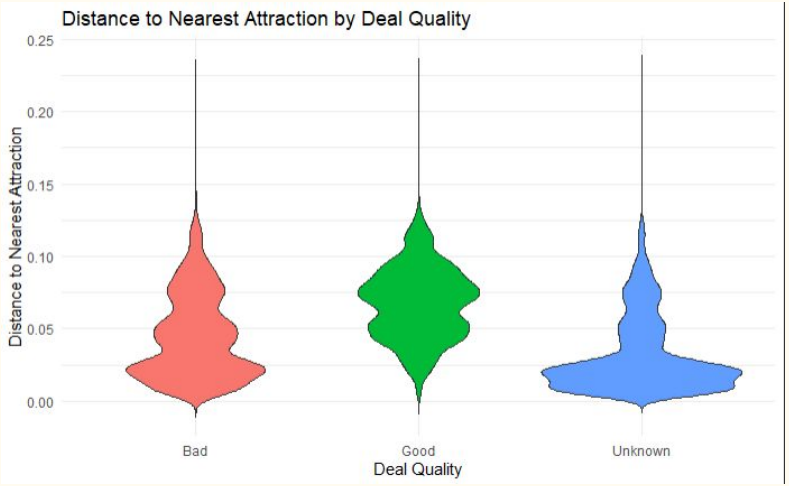
By Romil Patel

Introduction

- Introduction of the prediction challenge using the 'predictionChallenge.csv' file.
- Explanation of the challenge: to predict 'Deal Quality' as 'Good', 'Bad', or 'Unknown' using complex patterns and derived attributes.
- Highlighting the requirement to construct derived attributes that provide deeper insights into the data.
- Introduction of the 'Attractions.csv' file, which is key to unlocking the embedded pattern in the data.
- Emphasis on the need for precision and the possibility of achieving 100% accuracy without noise in the data.
- Brief overview of the two deliverables: a text file detailing the discovered pattern and the accompanying code.

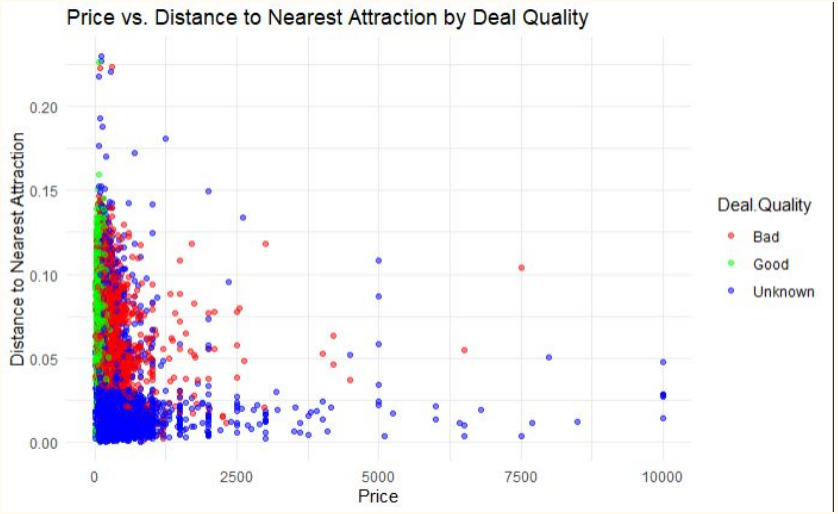
Analyzing Proximity to Attractions by Deal Quality

- Presentation of a violin plot comparing the distance to the nearest attraction across Airbnb listings with varying deal qualities.
- Distinct shapes of the distribution for 'Bad', 'Good', and 'Unknown' deal qualities, with 'Good' deals typically being closer to attractions.
- The plot suggests a potential pattern where proximity to attractions could influence the perceived deal quality of an Airbnb listing.
- The visual underscores the importance of location in the hospitality industry, particularly in the context of tourist attractions.



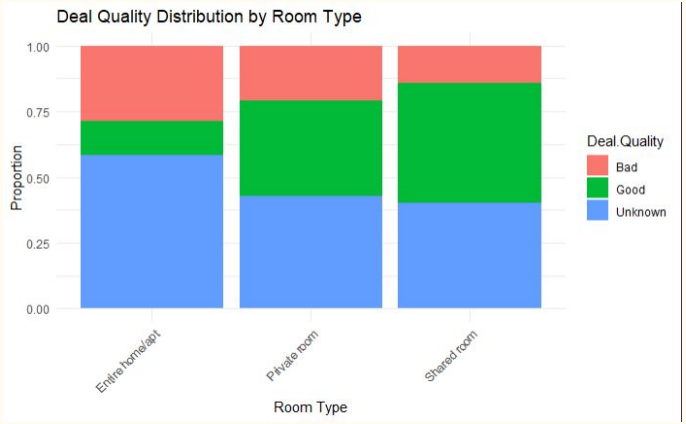
Price versus Proximity to Attractions by Deal Quality

- The scatter plot shows the relationship between the price of Airbnb listings and their distance to the nearest attraction, differentiated by deal quality.
- There is a visible concentration of 'Good' deals at lower prices and shorter distances to attractions, while 'Bad' and 'Unknown' deals are more dispersed.
- The plot illustrates that there may be a sweet spot in pricing and location that yields a 'Good' deal classification.



Deal Quality Segmentation Across Different Room Types

- This stacked bar chart displays the proportions of 'Good', 'Bad', and 'Unknown' deal qualities across various room types.
- It highlights that the distribution of deal quality is relatively consistent across different room types, suggesting room type alone may not be a strong predictor of deal quality.
- The visualization emphasizes the need to consider additional factors beyond room type when assessing the potential of a listing.



Model Comparisons - Logistic Regression vs. Random Forest

- Introduction of two predictive models: Multinomial Logistic Regression and Random Forest, to classify deal quality.
- Logistic Regression model details, including the use of variables such as number of reviews, price, and distance to attractions.
- Performance metrics of the Logistic Regression model showing an accuracy of 71.63%, with a kappa statistic of 0.5424 indicating moderate agreement.
- Overview of the Random Forest model, which considered the same predictors after cleaning the data and imputing missing values.
- The Random Forest model out-of-bag (OOB) error rate reported at 8.72%, suggesting a higher predictive accuracy compared to Logistic Regression.
- The Random Forest confusion matrix details showing strong classification performance across all deal quality categories.

Data Preprocessing & Predictive Accuracy

```r
# Similar preprocessing for test_data as done for train_data before model training
test_data$reviews_per_month[is.na(test_data$reviews_per_month)] <- mean(train_data$reviews_per_month, na.rm = TRUE) # Use train_data mean for consistency
test_data$neighbourhood.group[is.na(test_data$neighbourhood.group)] <- names(which.max(table(train_data$neighbourhood.group))) # Use most common category from train_data

# Ensure factor levels in test_data match those in train_data for categorical predictors
test_data$neighbourhood.group <- factor(test_data$neighbourhood.group, levels = levels(train_data$neighbourhood.group))
test_data$room.and.type <- factor(test_data$room.and.type, levels = levels(train_data$room.and.type))

# Generate predictions for test_data using the trained random forest model
predictions_test <- predict(rf_model, newdata = test_data)

# Calculate accuracy
correct_predictions_test <- sum(predictions_test == test_data$Deal.Quality)
total_predictions_test <- length(predictions_test)
accuracy_test <- correct_predictions_test / total_predictions_test

# Print the accuracy
print(paste("Test Accuracy:", accuracy_test))
```

```
[1] "Test Accuracy: 0.909072312570318"
```

Our goal was to create a model with the best accuracy possible, aiming for 100%. While we set out with the ambitious aim of achieving perfect accuracy in our model, the journey there brought us to a noteworthy milestone of 91% accuracy. Though it falls short of the ideal


ThaNK
YOU