**Airbnb Listing Success Prediction**

**GitHub Url:** <https://github.com/G-Narendra/AirBnb-Course-work>

## Q1. Domain Knowledge Building

To solve this problem, first I studied Airbnb business, then I got to know that, it is a two-sided market. On one side, **hosts** want to make money and keep their rooms full. On the other side, **guests** want the best place to stay, which feels like it’s worth the price. When I explored the dataset, I found a column called **host\_is\_superhost** but initially didn't understand its full value. Later I did research about **Super host** criteria and ranking algorithms of Airbnb. After all this research I decided choose business perspective as it balances both **financial goals** and **quality standards**.

## Q2. Data Sourcing

No. The provided dataset is rich enough. External data adds noise, so I chose to dig deep into existing features like reviews to gain valuable insights without using outside files

## Q3. Dependent Variable Formulation

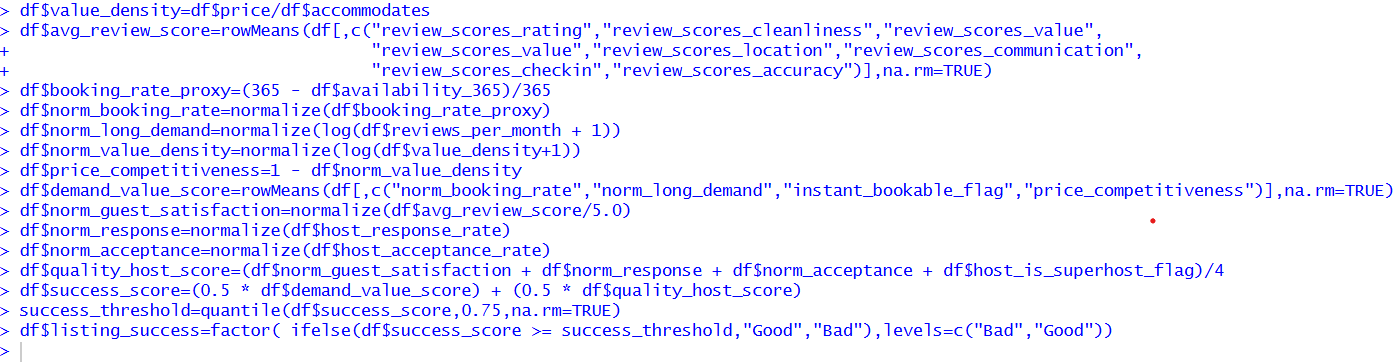
**(a) Describe your formula for defining Good/Bad listing**

From a business perspective, I defined listing as **Good** if that satisfies two core goals: **Profitability** and **Reputation**. Looking only at price may show costly listings with no bookings, while focusing only on reviews may highlight cheap listings with little profit. To solve this, I built a balanced **Success Score (Sᵢ)** that combines both goals.

1. **The Revenue Goal (Demand & Value Score):** I measured revenue potential by calculating “Value Density” (price per person) and Occupancy Rates. This tells me if a listing is competitively priced and actually getting booked.
2. **The Reputation Goal (Quality & Host Score):** I measured long-term health by averaging review scores and checking Superhost status, because guest satisfaction is what drives future growth.

I gave both goals equal weight (50/50). Finally, only the top 25% of scores are labeled “Good,” identifying true Market Leaders, not average performers.

**(b) Show the R formula or Excel function used**

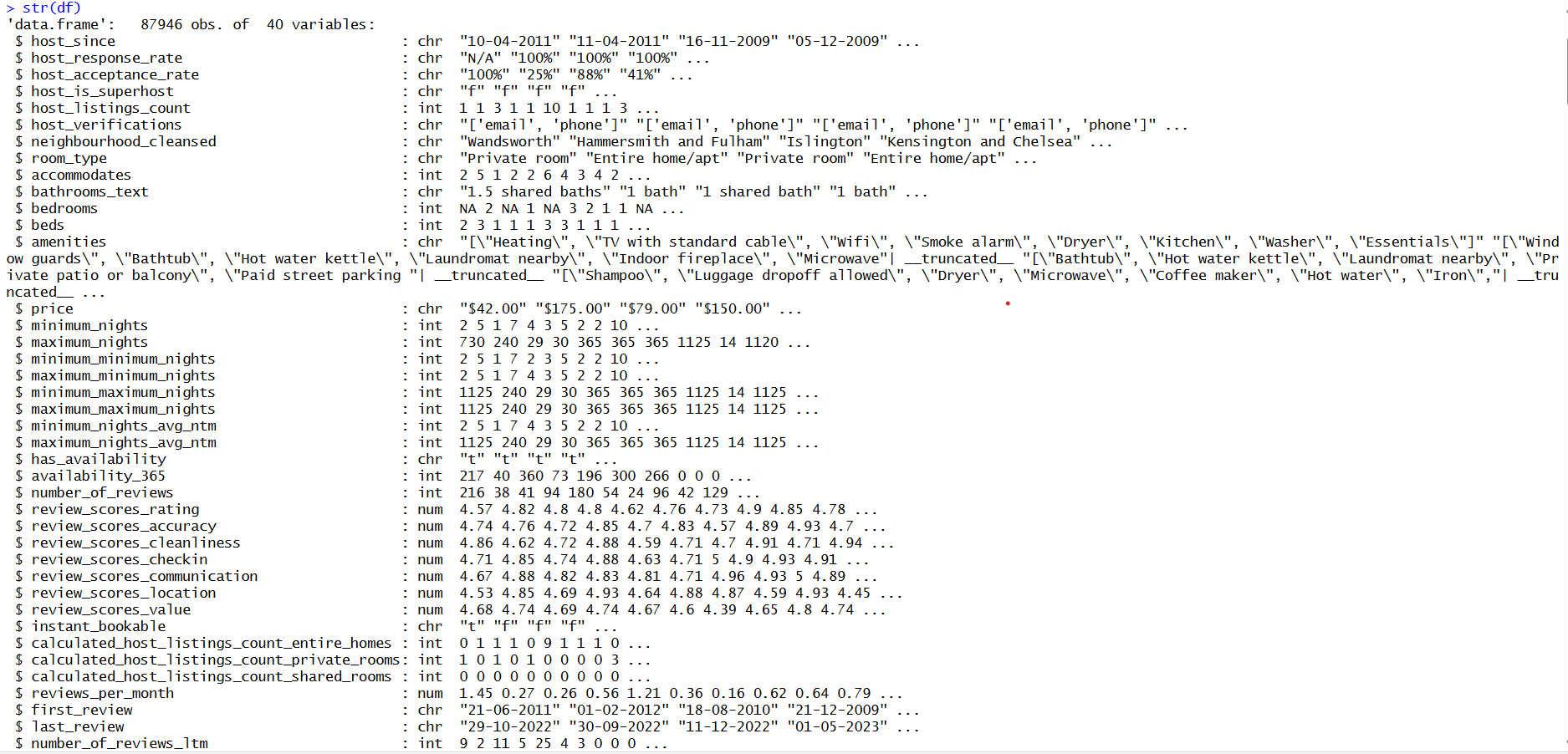


**Fig 1** Listing Success Formulation in R

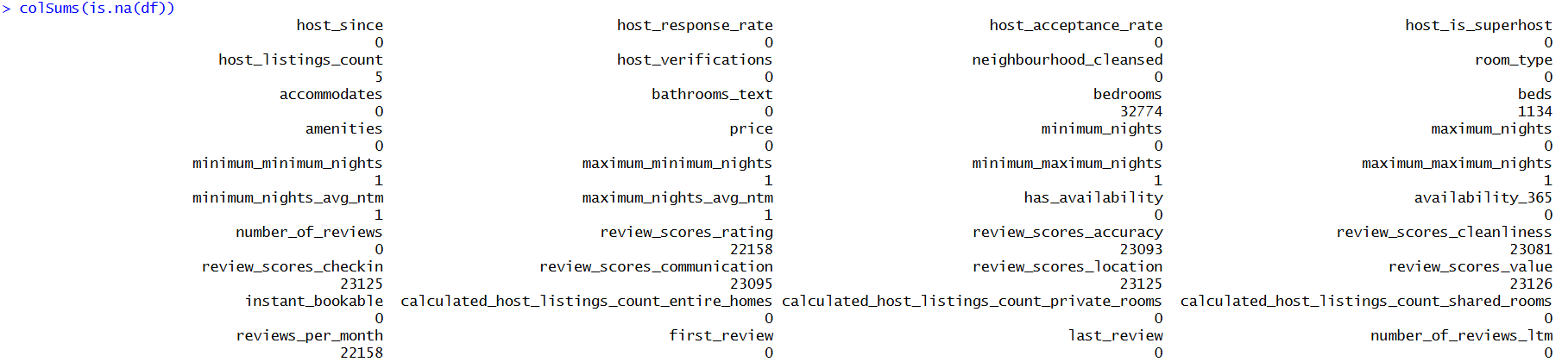
## Q4. Data Pre-processing

I am gonna say, this is the place where I focused the most, because poor quality data certainly leads to poor solutions.

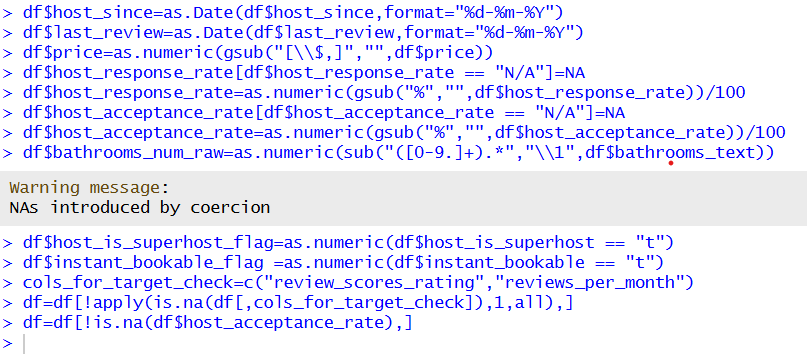
1. **Excel Clean-up:** I started by manually dropping 35 unwanted columns in Excel. After that,I loaded and checked the structure and counted null values for each column **(Fig 2, Fig 3).**
2. **Fixing Data Types:** I observed many numeric columns were stored as text, you can check the (**Fig 2)**, so I converted them into numeric type. I converted date columns into proper Date formats **(Fig 4).**
3. **Format Fixing:** columns likeprice and host\_responce\_rate and host\_response\_rate are not in correct format**(Fig 3)**.I stripped the dollar signs (**$**) from the price column and removed “**%**” from rate columns to make them simple numbers **(Fig 4).**and I found host\_response\_rate and host\_acceptance\_rate have NA values in this format “**N/A**”.I replaced with proper **NA** values **(Fig 4).**
4. **Regex Extraction:** I decided to use Regex to extract only the numeric value (1.5) and ignore the text from bathroom\_test .and later converted it into numeric type too **(Fig 4).**
5. **Dropping High-NA Rows:** I made a hard decision, over 15,000 NA values existed in host\_acceptance\_rate, host\_response\_rate, and review columns**.** Tried model imputation but every model got accuracy under 40% accuracy. As Success Score relies heavily on these metrics, I dropped those rows **(Fig 4).**
6. **Outlier Capping:** Before NA value imputation, I Replaced outliers with bounds to avoid data loss Using IQR **(Fig 5).**
7. **Standard Imputation:** Columns with less NA values are imputed using Appropriate Statistical function based on data distribution **(Fig 6).**
8. **Model Imputation:** I did model imputation for bedrooms. Linear and Logistic Regression gave ~70% accuracy, but I required ~90%. After testing other models, XGBoost achieved **90% accuracy** (R²) with **lowest MSE 0.056**. I used it to predict NA values **(Fig 7).**
9. **Amenity & Host Verification Engineering:** Amenities column parsed into binary flags. Host\_verifications parsed into verification\_count features **(Fig 8).**
10. **Leakage Columns Dropping**: After deriving the target column, I dropped the all the columns which I used and all the columns which I newly derived to derive Target to prevent the data leakage to the model. This is the code snippet of outlier capping **(Fig 8).**



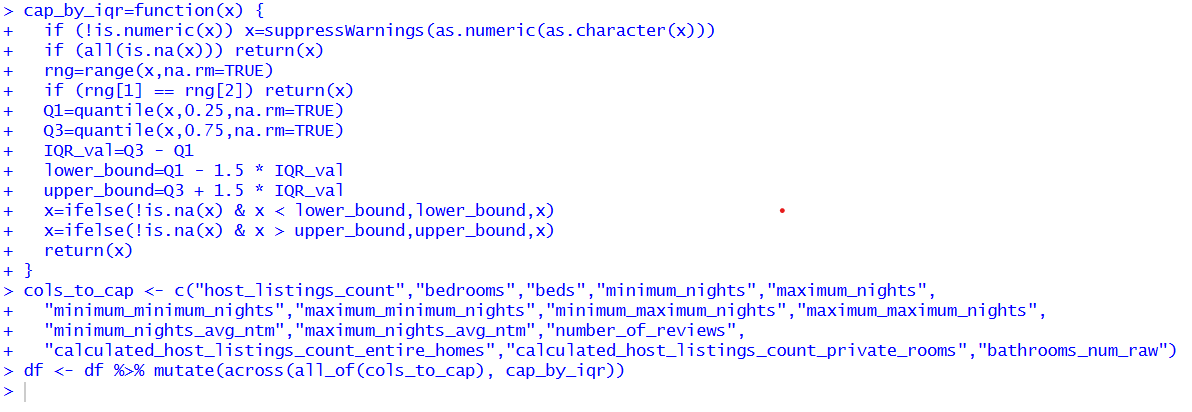
**Fig 2** Initial Structure of Dataset



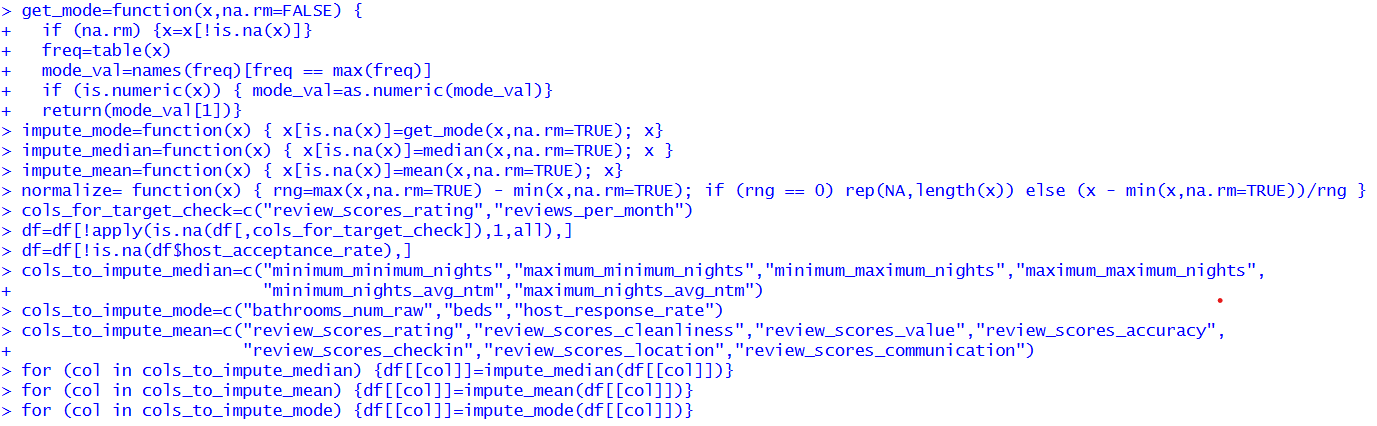
**Fig 3** Initial Null Value Count



**Fig 4** Type Conversion, Format Correction, NA value Drop



**Fig 5** Outlier Capping



**Fig 6** Less NA Values columns imputation



**Fig 7** Bedrooms NA value Imputation Using XGBoost Model



**Fig 8** Preprocessing After Formulation & Dropping Leakage Columns

## Q5. Hypotheses Formulation

I developed five success hypotheses based on research, Airbnb hospitality knowledge, and common logic about hosts and guests.

**Hypothesis 1: Social Proof with reviews**

* **Hypothesis Statement:** I believe listings with a high amount of reviews will have a higher significant probability of being “Good” because they provide social proof and trust to newly coming guests.
* **How I arrived at it:** I have used some **common logic and real-world scenario** from a customer perspective.
* **Four-Dimensional Reasoning:**

| **Variable** | **Hypothesis** | **Data Availability** | **Data Quality** |
| --- | --- | --- | --- |
| **number\_of\_reviews** | **Positive Correlation** | Available directly in the provided dataset. | **High** |

**Hypothesis 2: Latest review Effect**

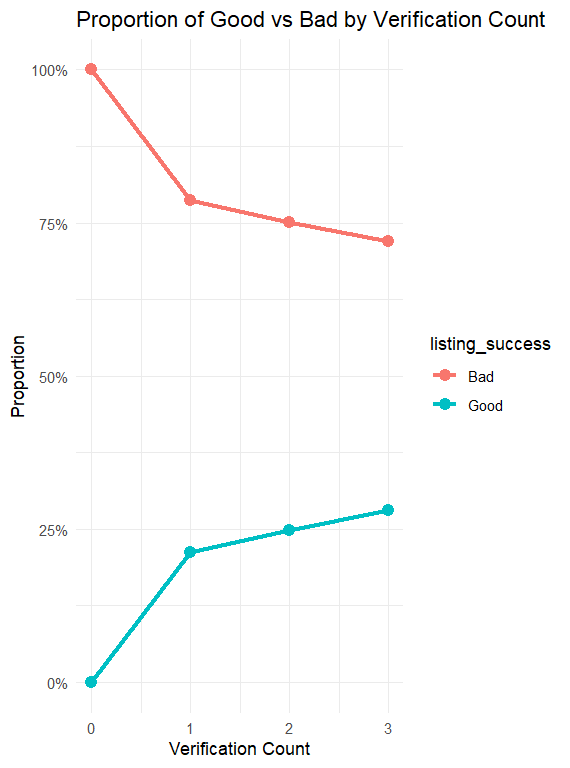
* **Hypothesis Statement:** Listings that have been reviewed recently (fewer days since last review) will beat hard listings (so many days since last review), as I know the Airbnb algorithm prioritizes active properties.
* **How I arrived at it:** I read online about **Airbnb Search Engine Optimization**. They will list the properties have recent reviews or active properties.
* **Four-Dimensional Reasoning:**

| **Variable** | **Hypothesis** | **Data Availability** | **Data Quality** |
| --- | --- | --- | --- |
| **review\_recency\_days** | **Negative Correlation** | Derived by subtracting last\_review date from the reference date ($07-10-2023$, the latest date in the dataset). | **Medium**  (Nearly 22,000 empty values). |

**Hypothesis 3: Host Trustworthiness**

* **Hypothesis Statement:** I thought hosts who verify their identity in multiple ways (email, phone, ID) are trustworthy, and that indicates safety for the customers.so I predict a higher verification count will have higher success.
* **How I arrived at it:** I have gone through **plot**, which is comparing the Listing Success with verification count, and I found verification count is directly proportional Listing Success **(Fig 9).**
* **Four-Dimensional Reasoning:**

| **Variable** | **Hypothesis** | **Data Availability** | **Data Quality** |
| --- | --- | --- | --- |
| **verification\_count** | **Positive Correlation** | Derived by counting items in the host\_verifications column. | **High** |



**Fig 9** Verification Count VS Listing Success

**Hypothesis 4: The Amenities**

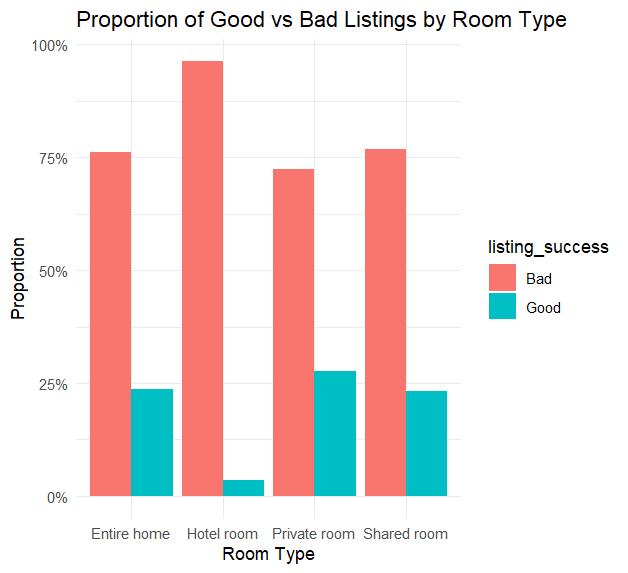
* **Hypothesis Statement:** As we all know the presence of basic amenities, specifically Wi-Fi, AC is mandatory. I predicted that it’s absence will predict Bad listing.
* **How I arrived at it:** I arrived with hypothesis using **common logic**. Now a days no one books a room without internet. if I were the customer, for sure I will prefer to book a room with Wi-Fi.
* **Four-Dimensional Reasoning:**

| **Variable** | **Hypothesis** | **Data Availability** | **Data Quality** |
| --- | --- | --- | --- |
| **has\_wifi** | **Strong Predictive Flag** | Extracted from the amenities column. | **High** |

**Hypothesis 5: Room Type Preference**

* **Hypothesis Statement:** If the room type is Hotel Room, then it will negatively correlate with good listing, as users prefer “authentic” home experiences.
* **How I arrived at it:** I have gone through **plot**, which is comparing the Listing Success with Room Type, and I found among all the four types customers the hotel category had 3% Good and 97% Bad **(Fig 10).**
* **Four-Dimensional Reasoning:**

| **Variable** | **Hypothesis** | **Data Availability** | **Data Quality** |
| --- | --- | --- | --- |
| **room\_type** | **Negative Flag** | Available directly in the dataset. | **High** |



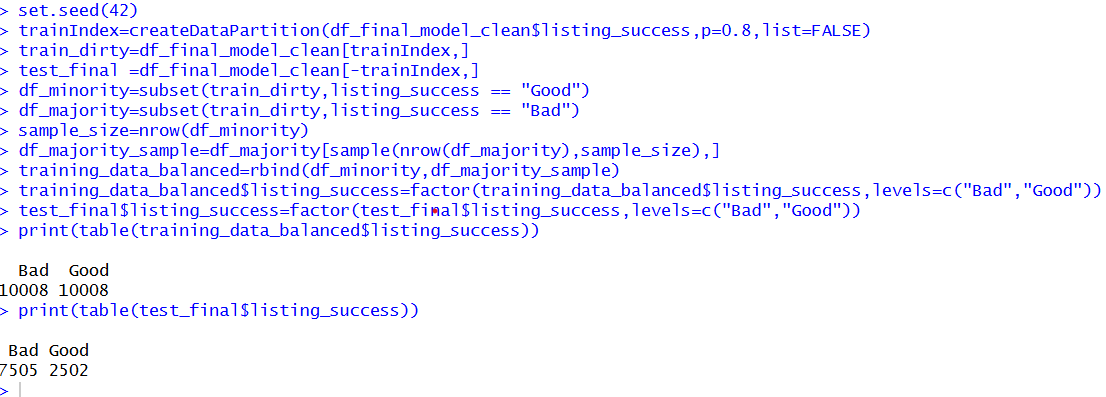
**Fig 10** Room Type VS Listing Success

## Q6. Model Building Process

As they mentioned in the problem statement, I have trained two different models for predictions, **Logistic Regression** and a **Decision Tree**.

**1. Training & Testing Strategy:**

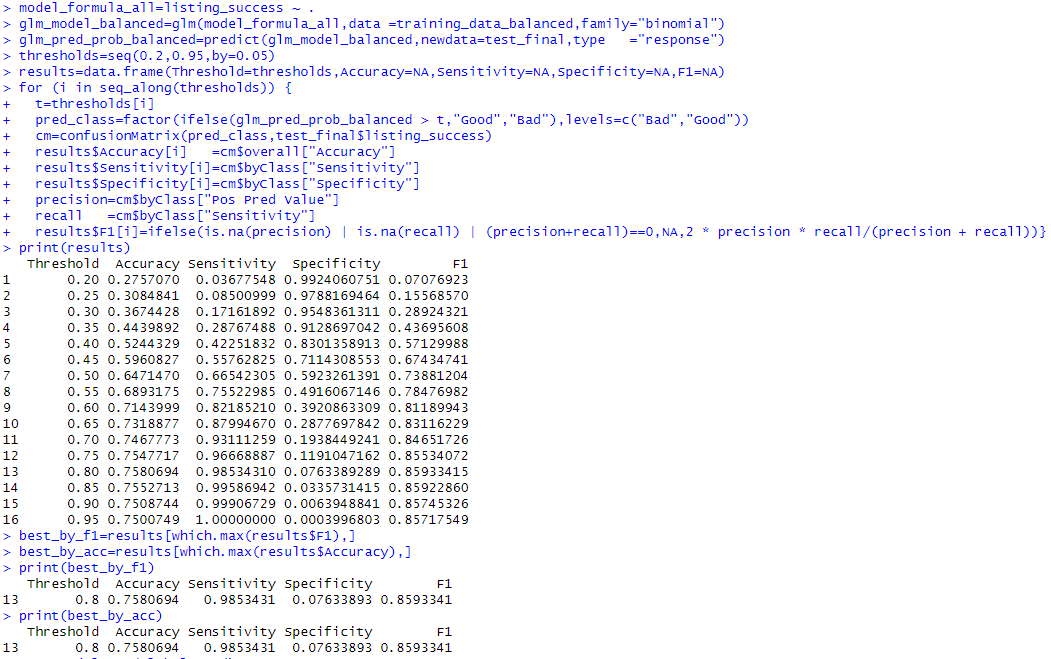
Following the Golden Rule of data science, I never touched the test set. The clean data was split into **80%** training and **20%** testing. Since the dataset was **imbalanced** (75% Bad, 25% Good), I applied **undersampling** only to the training set. The test set remained imbalanced (7505 Bad / 2502 Good) to ensure honest evaluation against **real‑world conditions** **(Fig 11).**



**Fig 11** Train,Test Spliting & Balancing Training Data

**2. Tuning & Threshold Optimization (Logistic Regression**):

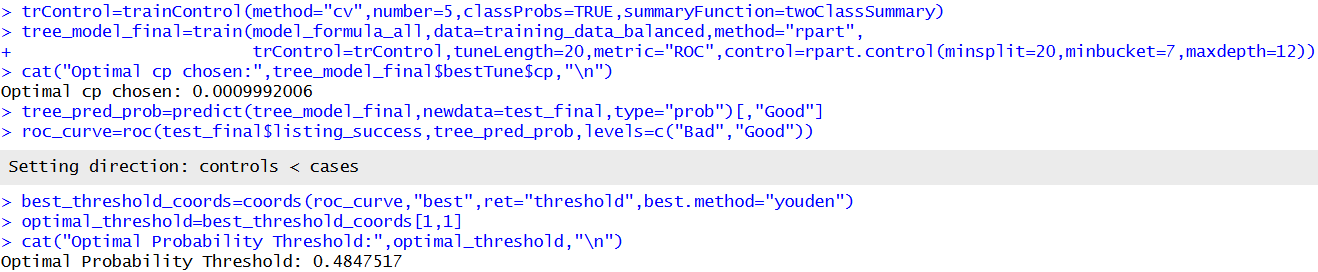
I tested **16 thresholds** (0.20–0.95). At 0.80, Accuracy reached 75.9% and F1 Score 0.86, but **Specificity fell to 7.6%,** making the model aggressive in predicting “Bad.” I also analyzed **0.50 to better balance** False Positives and False Negatives **(Fig 12).**



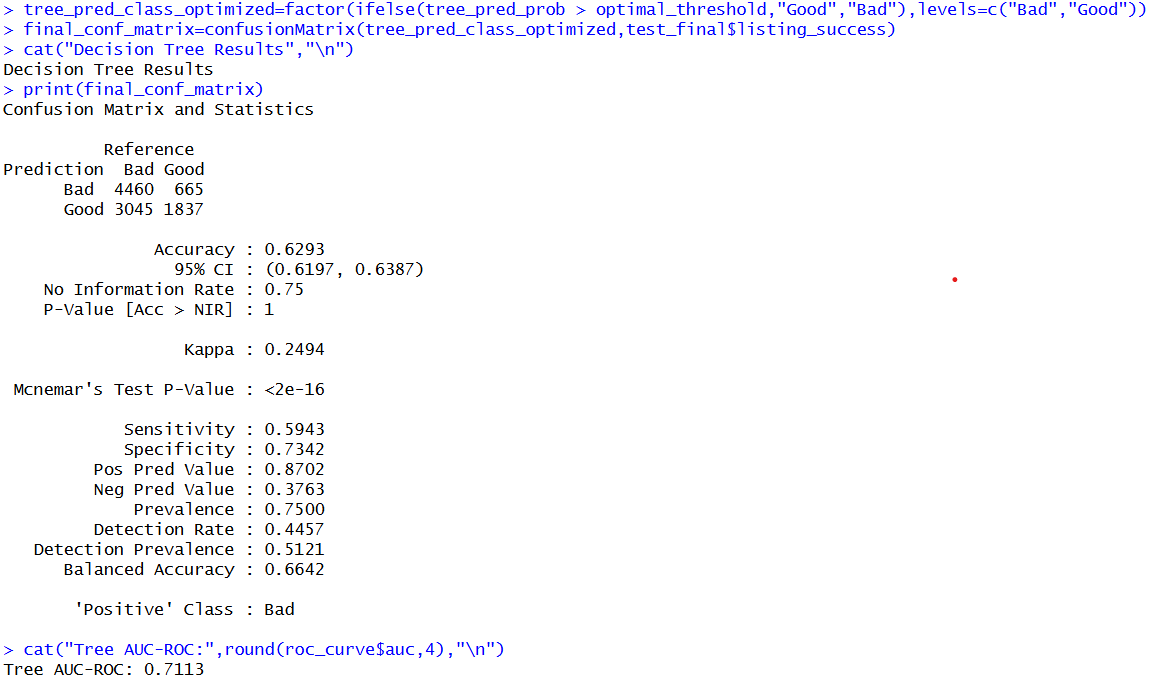
**Fig 12** Logistic Regression with Tuning

**3. Decision Tree Tuning (Cross-Validation):**

I used the caret package with **5‑Fold Cross‑Validation** to stabilize the Decision Tree. To reduce overfitting, I set maxdepth = 12 and minbucket = 7. Expanding the **tuning grid to 20 values**, the model selected an optimal **cp of 0.000999 & optimal threshold of 0.4848(Fig 13).** capturing subtle patterns without uncontrolled growth. Using **Youden’s Index**, I optimized the threshold to **0.4847**, balancing Sensitivity and Specificity. Final performance achieved an **AUC of 0.7113** and Balanced Accuracy of **65.22%,** outperforming un‑tuned models **(Fig 14).**



**Fig 13** Decision Tree with Tuning



**Fig 14** Decision Tree Results

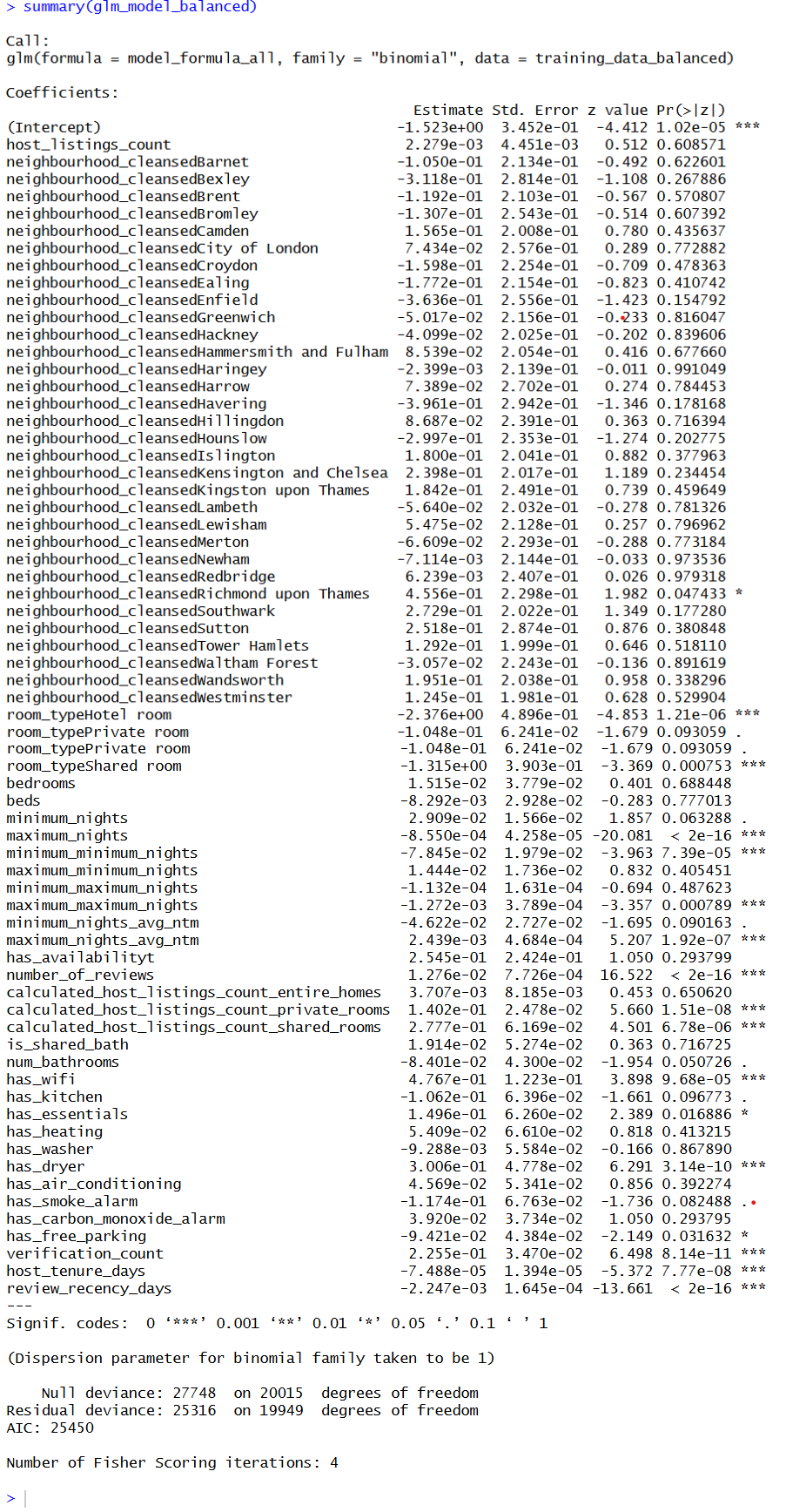
**4. Model Comparison & Selection:**

Logistic Regression showed high overall accuracy but had trouble handling the imbalance between classes. It was mostly biased toward the majority class “Bad” **(Fig 12)**. On the other hand, the Decision Tree, after tuning with Youden’s Index, gave a much better balance between Sensitivity and Specificity, with a Balanced Accuracy of 65.22% **(Fig 14).** Since my main goal is to correctly identify “Good” listings without missing too many, the Decision Tree worked better. It was able to capture complex patterns better than Logistic regerssion.

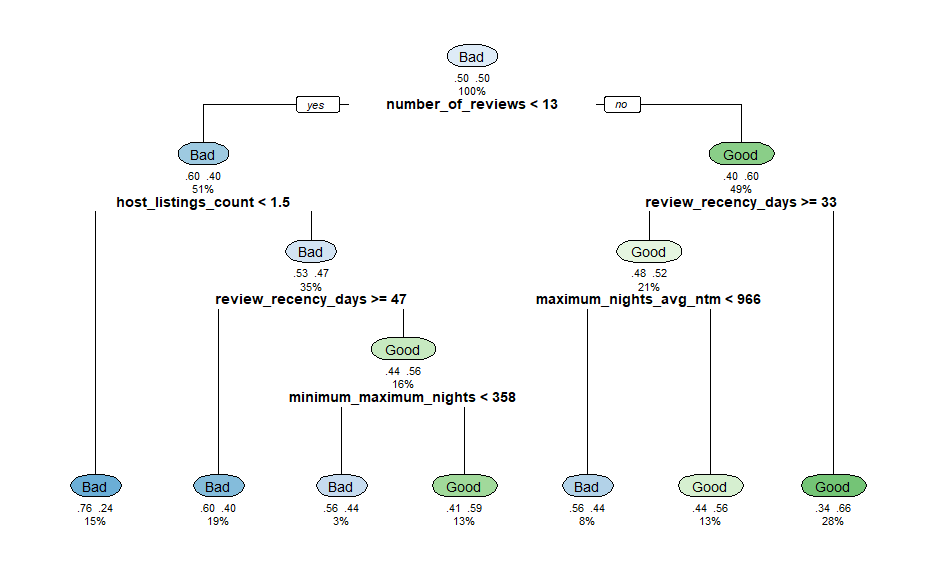
## Q7. Model Interpretation (Business and Applied Perspective)

From what I learnt from my previous classes, I decided to not to just look at the accuracy, I decided to understand the real business rules by looking at the **P-values** in my Logistic Regression and the **Nodes** in my Decision Tree.

1. **Review Volume is King:** My Decision Tree logic consistently prioritizes number\_of\_reviews **(Fig 16).** If a listing has very high reviews, the probability of being Good increases significantly. Hosts must focus on getting those first 10-15 reviews quickly.
2. **Recency Matters:** Thereview\_recency\_days was a key splitter. Listings without recent reviews dropped in ranking, showing the importance of keeping calendars active**(Fig 16).**
3. **Trust is Currency:** TheVerification\_count was strongly positive Significance **(Fig 15).** Guests trusts hosts who verify email, phone, and ID. And not the Anonymous hosts.
4. **Wi-Fi & Dryers are Mandatory:** Positive coefficients for has\_wifi and has\_dryer highlight that these amenities are essential, not optional **(Fig 15).**
5. **Hotel Rooms Fail:** The coefficient for room\_typeHotel room was strongly negative Significance (‑2.37) **(Fig 15).**  Guests prefer authentic homes over hotel inventory.
6. **Location Hidden Gem:** Surprisingly, outer regions like Richmond upon Thames showed strong signals of success **(Fig 15)**, suggesting these are underestimated markets where it is easier to be successful.
7. **Professional Hosts Win:** Thecalculated\_host\_listings\_count was positive, showing multi‑property “Pro” hosts beat single‑listing hosts **(Fig 15) (Fig 16)**.
8. **Minimum Nights Strategy:** Theminimum\_minimum\_nights were significant, it suggests that setting minimum night’s stay rules attracts better guests and stable income. **(Fig 16) (Fig 15)**.
9. **Shared Rooms are Dead:** Strongly negative coefficients for shared rooms showing guest are not interested in shared properties **(Fig 15)**.
10. **Host Experience and Trust:** Interestingly,Longer **host tenure** (p < 0.01) **negatively** predicts “Good” listings, suggesting experienced hosts become **satisfied**. Continuous updates and high effort are key **(Fig 15)**.



**Fig 15** Logistic Regression Summary



**Fig 16** Decision Tree Plot

## Q8. Critical Conclusions & Automation

**Critical Analysis:** I need to be honest, my model is useful but not fully perfect. My Decision Tree AUC is **0.7113 (Fig 14**), showing improvement is needed. Logistic Regression suffers from **Accuracy-Specificity trade‑off**, maximizing Accuracy (75.9%) dropped Specificity to 7% due to imbalance **(Fig 11).**

**Improvements:** Beyond Logistic Regression and Decision Tree, I’d like to **explore Random Forest or XGBoost** for complex data. I ignored property **neighbourhood descriptions**, but planning to apply **NLP to extract keywords**. Also collecting guest reviews will help sentiment analysis. Finally, **CNN could evaluate listing photos**, as we know strong visuals significantly affect booking decisions.

**Productization:** Finally, I wanna turn this R code into a real tool. I am thinking to build an automated dashboard called **The Airbnb Doctor**, A host could log in, enter their price and amenities etc.., and my model would run in the background. Based on my Decision Tree rules, it wouldn't just say “Good/Bad” It would give specific advice, like “You need 3 more verifications” or “Your price is too high for your review count”.