Brooklyn Airbnb Pricing

Shashi Roshan (sroshan2@illinois.edu)

Abstract

Predicting the prices of real estate rental listings can be of interest to online marketplaces like Airbnb. It can also be used to suggest an optimal property listing price to a property owner. Statistical learning techniques are used to determine if it is possible to effectively predict the price of real estate rental properties.

Introduction

The prices of real estate rental properties can depend on a lot of factors, like the location, reviews, minimum days of booking required, etc. Many online marketplaces list real estate rental properties and manage their pricing. If the listed properties are priced optimally, it will serve two purposes simultaneously - maintain profit for these marketplaces, and keep the customers happy about the prices. Hence, optimally pricing the listed properties benefits both the marketplace and the customers. In addition, often it is the case that the properties are not owned by these marketplaces. In case a real estate rental property manager want to list their properties on the online marketplaces, then they need to come up with a price. For new customers, it can be a difficult task.

We can use data to predict the price of real estate rental properties. Features like the location, past prices, number of reviews, quality of reviews, etc can be used to do these predictions. However, the effectiveness of these predictions will depend on how comprehensive the data is, and how well we can leverage it.

We have collected data from Airbnb, which is an online marketplace which provides lodging, primarily homestays, or tourism experiences. The data is about price of Airbnb rentals in Brooklyn, NY. Statistical learning techniques were applied on this data to make predictions for the prices of real estate rental properties. However, few practical and statistical limitations suggest the need for further investigation.

Methods

Data

The data was accessed via Kaggle. [1] It contains information on Airbnb listings in New York, NY during 2019 including price, rental attributes, and location. For the purposes of this analysis, the data was restricted to short term (one week or less) rentals in Brooklyn that rent for less than $1000 a night. (Additionally, only rentals that have been reviewed are included.)

[1] New York City Airbnb Open Data

```
airbnb = read_csv(file = "data/AB_NYC_2019.csv")
```

```
brooklyn = airbnb %>%
  filter(minimum_nights <= 7) %>%
  filter(neighbourhood_group == "Brooklyn") %>%
  filter(number_of_reviews > 0) %>%
  filter(price > 0, price < 1000) %>%
  na.omit() %>%
  select(
    latitude,
    longitude,
    room_type,
    price,
    minimum_nights,
    number_of_reviews,
    reviews_per_month,
    calculated_host_listings_count,
    availability_365
  ) %>%
  mutate(room_type = as.factor(room_type))
```

```
set.seed(42)
# test-train split
bk_tst_trn_split = initial_split(brooklyn, prop = 0.80)
bk_trn = training(bk_tst_trn_split)
bk_tst = testing(bk_tst_trn_split)
# estimation-validation split
bk_est_val_split = initial_split(bk_trn, prop = 0.80)
bk_est = training(bk_est_val_split)
bk_val = testing(bk_est_val_split)
```

Modeling

In order to predict the price of rentals, three modeling techniques were considered: linear models, k-nearest neighbors models, and decision tree models.

- Linear models with and without log transformed responses were considered. Various subsets of predictors, with and without interaction terms were explored.
- k-nearest neighbors models were trained using all available predictor variables. The choice of k was chosen using a validation set.
- Decision tree models were trained using all available predictors. The choice of the complexity parameter was chosen using a validation set.

```
lm_1 = lm(price ~ ., data = bk_est)

# Loading MASS library here, since it was conflicting with select method used in creating "brooklyn" da
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lm_2 = stepAIC(lm_1, direction = "backward", trace = FALSE)

lm_3 = stepAIC(lm(price ~ . ^ 2, data = bk_est),
               direction = "backward",
               trace = FALSE)

lm_log_1 = lm(log(price) ~ ., data = bk_est)

lm_log_2 = stepAIC(lm_log_1, direction = "backward", trace = FALSE)

lm_log_3 =  stepAIC(lm(log(price) ~ . ^ 2, data = bk_est),
                    direction = "backward",
                    trace = FALSE)

linear_models_list = c()
linear_models_list[[length(linear_models_list) + 1]] = lm_1
linear_models_list[[length(linear_models_list) + 1]] = lm_2
linear_models_list[[length(linear_models_list) + 1]] = lm_3

linear_models_log_transformed_list = c()
linear_models_log_transformed_list[[length(linear_models_log_transformed_list) + 1]] = lm_log_1
linear_models_log_transformed_list[[length(linear_models_log_transformed_list) + 1]] = lm_log_2
linear_models_log_transformed_list[[length(linear_models_log_transformed_list) + 1]] = lm_log_3
```

```
k = 1:100

knn_models_list = c()

for (k_value in k) {
  knn_model = knnreg(price ~ ., data = bk_est, k = k_value)
  knn_models_list[[k_value]] = knn_model
}
```

```
cp = c(1.000, 0.100, 0.010, 0.001, 0)

tree_models_list = c()

for (cp_val in cp) {
  tree_model = rpart(price ~ ., data = bk_est, cp = cp_val)
  tree_models_list[[length(tree_models_list) + 1]] = tree_model
}
```

Evaluation

To evaluate the ability to predict rental prices, the data was split into estimation, validation, and testing sets. Error metrics and graphics are reported using the validation data in the Results section.

```r
calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}

calc_rmse_model = function(model, data, response) {
  actual = data[[response]]
  predicted = predict(model, data)
  sqrt(mean((actual - predicted) ^ 2))
}

calc_rmse_log_model = function(model, data, response) {
  actual = data[[response]]
  predicted = exp(predict(model, data))
  sqrt(mean((actual - predicted) ^ 2))
}
```

Results

```r
linear_model_results = c()

for (linear_model in linear_models_list) {
  rmse_validation = calc_rmse(bk_val$price, predict(linear_model, bk_val))
  linear_model_results[[length(linear_model_results) + 1]] = rmse_validation
}

linear_models_log_transformed_results = c()

for (linear_models_log_transformed in linear_models_log_transformed_list) {
  rmse_validation = calc_rmse_log_model(linear_models_log_transformed, bk_val, "price")
  linear_models_log_transformed_results[[length(linear_models_log_transformed_results) + 1]] = rmse_vali
}
```

```r
knn_model_results = c()

for (knn_model in knn_models_list) {
  rmse_validation = calc_rmse(bk_val$price, predict(knn_model, bk_val))
  knn_model_results[[length(knn_model_results) + 1]] = rmse_validation
}
```

```r
tree_model_results = c()

for (tree_model in tree_models_list) {
  rmse_validation = calc_rmse(bk_val$price, predict(tree_model, bk_val))
  tree_model_results[[length(tree_model_results) + 1]] = rmse_validation
}
```

```r
# Linear model
linear_model_summary = tibble(
  "Linear Models" = c(
    "Linear Model 1",
    "Linear Model 2",
    "Linear Model 3",
    "Linear Model 4",
    "Linear Model 5",
    "Linear Model 6"
  ),
  "Description" = c(
    "All available predictors",
    "All available predictors using backwards selection and AIC",
    "All available predictors, as well as all two-way interactions using backwards selection and AIC",
    "Log-transformed response and all available predictors",
    "Log-transformed response and selects from all available predictors using backwards selection and A
    "Log-transformed response and selects from all available predictors, as well as all two-way interac
  ),
  "Validation RMSE" = c(linear_model_results, linear_models_log_transformed_results)
)

linear_model_summary  %>%
  kable(digits = 3) %>%
  kable_styling("striped", full_width = FALSE)
```

Linear Models	Description
Linear Model 1	All available predictors
Linear Model 2	All available predictors using backwards selection and AIC
Linear Model 3	All available predictors, as well as all two-way interactions using backwards selection and AIC
Linear Model 4	Log-transformed response and all available predictors
Linear Model 5	Log-transformed response and selects from all available predictors using backwards selection and AIC
Linear Model 6	Log-transformed response and selects from all available predictors, as well as all two-way interactions us

```r
# KNN
min_removed_once = knn_model_results[knn_model_results != min(knn_model_results)]
min_removed_twice = min_removed_once[min_removed_once != min(min_removed_once)]

knn_model_summary = tibble(
  "KNN Models" = c("KNN Model 1", "KNN Model 2", "KNN Model 3"),
  "Number of neighbors" = c(
    which(knn_model_results == min(knn_model_results)),
    which(knn_model_results == min(min_removed_once)),
    which(knn_model_results == min(min_removed_twice))
  ),
  "Validation RMSE" = sort(knn_model_results)[1:3]
)

knn_model_summary  %>%
  kable(digits = 3) %>%
  kable_styling("striped", full_width = FALSE)
```

KNN Models	Number of neighbors	Validation RMSE
KNN Model 1	44	81.397
KNN Model 2	46	81.409
KNN Model 3	45	81.419

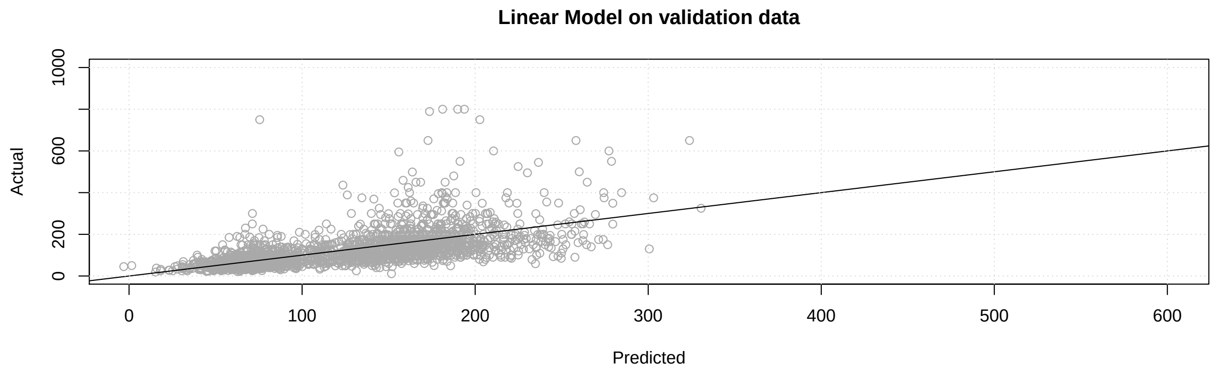
```r
# Tree model
tree_model_summary = tibble(
  "Tree Models" = c(
    "Tree Model 1",
    "Tree Model 2",
    "Tree Model 3",
    "Tree Model 4",
    "Tree Model 5"
  ),
  "cp" = c(cp),
  "Validation RMSE" = c(tree_model_results)
)

tree_model_summary  %>%
  kable(digits = 3) %>%
  kable_styling("striped", full_width = FALSE)
```

Tree Models	cp	Validation RMSE
Tree Model 1	1.000	85.391
Tree Model 2	0.100	71.289
Tree Model 3	0.010	69.471
Tree Model 4	0.001	69.017
Tree Model 5	0.000	73.552

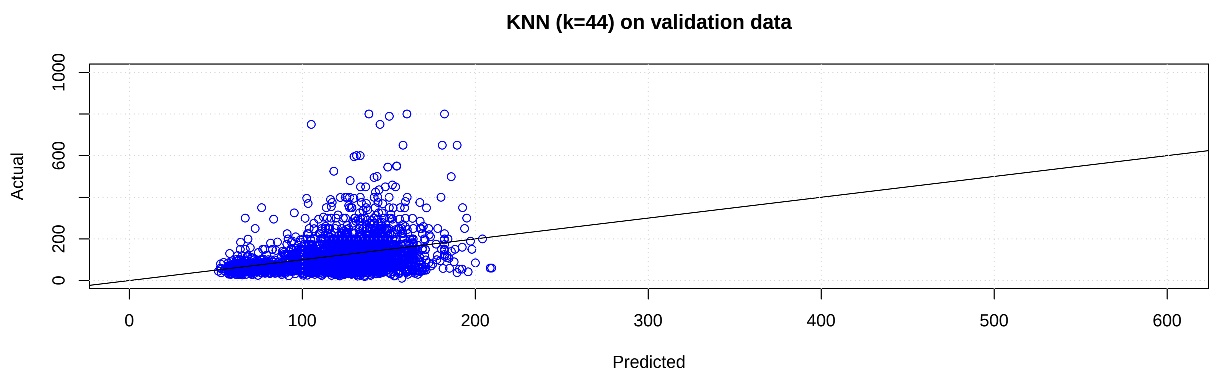
```r
# Best models
best_model_summary = tibble(
  "Model" = c("Linear Model",
              "KNN Model",
              "Tree Model"),
  "Description" = c(
    "All available predictors, as well as all two-way interactions using backwards selection and AIC",
    "K = 44",
    "cp = 0.001"
  ),
  "Validation RMSE" = c("66.285",
                        "81.397",
                        "69.017")
)

best_model_summary  %>%
  kable(digits = 3) %>%
  kable_styling("striped", full_width = FALSE)
```

Model	Description	Validatio
Linear Model	All available predictors, as well as all two-way interactions using backwards selection and AIC	66.285
KNN Model	K = 44	81.397
Tree Model	cp = 0.001	69.017

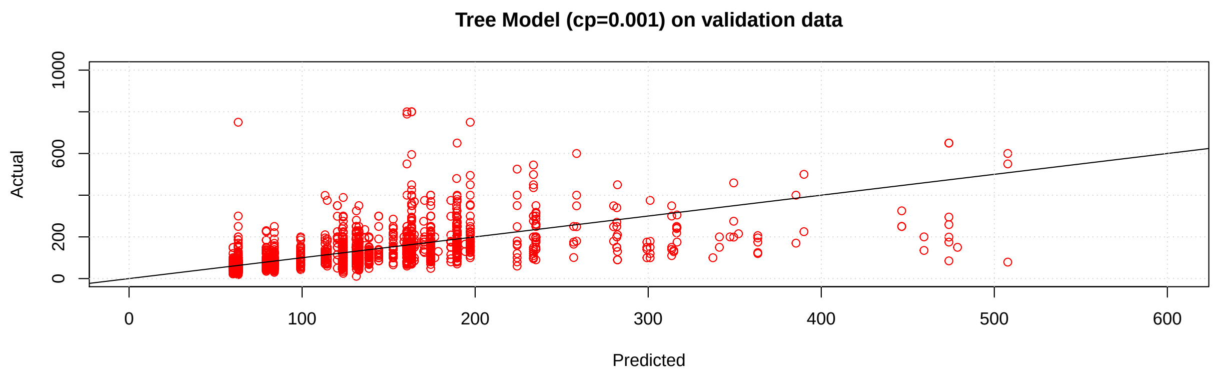
```r
# Linear Model
plot(
  predict(lm_3, bk_val),
  bk_val$price,
  xlim = c(1, 600),
  ylim = c(1, 1000),
  xlab = "Predicted",
  ylab = "Actual",
  main = "Linear Model on validation data",
  col = 'darkgrey'
)
abline(a = 0, b = 1)
grid()
```



```r
# KNN Model
plot(
  predict(knn_models_list[[44]], bk_val),
  bk_val$price,
  xlim = c(1, 600),
  ylim = c(1, 1000),
  xlab = "Predicted",
  ylab = "Actual",
  main = "KNN (k=44) on validation data",
  col = 'blue'
)
abline(a = 0, b = 1)
grid()
```



```r
# Tree Model
plot(
  predict(tree_models_list[[4]], bk_val),
  bk_val$price,
  xlim = c(1, 600),
  ylim = c(1, 1000),
  xlab = "Predicted",
  ylab = "Actual",
  main = "Tree Model (cp=0.001) on validation data",
  col = 'red'
)
abline(a = 0, b = 1)
grid()
```



Discussion

```r
# Best model : Linear Model 3, which uses all available predictors, as well as all two-way interactions
best_model = stepAIC(lm(price ~ . ^ 2, data = bk_trn),
                     direction = "backward",
                        trace = FALSE)

test_rmse = calc_rmse(bk_tst$price, predict(best_model, bk_tst))
```

The test RMSE score of the best model is 72.626

Given the results of assessing model performance on the validation data, the linear model appears to be a reasonable choice. It uses all available predictors, as well as all two-way interactions using backwards selection and AIC. Using the test data, we obtain a RMSE of $72.62. Compared to the range of prices seen in this data, ranging from roughly $10 to $999, with most observations greater than $60, this seems to suggest a model that is performing well at the prediction task. However, upon further inspection, recommending this model for use in practice would come with a few qualifications, both practical and statistical.

From a practical perspective, it seems reasonable to use past pricing data to make predictions for the future. The data that we used had rich set of features like the location (in terms of latitude, longitude), room type, minimum nights, number of reviews, etc. The data which we have used is from one city only, and the results of the statistical analysis might not scale well to other locations. We will need data from other cities as well. Also, pricing of rental properties might change with their availability, and it might be interesting to know the availability of similar rooms in the property, which the property was booked. Most of the features that we used for predicting the price don't change much with time, except the reviews.

By looking at the graph of actual vs predicted data, the model seems to be performing well for predicting lower priced properties, and was underestimating the prices of high priced properties. The reason might be that rental properties are costlier on few occasions, and the data which we used does not provide information about when the price of the rental property was recorded. Also, we can note that the RMSE value is close to the mean price of the data that was used for training - this suggests that the model might not be performing well, since the RMSE error is almost equal to the mean of the data.

In conclusion, the statistical learning techniques used on this data set provided good results, but can be improved further. One additional set of data that could have been really helpful is the time of the year when the price of the property was recorded. This is crucial, since festivals/few occasions during the year raise the prices of the rental properties. We can also expand this analysis to data which is collected from various locations.

Appendix

Data Dictionary

- `latitude` - latitude coordinates of the listing
- `longitude` - longitude coordinates of the listing
- `room_type` - listing space type
- `price` - price in dollars
- `minimum_nights` - amount of nights minimum
- `number_of_reviews` - number of reviews
- `reviews_per_month` - number of reviews per month
- `calculated_host_listings_count` - amount of listing per host
- `availability_365` - number of days when listing is available for booking

For additional background on the data, see the data source on Kaggle.