Predicting Airbnb Booking Popularity in Washington, DC

Project Overview

This project explores a dataset of Airbnb listings in Washington, DC, to predict whether a listing has a "high booking rate," indicating frequent bookings.

Objectives

- **EDA & Cleaning:** Explore the data to understand structure, discover patterns, and handle missing values.
- **Model Development:** Build linear and logistic regression models to predict the target variable, `high_booking_rate`.
- **Interpretation & Insights:** Identify key features influencing booking popularity.
- **Model Evaluation:** Validate model performance on training and testing sets using relevant classification metrics.

Dataset Description

The dataset includes 4,936 Airbnb listings with property details, location, and booking information. The goal is to predict `high_booking_rate`, showing the likelihood of a listing being frequently booked.

Approach

- **Exploration & Visualization:** Identify key trends using Python.
- **Modeling:** Develop regression models, applying feature engineering to improve accuracy.
- **Validation:** Evaluate models based on metrics like accuracy, precision, recall, and AUC.

This project enhances my skills in data exploration, modeling, and predictive analysis within the short-term rental market context.

```r
airbnb <- read_csv("dc_airbnb_hw1.csv")  #read the dataset in R
```

```
## Rows: 4936 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): name, bed_type, cancellation_policy, cleaning_fee, price, property_...
## dbl (5): accommodates, bedrooms, beds, host_total_listings_count, high_booki...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
names(airbnb)                    #variables used in dataset
```

```
##  [1] "name"                      "accommodates"
##  [3] "bed_type"                  "bedrooms"
##  [5] "beds"                      "cancellation_policy"
##  [7] "cleaning_fee"              "host_total_listings_count"
##  [9] "price"                     "property_type"
## [11] "room_type"                 "high_booking_rate"
```

```r
age_mean <- airbnb %>%
  summarise(mean_accommodates = mean(accommodates))
```

1: EDA and Data Cleaning

a) To prepare the dataset for analysis, the following cleaning and preprocessing steps were performed:

- **Cancellation Policy Grouping:** Combined the `"strict"` and `"super_strict_30"` categories into a single `"strict"` category to simplify analysis.
- **Convert to Numeric:** Transformed `cleaning_fee` and `price` into numeric values.
- **Handling Missing Values (NAs):**
  - Replaced missing values in `cleaning_fee` and `price` with `0`.
  - Imputed missing values in other numeric variables with their mean.

```r
data_airbnb_clean <- airbnb %>%
  mutate(cancellation_policy = ifelse(cancellation_policy %in% c("strict", "super_strict_30"), "strict", cancella
tion_policy),
   cleaning_fee = as.numeric(gsub("[$,]", "", cleaning_fee)),
   price = as.numeric(gsub("[$,]", "", price)),
   cleaning_fee = ifelse(is.na(cleaning_fee), 0, cleaning_fee),
   price = ifelse(is.na(price), 0, price)
  )

num_cols <- c("accommodates", "bedrooms", "beds", "host_total_listings_count")

data_airbnb_clean <- data_airbnb_clean %>%
  mutate_at(num_cols, ~replace_na(., mean(., na.rm = TRUE)))
```

b) Feature Engineering: Creating New Variables

To enhance the dataset, the following new variables were created:

- **price_per_person:** Calculated as the nightly `price` divided by the `accommodates`, representing the cost per person per night.
- **has_cleaning_fee:** A binary variable indicating whether a cleaning fee exists. Coded as `"YES"` if `cleaning_fee` > 0, `"NO"` otherwise.
- **bed_category:** A categorical variable with values `"bed"` if `bed_type` is `"Real Bed"` and `"other"` for all other bed types.
- **property_category:** Categorized based on the `property_type`:
  - `"apartment"` for `"Apartment"`, `"Serviced apartment"`, or `"Loft"`.
  - `"hotel"` for `"Bed & Breakfast"`, `"Boutique hotel"`, or `"Hostel"`.
  - `"condo"` for `"Townhouse"` or `"Condominium"`.
  - `"house"` for `"Bungalow"` or `"House"`.
  - `"other"` for any other property types. This variable was converted to a factor for analysis.
- **ppp_ind:** A binary indicator where `1` denotes that `price_per_person` is greater than the median `price_per_person` within its `property_category`, and `0` otherwise.

```r
data1 <- data_airbnb_clean %>%
  mutate(
    price_per_person = price/accommodates,
    has_cleaning_fee = ifelse(cleaning_fee > 0, 'YES', 'NO'),
    bed_category = ifelse(bed_type == "Real Bed", "bed", "other"),
    property_category = case_when(
      property_type %in% c("Apartment", "Serviced apartment", "Loft") ~ "apartment",
      property_type %in% c("Bed & Breakfast", "Boutique hotel", "Hostel") ~ "hotel",
      property_type %in% c("Townhouse", "Condominium") ~ "condo",
      property_type %in% c("Bungalow", "House") ~ "house",
      TRUE ~ "other") ) %>%
  group_by(property_category) %>%
  mutate(
    ppp_ind = ifelse(price_per_person > median(price_per_person), 1, 0)
  ) %>%
  ungroup() %>%
  mutate(
    property_category = factor(property_category, levels = c("apartment", "hotel", "condo", "house", "other"))
  )
```

c) Converting the remaining character variables to factors:

- bed_type
- cancellation_policy
- room_type

```r
data2 <- data1 %>%
  mutate(
    bed_type = factor(bed_type),
    cancellation_policy = factor(cancellation_policy),
    room_type = factor(room_type) )
```

d) Visualizing Price Per Person by Booking Rate

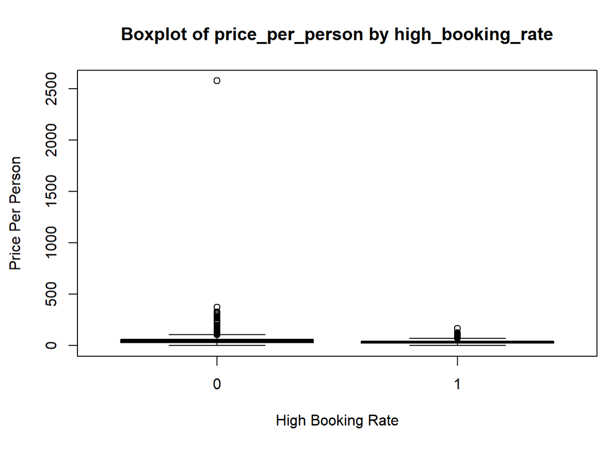
To examine the relationship between the nightly price per person and the booking rate, boxplots were constructed for both `price_per_person` and `log(price_per_person)`, grouped by `high_booking_rate`.

- **Boxplots of `price_per_person`:** The distribution of prices per person is visualized across different values of `high_booking_rate`.
- **Boxplots of `log(price_per_person)`:** A logarithmic transformation is applied to `price_per_person` to better visualize the distribution and reduce the effect of outliers.

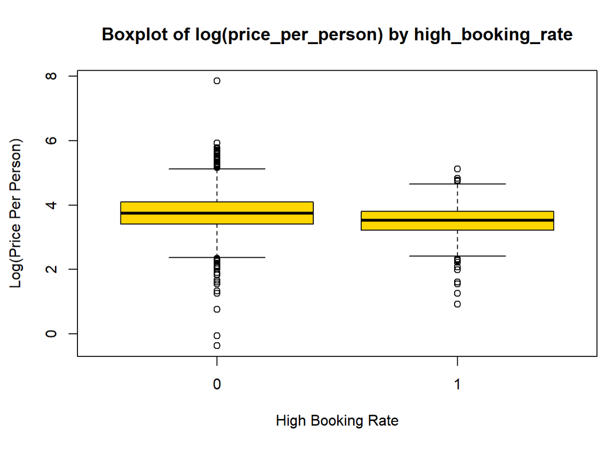
Observations:

```r
library(ggplot2)

#boxplot of price_per_person
boxplot(price_per_person ~ high_booking_rate, data = data2,
        main = "Boxplot of price_per_person by high_booking_rate",
        xlab = "High Booking Rate",
        ylab = "Price Per Person",
        col = "red")
```




```r
#boxplot of log(price_per_person)
boxplot(log(price_per_person) ~ high_booking_rate, data = data2,
        main = "Boxplot of log(price_per_person) by high_booking_rate",
        xlab = "High Booking Rate",
        ylab = "Log(Price Per Person)",
        col = "gold")
```



Observations: There appears to be negative relationship between price_per_person and high_booking_rate.

e) To explore how booking rates vary across property types, a two-way table was constructed between `high_booking_rate` and `property_category`.

```r
#two way table
high_book_rate_2way_t <- table(data2$high_booking_rate, data2$property_category)
high_book_rate_2way_t
```

```
##
##     apartment hotel condo house other
##   0      2013    50   387  1198    50
##   1       672    12   171   363    20
```

```r
#prop table
high_book_rate_prop_t <- prop.table(high_book_rate_2way_t, margin = 2)
high_book_rate_prop_t
```

```
##
##     apartment     hotel     condo     house     other
##   0 0.7497207 0.8064516 0.6935484 0.7674568 0.7142857
##   1 0.2502793 0.1935484 0.3064516 0.2325432 0.2857143
```

2: Linear Regression

a) Training a linear regression to predict high_booking_rate using the variables listed below.

- cancellation_policy
- cleaning_fee
- price_per_person
- ppp_ind
- has_cleaning_fee
- accommodates
- bed_category
- bedrooms
- beds
- host_total_listings_count
- property_category

```r
model1 <- lm(high_booking_rate ~ cancellation_policy + cleaning_fee + price_per_person + ppp_ind + has_cleaning_f
ee + accommodates + bed_category + bedrooms + beds + host_total_listings_count + property_category, data = data2)

summary(model1)$r.squared
```

```
## [1] 0.07898232
```

Model 1 has an R^2 of 0.7898.

b) Given a set of listing characteristics, the goal is to predict the `high_booking_rate`. The prediction is based on the model built earlier in the analysis.

New Listing:

- cancellation_policy = super_strict_30
- cleaning_fee = $30
- price = $200
- accommodates = 4
- bed_type = Real Bed
- bedrooms = 3
- beds = 4
- host_total_listings_count = 1
- property_type = townhouse

```r
new_listing <- data.frame(cancellation_policy = "strict",
                          cleaning_fee = 30, price_per_person = 200/4,
                          ppp_ind = 1, has_cleaning_fee = "YES",
                          accommodates = 4,
                          bed_category = "other",
                          bedrooms = 3,
                          beds = 4,
                          host_total_listings_count = 1,
                          property_category = "condo"
)



high_booking_rate_pred <- predict(model1, newdata = new_listing)
high_booking_rate_pred
```

```
##         1
## 0.2602891
```

The high_booking_rate prediction value for the new listing, was closer to 0 than it is to 1, which suggests that our model has predicted that this new listing will not have a high booking rate.

3: Logistic Regression

a) Training a Logistic Regression model using the same variables as in Linear Regression model:

```r
model1_logit <- glm(high_booking_rate ~ cancellation_policy + cleaning_fee + price_per_person + ppp_ind + has_cle
aning_fee + accommodates + bed_category + bedrooms + beds + host_total_listings_count + property_category, data =
data2, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(model1_logit)
```

```
##
## Call:
## glm(formula = high_booking_rate ~ cancellation_policy + cleaning_fee +
##     price_per_person + ppp_ind + has_cleaning_fee + accommodates +
##     bed_category + bedrooms + beds + host_total_listings_count +
##     property_category, family = binomial, data = data2)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -0.978980   0.160907  -6.084 1.17e-09 ***
## cancellation_policymoderate  0.510296   0.093378   5.465 4.63e-08 ***
## cancellation_policystrict    0.444980   0.095816   4.644 3.42e-06 ***
## cleaning_fee                -0.006897   0.001299  -5.308 1.11e-07 ***
## price_per_person            -0.019105   0.002891  -6.608 3.91e-11 ***
## ppp_ind                     -0.053136   0.108611  -0.489   0.6247
## has_cleaning_feeYES          0.841786   0.107402   7.838 4.59e-15 ***
## accommodates                 0.040726   0.035167   1.158   0.2468
## bed_categoryother           -0.264513   0.208151  -1.271   0.2038
## bedrooms                    -0.399557   0.069183  -5.775 7.68e-09 ***
## beds                         0.223325   0.050891   4.388 1.14e-05 ***
## host_total_listings_count   -0.007824   0.003194  -2.449   0.0143 *
## property_categoryhotel       0.100804   0.352798   0.286   0.7751
## property_categorycondo       0.349404   0.109380   3.194   0.0014 **
## property_categoryhouse      -0.052321   0.083447  -0.627   0.5307
## property_categoryother      -0.140461   0.291836  -0.481   0.6303
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5560.1  on 4935  degrees of freedom
## Residual deviance: 5082.6  on 4920  degrees of freedom
## AIC: 5114.6
##
## Number of Fisher Scoring iterations: 8
```

The AIC of this model is 5114.6.

b) The coefficient for price per person is -0.019105, which means that for a one-unit increase in price per person, the log-odds of the high_booking_rate is would decrease by 0.019105, if all other variables in the model remain constant.

c) The coefficient for the condo property category is 0.349404, which means that the log odds of a condo listing having a high booking rate is expected to be 0.349404 higher than the log odds of the base property_category, if all else remain constant.

d) To estimate the likelihood that the given listing has `high_booking_rate = 1`, we use the logistic regression model and compute the predicted probability.

```r
prob_pred<- predict(model1_logit, newdata=new_listing,type = "response")
print(prob_pred)
```

```
##         1
## 0.2743069
```

###e) Logistic regression is more suitable for binary outcomes, providing a more accurate and interpretable model for explaining the relationship between features and the target variable high_booking_rate.

4: Classification and Evaluation

a) The data is split into 70% training and 30% validation sets. The linear and logistic regression models are retrained using only the training data, and their performance metrics (R^2 for linear regression and AIC for logistic regression) are reported:

```r
set.seed(12345)

data_index <- sample(nrow(data2), 0.7 * nrow(data2))

train <- data2[data_index, ]
test <- data2[-data_index, ]

# linear regression model
model1_train <- lm(high_booking_rate ~ cancellation_policy + cleaning_fee + price_per_person + ppp_ind + has_clea
ning_fee + accommodates + bed_category + bedrooms + beds + host_total_listings_count + property_category,data = t
rain)

summary(model1_train)
```

```
##
## Call:
## lm(formula = high_booking_rate ~ cancellation_policy + cleaning_fee +
##     price_per_person + ppp_ind + has_cleaning_fee + accommodates +
##     bed_category + bedrooms + beds + host_total_listings_count +
##     property_category, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8996 -0.2870 -0.1789  0.3190  0.9853
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  2.111e-01  2.712e-02   7.785 9.15e-15 ***
## cancellation_policymoderate  9.104e-02  1.866e-02   4.879 1.11e-06 ***
## cancellation_policystrict    6.635e-02  1.890e-02   3.511 0.000453 ***
## cleaning_fee                -1.179e-03  2.273e-04  -5.189 2.24e-07 ***
## price_per_person            -1.305e-03  3.094e-04  -4.219 2.52e-05 ***
## ppp_ind                     -5.533e-02  1.870e-02  -2.958 0.003115 **
## has_cleaning_feeYES          1.440e-01  2.050e-02   7.024 2.58e-12 ***
## accommodates                 1.926e-02  7.402e-03   2.602 0.009310 **
## bed_categoryother           -8.357e-02  4.248e-02  -1.967 0.049238 *
## bedrooms                    -7.719e-02  1.405e-02  -5.493 4.24e-08 ***
## beds                         2.970e-02  1.056e-02   2.812 0.004953 **
## host_total_listings_count   -2.291e-04  8.464e-05  -2.707 0.006831 **
## property_categoryhotel       2.789e-03  6.497e-02   0.043 0.965763
## property_categorycondo       6.249e-02  2.359e-02   2.648 0.008124 **
## property_categoryhouse      -1.250e-02  1.711e-02  -0.730 0.465141
## property_categoryother       5.249e-03  6.103e-02   0.086 0.931474
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4158 on 3439 degrees of freedom
## Multiple R-squared:  0.08106,    Adjusted R-squared:  0.07705
## F-statistic: 20.22 on 15 and 3439 DF,  p-value: < 2.2e-16
```

```r
#logistic regression model
model1_logit_train <- glm(high_booking_rate ~ cancellation_policy + cleaning_fee + price_per_person + ppp_ind + h
as_cleaning_fee + accommodates + bed_category + bedrooms + beds + host_total_listings_count + property_category,f
amily = binomial, data = train)

summary(model1_logit_train)
```

```
##
## Call:
## glm(formula = high_booking_rate ~ cancellation_policy + cleaning_fee +
##     price_per_person + ppp_ind + has_cleaning_fee + accommodates +
##     bed_category + bedrooms + beds + host_total_listings_count +
##     property_category, family = binomial, data = train)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -1.126890   0.190968  -5.901 3.61e-09 ***
## cancellation_policymoderate  0.548769   0.110561   4.963 6.92e-07 ***
## cancellation_policystrict    0.436075   0.114053   3.823 0.000132 ***
## cleaning_fee                -0.007411   0.001579  -4.693 2.69e-06 ***
## price_per_person            -0.017161   0.003293  -5.211 1.88e-07 ***
## ppp_ind                     -0.035076   0.127379  -0.275 0.783035
## has_cleaning_feeYES          0.887613   0.127960   6.937 4.01e-12 ***
## accommodates                 0.101218   0.043782   2.312 0.020784 *
## bed_categoryother           -0.480226   0.265011  -1.812 0.069971 .
## bedrooms                    -0.410053   0.082087  -4.995 5.87e-07 ***
## beds                         0.149429   0.060057   2.488 0.012842 *
## host_total_listings_count   -0.008652   0.004121  -2.100 0.035753 *
## property_categoryhotel       0.159711   0.430675   0.371 0.710757
## property_categorycondo       0.331978   0.130334   2.547 0.010861 *
## property_categoryhouse      -0.129471   0.100368  -1.290 0.197066
## property_categoryother      -0.029485   0.337172  -0.087 0.930316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3881.9  on 3454  degrees of freedom
## Residual deviance: 3553.3  on 3439  degrees of freedom
## AIC: 3585.3
##
## Number of Fisher Scoring iterations: 8
```

Linear Model: - R^2: 0.08106

Logistic Regression Model: - AIC: 3585.3

b) Computing the RMSE in the training and validation sets for the linear model:

```r
#predicting and computing rmse with training data
train_pred <- predict(model1_train, newdata = train )
rmse_train <- sqrt(mean((train_pred - train$high_booking_rate)^2))

#predicting and computing rmse with validation data
test_pred <- predict(model1_train, newdata = test)
rmse_test <- sqrt(mean((test_pred - test$high_booking_rate)^2))


rmse_train
```

```
## [1] 0.4148109
```

```r
rmse_test
```

```
## [1] 0.4250136
```

RMSE of training data = 0.4148109 RMSE of validation data = 0.4250136

The validation set having a higher RMSE is normal. When training a model, it's fine-tuned to minimize errors on the data it learns from, resulting in a lower RMSE for the training set. However, when the model faces new, unseen data, it might struggle to adapt, causing higher prediction errors and a higher RMSE. This performance difference shows how well the model handles unfamiliar data. A much higher RMSE on the validation set could mean the model is too fixated on the training data, picking up noise instead of the actual patterns in the data.

c) Using the models trained on the training dataset, we predict `high_booking_rate` on the validation dataset with a threshold of 0.5 to classify the outcome. Confusion matrices are then generated to evaluate model performance:

```r
#linear regression confusion matrix
lm_pred <- predict(model1, newdata = test)
binary_lm_pred <-  ifelse(lm_pred >= 0.5, 1, 0)

lm_conf_matrix <- table(Actual = test$high_booking_rate, Predicted = binary_lm_pred)

print("Linear Regression Model Confusion Matrix")
```

```
## [1] "Linear Regression Model Confusion Matrix"
```

```r
print(lm_conf_matrix)
```

```
##       Predicted
## Actual    0    1
##      0 1100    5
##      1  371    5
```

```r
#logistic regression confusion matrix
glm_pred <- predict(model1_logit, newdata = test, type = "response")
binary_glm_pred <-  ifelse(glm_pred >= 0.5, 1, 0)

glm_conf_matrix <- table(Actual = test$high_booking_rate, Predicted = binary_glm_pred)

print("Logistic Regression Model Confusion Matrix")
```

```
## [1] "Logistic Regression Model Confusion Matrix"
```

```r
print(glm_conf_matrix)
```

```
##       Predicted
## Actual    0    1
##      0 1082   23
##      1  351   25
```

d) Reporting the accuracy, TPR, and FPR:

```r
#extracting values
TN <- lm_conf_matrix[1, 1]
FP <- lm_conf_matrix[1, 2]
FN <- lm_conf_matrix[2, 1]
TP <- lm_conf_matrix[2, 2]

#accuracy
lm_accuracy <- (TP + TN) / sum(lm_conf_matrix)
lm_tpr <- TP / (TP + FN)
lm_fpr <- FP / (FP + TN)

print("Linear Regression Model accuracy")
```

```
## [1] "Linear Regression Model accuracy"
```

```r
lm_accuracy
```

```
## [1] 0.7461175
```

```r
print("Linear Regression Model True Positive Rate")
```

```
## [1] "Linear Regression Model True Positive Rate"
```

```r
lm_tpr
```

```
## [1] 0.01329787
```

```r
print("Linear Regression Model False Positive Rate")
```

```
## [1] "Linear Regression Model False Positive Rate"
```

```r
lm_fpr
```

```
## [1] 0.004524887
```

```r
print("-------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------"
```

```r
# Extract values from confusion matrix

log_TN <- glm_conf_matrix[1, 1]
log_FP <- glm_conf_matrix[1, 2]
log_FN <- glm_conf_matrix[2, 1]
log_TP <- glm_conf_matrix[2, 2]

#accuracy
log_accuracy <- (log_TP + log_TN) / sum(glm_conf_matrix)
log_tpr <- log_TP / (log_TP + log_FN)
log_fpr <- log_FP / (log_FP + log_TN)

print("Logistic Model accuracy")
```

```
## [1] "Logistic Model accuracy"
```

```r
log_accuracy
```

```
## [1] 0.7474679
```

```r
print("Linear Regression Model True Positive Rate")
```

```
## [1] "Linear Regression Model True Positive Rate"
```

```r
log_tpr
```

```
## [1] 0.06648936
```

```r
print("False Positive Rate")
```

```
## [1] "False Positive Rate"
```

```r
log_fpr
```

```
## [1] 0.02081448
```

e) For the more accurate model (either linear or logistic based on previous evaluations), we calculate accuracy, TPR (sensitivity), and FPR (1-specificity) using cutoffs of 0.4 and 0.6 on the validation data.

ANSWER TO QUESTION 4e HERE:

```r
# Predicted probabilities from the logistic regression model

glm_pred <- predict(model1_logit_train, newdata = test, type = "response")



# Create confusion matrix using cutoff of 0.4

glm_pred_0.4 <- ifelse(glm_pred >= 0.4, 1, 0)

glm_matrix_0.4 <- table(glm_pred_0.4, test$high_booking_rate)



# Calculate accuracy, TPR, and FPR using cutoff of 0.4

accuracy_0.4 <- sum(diag(glm_matrix_0.4)) / sum(glm_matrix_0.4)
tpr_0.4 <- glm_matrix_0.4[2, 2] / sum(glm_matrix_0.4[2, ])
fpr_0.4 <- glm_matrix_0.4[2, 1] / sum(glm_matrix_0.4[1, ])



# Creating confusion matrix using cutoff of 0.6

log_binary_0.6 <- ifelse(glm_pred >= 0.6, 1, 0)
glm_matrix_0.6 <- table(log_binary_0.6, test$high_booking_rate)



# Calculating accuracy, TPR, and FPR using cutoff of 0.6

accuracy_0.6 <- sum(diag(glm_matrix_0.6)) / sum(glm_matrix_0.6)
tpr_0.6 <- glm_matrix_0.6[2, 2] / sum(glm_matrix_0.6[2, ])
fpr_0.6 <- glm_matrix_0.6[2, 1] / sum(glm_matrix_0.6[1, ])



#results
print("Accuracy 0.4 cutoff")
```

```
## [1] "Accuracy 0.4 cutoff"
```

```r
accuracy_0.4
```

```
## [1] 0.7359892
```

```r
print("TPR 0.4 cutoff")
```

```
## [1] "TPR 0.4 cutoff"
```

```r
tpr_0.4
```

```
## [1] 0.4651163
```

```r
print("FPR 0.4 cutoff")
```

```
## [1] "FPR 0.4 cutoff"
```

```r
fpr_0.4
```

```
## [1] 0.09083728
```

```r
#results
print("Accuracy 0.6 cutoff")
```

```
## [1] "Accuracy 0.6 cutoff"
```

```r
accuracy_0.6
```

```
## [1] 0.7461175
```

```r
print("TPR 0.6 cutoff")
```

```
## [1] "TPR 0.6 cutoff"
```

```r
tpr_0.6
```

```
## [1] 0.5
```

```r
print("FPR 0.6 cutoff")
```

```
## [1] "FPR 0.6 cutoff"
```

```r
fpr_0.6
```

```
## [1] 0.0006761325
```

The Cutoff with the highest accuracy is the 0.6 cutoff. The Cutoff with the highest TPR is the 0.6 cutoff. The Cutoff with the highest FPR is the 0.4 cutoff.

In this case, the 0.6 cutoff appears to be the most promising for the given application. It has a relatively high accuracy of 74.61%, which is close to the 0.5 cutoff. Additionally, the 0.6 cutoff boasts an impressive True Positive Rate (TPR) of 50% and an extremely low False Positive Rate (FPR) of 0.07%.

While the 0.4 cutoff has a comparable accuracy of 73.60%, its TPR is lower at 46.51%, and the FPR is slightly higher at 9.08%. The 0.6 cutoff strikes a good balance by achieving a high TPR while keeping the FPR exceptionally low, making it a favorable choice for this application.