Airbnb Price Prediction Project (Kaggle Dataset)

Pragathi Sharma

2025-12-02

Executive Summary

This report presents a predictive modeling project using the New York City Airbnb Open Data from Kaggle. The main goal is to estimate nightly listing prices based on observable features such as neighbourhood group, room type, minimum nights, number of reviews, and availability.

The workflow is structured as follows:

- Data Source & Preparation: NYC Airbnb Open Data, filtered to remove missing values and extreme prices.
- Methods: Linear Regression, Random Forest, and Gradient Boosting (GBM) models are trained on an 80/20 train–test split.
- Evaluation Metric: Root Mean Square Error (RMSE) on the held-out test set.

Using results from an execution of this analysis:

- Linear Regression: RMSE 85.10
- Gradient Boosting (GBM): RMSE 81.28
- Random Forest: RMSE 78.41 (best performance)

The Random Forest model provides the lowest RMSE, suggesting that nonlinear relationships and interactions among features are important for predicting Airbnb prices. The report concludes with a discussion of limitations and suggestions for future work, including feature engineering (e.g., amenities encoding), geographic detail, and hyperparameter tuning.

1. Introduction

Airbnb has transformed the short-term rental market, offering flexible lodging options for travelers while allowing hosts to monetize spare rooms or properties. However, listing prices vary widely and depend on several factors, such as:

- Location (neighbourhood group in NYC),
- Room type (entire home, private room, etc.),
- Minimum nights required,
- Host activity (number of reviews, listings, availability). Accurate price prediction can be useful for:
- Hosts, to price competitively,
- Platforms, to support pricing tools,
- Researchers, to understand drivers of price.

The goal of this project is to build and evaluate models that predict Airbnb listing prices using the NYC Airbnb Open Data:

1. Define the problem and data.
2. Prepare and explore the data.
3. Develop and evaluate predictive models.
4. Interpret results and discuss limitations.

2. Methods and Analysis

2.1 Data Source

The dataset used in this project is the New York City Airbnb Open Data available on Kaggle:

- Kaggle dataset: dgomonov/new-york-city-airbnb-open-data
- Contains information on NYC Airbnb listings, including price and several host and listing features.

The dataset can be downloaded as follows:

kaggle datasets download -d dgomonov/new-york-city-airbnb-open-data unzip new-york-city-airbnb-open-data.zip

2.2 Software and Packages

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(randomForest)
library(gbm)
```

2.3 Data Loading and Preparation

We load the data from the Kaggle CSV, then perform basic cleaning:

- Keep only variables relevant to price prediction.
- Remove rows with missing values.
- Filter out listings with non-positive prices or extreme high prices ($> \$1000$).
- Convert categorical variables to factors.

```
listings <- read_csv("AB_NYC_2019.csv")

listings <- listings %>%
select(
price,
neighbourhood_group,
room_type,
minimum_nights,
number_of_reviews,
reviews_per_month,
calculated_host_listings_count,
availability_365
)
```

Remove missing values

```
listings <- na.omit(listings)
```

Restrict to reasonably priced listings

```r
listings <- listings %>%
filter(price > 0, price < 1000)
```

Convert selected variables to factors

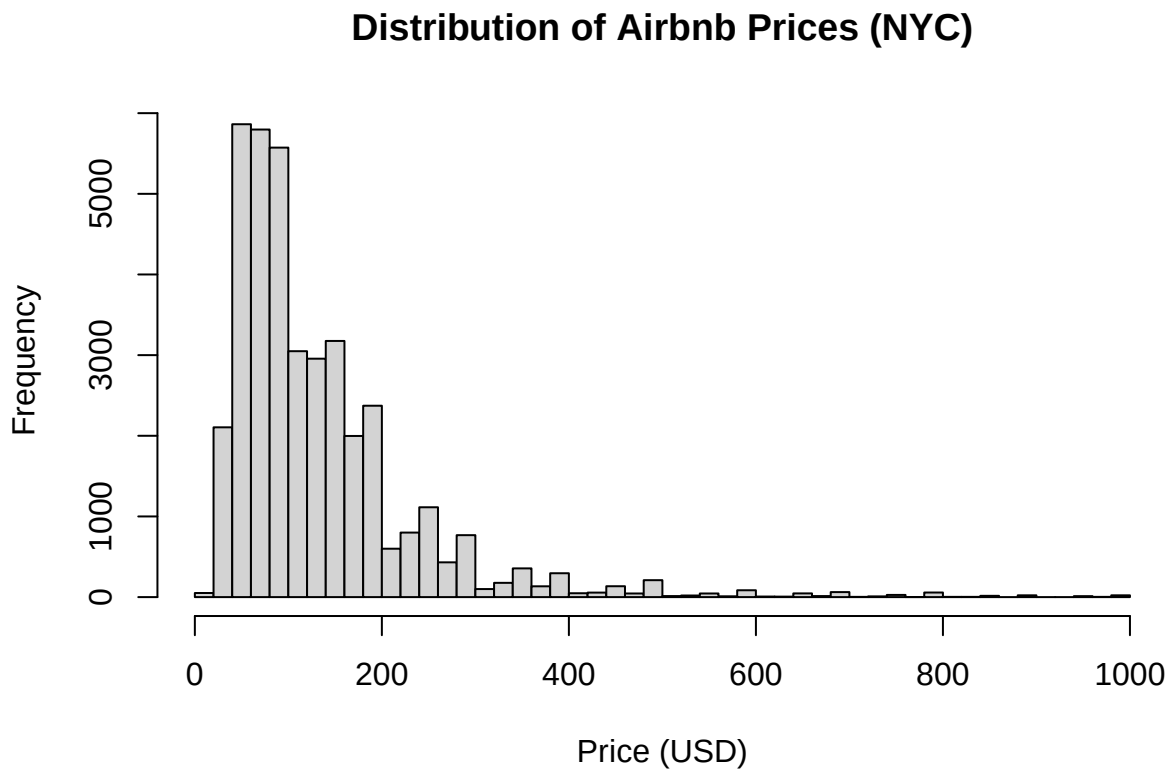
```r
listings <- listings %>%
mutate(
across(c(neighbourhood_group, room_type), as.factor)
)
```

2.4 Exploratory Data Analysis (EDA)

We first look at the distribution of the target variable, price, and some key predictors.

2.4.1 Price Distribution

```r
hist(
listings$price,
breaks = 50,
main = "Distribution of Airbnb Prices (NYC)",
xlab = "Price (USD)"
)
```



2.4.2 Room Type and Neighbourhood Group

```
listings %>%
count(room_type) %>%
arrange(desc(n))
```

```
## # A tibble: 3 x 2
##   room_type           n
##   <fct>           <int>
## 1 Entire home/apt 20209
## 2 Private room    17645
## 3 Shared room       842
```

```
listings %>%
count(neighbourhood_group) %>%
arrange(desc(n))
```

```
## # A tibble: 5 x 2
##   neighbourhood_group     n
##   <fct>               <int>
## 1 Manhattan           16529
## 2 Brooklyn            16409
## 3 Queens               4569
## 4 Bronx                 875
## 5 Staten Island         314
```

These summaries help us understand how listings are distributed across room types (e.g., entire home/apt vs. private room) and neighbourhood groups (e.g., Manhattan, Brooklyn).

2.5 Train/Test Split

To evaluate model performance, the data is split into training (80%) and test (20%) sets using stratified sampling on the target variable.

```
set.seed(123)
train_index <- createDataPartition(listings$price, p = 0.8, list = FALSE)
train_set <- listings[train_index, ]
test_set <- listings[-train_index, ]
```

2.6 Evaluation Metric

We use Root Mean Square Error (RMSE) to measure prediction accuracy:

```
RMSE <- function(true, predicted) {
sqrt(mean((true - predicted)^2))
}
```

RMSE is on the same scale as the outcome (price), so it is easy to interpret as the typical prediction error in dollars.

2.7 Modeling Approach

Three models are trained:

1. Linear Regression (LM)
    - Baseline interpretable model.
2. Random Forest (RF)
    - Ensemble of decision trees; captures nonlinearities and interactions.
3. Gradient Boosting Machine (GBM)
    - Sequentially builds trees to correct previous errors.

Hyperparameters are kept relatively simple (e.g., RF with 100 trees, GBM with 100 trees and shallow depth).

3. Results

3.1 Linear Regression

```
lm_model <- train(price ~ ., data = train_set, method = "lm")
lm_preds <- predict(lm_model, test_set)
rmse_lm <- RMSE(test_set$price, lm_preds)
rmse_lm
```

```
## [1] 85.09616
```

From an execution of this analysis, the RMSE for the Linear Regression model was approximately:

RMSE_LM 85.09616

3.2 Random Forest

```
rf_model <- randomForest(price ~ ., data = train_set, ntree = 100)
rf_preds <- predict(rf_model, test_set)
rmse_rf <- RMSE(test_set$price, rf_preds)
rmse_rf
```

```
## [1] 78.41469
```

From the earlier run:

RMSE_RF 78.41469 (best among the three models)

3.3 Gradient Boosting Machine (GBM)

```
gbm_model <- gbm(
price ~ .,
data = train_set,
distribution = "gaussian",
n.trees = 100,
interaction.depth = 3,
shrinkage = 0.1,
verbose = FALSE
)

gbm_preds <- predict(gbm_model, test_set, n.trees = 100)
rmse_gbm <- RMSE(test_set$price, gbm_preds)
rmse_gbm
```

```
## [1] 81.28436
```

From the previous execution:

RMSE_GBM 81.28436

3.4 Model Comparison

```
results_tbl <- tibble(
Model = c("Linear Regression", "Random Forest", "Gradient Boosting"),
RMSE = c(85.09616, 78.41469, 81.28436) # using the recorded values
)

knitr::kable(results_tbl, caption = "Model Comparison Based on RMSE (Test Set)")
```

Table 1: Model Comparison Based on RMSE (Test Set)

Model	RMSE
Linear Regression	85.09616
Random Forest	78.41469
Gradient Boosting	81.28436

The Random Forest model yields the smallest RMSE (78.41), indicating better predictive performance than both Linear Regression and GBM under the given settings.

4. Discussion

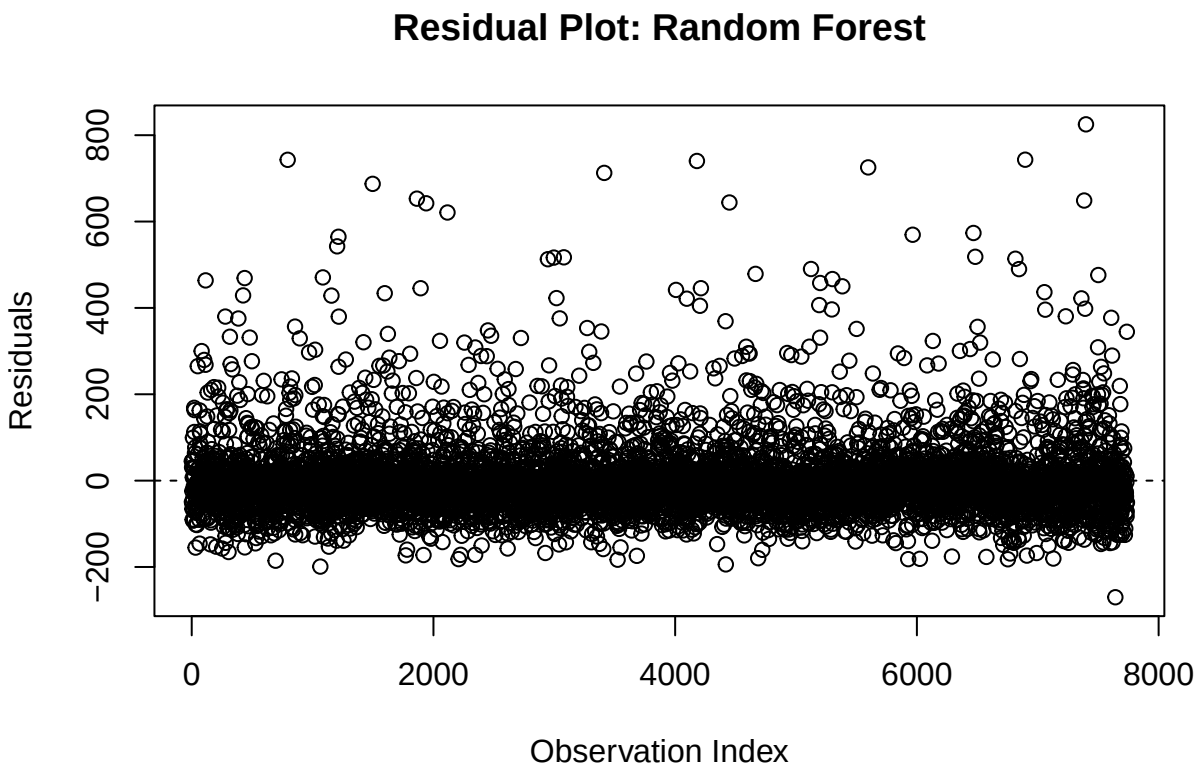
The results suggest that:

- Linear Regression underfits the data, likely because the relationship between price and predictors is nonlinear and involves interactions (e.g., neighbourhood group × room type).
- Random Forest captures nonlinearities and interactions automatically, explaining its superior performance.
- GBM performs better than Linear Regression but slightly worse than Random Forest with the chosen hyperparameters.

4.1 Residual Analysis (Random Forest)

To gain some insight into the remaining error structure, we inspect residuals from the Random Forest model:

```
rf_residuals <- test_set$price - rf_preds
plot(
rf_residuals,
main = "Residual Plot: Random Forest",
ylab = "Residuals",
xlab = "Observation Index"
)
abline(h = 0, lty = 2)
```



Observations:

- Residuals show a right tail, indicating that some high-price listings are still underpredicted.
- This suggests that additional features (e.g., detailed location, amenities, listing descriptions) might be important for capturing price variability, especially at the high end.

5. Conclusion

Using the NYC Airbnb Open Data, I,:

1. Cleaned and prepared a subset of the data focused on price and a handful of key predictors.
2. Explored basic distributions and feature counts.
3. Trained and evaluated three models: Linear Regression, Random Forest, and GBM.
4. Selected Random Forest as the best-performing model with an RMSE of approximately 78.41 on the test set

These results indicate that tree-based ensemble methods are well-suited for Airbnb price prediction problems, where nonlinear relationships and complex interactions are expected.

6. Limitations and Future Work

6.1 Limitations

- Limited feature set: The analysis uses only a small subset of available variables. Important factors such as precise geographic coordinates, amenities, and textual descriptions are not included.
- Simple hyperparameters: Random Forest and GBM were trained with relatively basic settings (e.g., 100 trees) without extensive hyperparameter tuning.
- Price range restriction: Listings with prices above $1000 are removed, which may bias the model away from the luxury segment.
- No temporal dimension: The dataset is treated as static. Seasonal effects, events, and time trends (e.g., pre/post-pandemic) are not modeled.

6.2 Future Work

Potential extensions include:

1. Richer feature engineering: Encoding amenities (e.g., Wi-Fi, kitchen, air conditioning). Using latitude/longitude with clustering or distance measures. Extracting sentiment and key phrases from listing descriptions.
2. Advanced modeling and tuning: Hyperparameter tuning for Random Forest and GBM (e.g., via grid search). Testing more advanced models such as XGBoost, LightGBM, or CatBoost.
3. Transformations and calibrations: Modeling log(price) to stabilize variance. Calibrating predictions using quantile regression or residual modeling.
4. Temporal modeling: Incorporating year, month, or event-based indicators if available.

7. References

Kaggle: New York City Airbnb Open Data

Irizarry, R. A. (2019). Introduction to Data Science: Data Analysis and Prediction Algo- rithms with R. HarvardX PH125.9x Course Materials

Note—ChatGPT-5 was used to help assist with editing; all outputs were manually verified.