Choose Your Own Project: House Price Predictions

Hannah Siegel

10/30/2024

Introduction

This project will be submitted for the last course in HarvardX PH125.9x Data Science: Captstone for Project Submission: Choose Your Own. I have selected a dataset from Kaggle titled “House Price Prediction Dataset”, which I downloaded as a csv file. In this project, I will first analyze the data itself, and then apply various models to determine a predicted estimate for a house’s price given various independent variables. Using these models, I will calculate an RMSE based on my prediction on the test set and select the model with the smallest RMSE, and therefore the best predicting power.

Loading the Dataset and Packages

We load in the dataset given the data file that I attached with my project submission titled “House Price Prediction Dataset.csv”. I will also load in the required 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(data.table)) install.packages("data.table", repos =
"http://cran.us.r-project.org")

## read in the data
house_prices<- read.csv("House Price Prediction Dataset.csv")
```

Primary Review of the Data

Let’s start by looking at the dimensions of the data and the columns that are included in the dataset.

```
# dimensions of the data
ncol(house_prices)

## [1] 10

nrow(house_prices)

## [1] 2000

# column names and types
str(house_prices)
```

```
## 'data.frame':    2000 obs. of  10 variables:
##  $ Id       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Area     : int  1360 4272 3592 966 4926 3944 3671 3419 630 2185 ...
##  $ Bedrooms : int  5 5 2 4 1 1 1 2 2 3 ...
##  $ Bathrooms: int  4 4 2 2 4 2 1 4 2 3 ...
##  $ Floors   : int  3 3 3 2 2 1 2 1 1 1 ...
##  $ YearBuilt: int  1970 1958 1938 1902 1975 1906 1948 1925 1932 2000 ...
##  $ Location : chr  "Downtown" "Downtown" "Downtown" "Suburban" ...
##  $ Condition: chr  "Excellent" "Excellent" "Good" "Fair" ...
##  $ Garage   : chr  "No" "No" "No" "Yes" ...
##  $ Price    : int  149919 424998 266746 244020 636056 93262 448722 594893
652878 340375 ...
```

The dataset includes 2000 observations (rows) and 10 columns, including area (ie. square footage), number of bedrooms, number of bathrooms, floors, the year built, price, location, condition, and if the house has a garage.

```
min(house_prices$YearBuilt)

## [1] 1900

max(house_prices$YearBuilt)

## [1] 2023
```

We can see that the dataset includes homes that were built from 1900 to 2023.

Let's take a look at some of the other variables included in our dataset.

```
#locations included
unique(house_prices$Location)

## [1] "Downtown" "Suburban" "Urban"    "Rural"

#conditions included
unique(house_prices$Condition)

## [1] "Excellent" "Good"      "Fair"      "Poor"

#bedrooms included
sort(unique(house_prices$Bedrooms))

## [1] 1 2 3 4 5

#bedrooms included
sort(unique(house_prices$Bathrooms))

## [1] 1 2 3 4
```

Based on the above code, we see the following observations: 1. The dataset includes houses from 4 different locations: Downtown, Suburban, Urban and Rural. 2. The dataset includes houses in 4 different conditions: Excellent, Good, Fair and Poor. 3. The dataset includes houses wtih 1,2,3,4 and 5 bedrooms. 4. The dataset includes houses with 1,2,3 and 4 bathroom.

Pre-processing of the Data

We need to do some pre-processing to ensure that our data is ready for modeling. First, I will check if there are any blanks or NA in the housing prices.

```
# preprocessing of data. check if any NA or blank prices
check<- house_prices %>% filter(is.na(Price) | Price=="" | Price==" ")
nrow(check)

## [1] 0
```

There are no NAs or blanks, so let's continue.

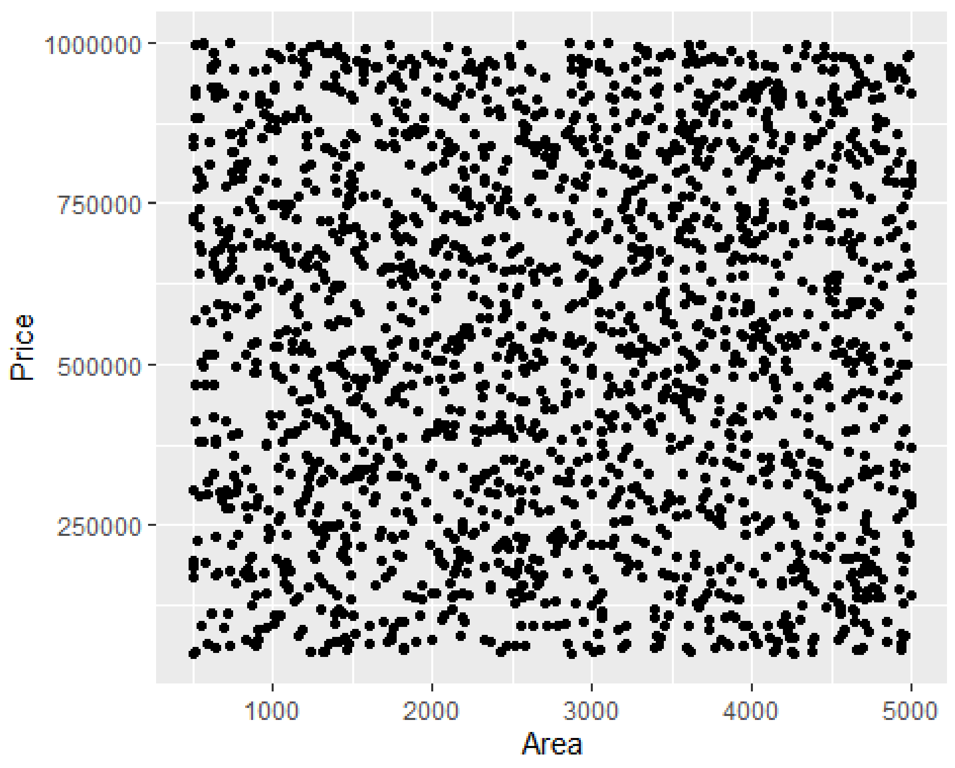
I also need to convert the categorical variables into dummy variables so that I am able to run a linear regression using the lm function in R (to come later in the report..). See below the following code.

```
# preprocessing of data: convert categorical variables (ie. location,
condition, garage)
house_prices<- house_prices %>% mutate(location_encoded =
case_when(Location=="Downtown" ~ 0,

Location=="Suburban" ~ 1,

Location=="Urban" ~ 2,

Location=="Rural" ~ 3),
                                       condition_encoded =
case_when(Condition=="Excellent" ~ 0,

Condition=="Good" ~ 1,

Condition=="Fair" ~ 2,

Condition=="Poor" ~ 3),
                                       garage_encoded =
case_when(Garage=="No" ~ 0,
                                                                 Garage ==
"Yes" ~ 1))
```

Secondary Review of the Data Let's begin looking at some of the relationships between the dependent variable (Prices) and some potential independent variables.

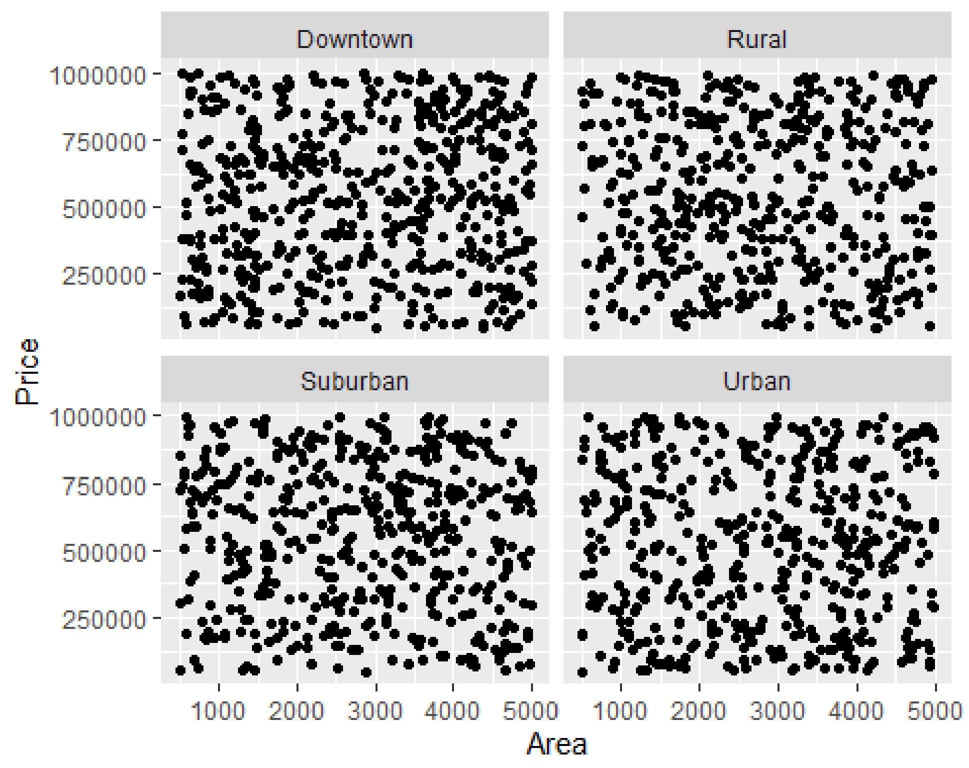
Let's start by looking at the visual relationship between area (ie. square footage) and price. My initial thought it that houses with a higher square footage should have a higher price.

```
ggplot(house_prices, aes(x = Area, y = Price)) +
  geom_point()
```



It is a bit unclear from this graph if my initial assumption is true. Let's take it one step further and see the relationship between area and price within each location.

```
ggplot(house_prices, aes(x = Area, y = Price)) +
  geom_point()+
  facet_wrap(~Location)
```



Once again, the relationship here is a bit unclear.

Let's see what the average price is per location.

```
per_location<- house_prices %>% group_by(Location) %>% summarise(avg_price =
mean(Price)) %>%
  arrange(desc(avg_price))
per_location

## # A tibble: 4 x 2
##   Location avg_price
##   <chr>        <dbl>
## 1 Suburban   557416.
## 2 Rural      538614.
## 3 Downtown   536060.
## 4 Urban      518964.
```

According to the above table, it appears that houses in suburbia have, on average, the highest prices and houses in urban areas have, on average, the lowest prices.

Let's see what the average price is per number of bedrooms.

```
per_bedroom<- house_prices %>% group_by(Bedrooms) %>% summarise(avg_price =
mean(Price)) %>%
  arrange(desc(avg_price))
per_bedroom
```

```
## # A tibble: 5 x 2
##   Bedrooms avg_price
##      <int>     <dbl>
## 1        3   546978.
## 2        2   545547.
## 3        4   533697.
## 4        5   532500.
## 5        1   530562.
```

Not to my suprise, 1 bedrooms homes have, on averge, the lowest prices. However, interesting to note that 5 bedroom homes do not, on average, have the highest prices.

Let's see what the average price is per number of bathrooms.

```
per_bathroom<- house_prices %>% group_by(Bathrooms) %>% summarise(avg_price =
mean(Price)) %>%
  arrange(desc(avg_price))
per_bathroom

## # A tibble: 4 x 2
##   Bathrooms avg_price
##       <int>     <dbl>
## 1         1   558758.
## 2         3   540202.
## 3         4   536478.
## 4         2   516590.
```

Let's see the average price of houses with a garage as opposed to without a garage

```
per_garage<- house_prices %>% group_by(Garage) %>% summarise(avg_price =
mean(Price)) %>%
  arrange(desc(avg_price))
```

Modeling

We will now move onto the modeling portion of this report. We will first drop the columns that are no longer relevant to our analysis. Next, we will split the data into our training and our testing set. Our testing set will consist of 40% of the entire house pricing data set. Then, I will perform 7 different models, where I will train the set and then calculate an RMSE based on the test set.

```
# keep and drop relevant columns for our analysis
house_prices<- house_prices %>% select(-c(Location, Condition, Garage, Id))

# split the data into test and training set. we will use 40% of the data as
our test set
set.seed(100)
test_index <- createDataPartition(y = house_prices$Price, times = 1, p = 0.4,
list = FALSE)
train <- house_prices[-test_index,]
test <- house_prices[test_index,]
```

Model 1: Baseline Model The baseline model is our simplest model, where the housing price prediction will be the average house price.

```r
## model 1: baseline model, average price
mu<- mean(train$Price)

model1_rmse <- RMSE(test$Price, mu)

rmse_table <- data.frame(
  Model = c("Model 1"),
  RMSE = round(model1_rmse, 5)
)
rmse_table

##     Model     RMSE
## 1 Model 1 276534.2
```

Model 2: Location Effect The location effect model takes into account the effect that a location (ie. Downtown, Rural, Urban, Suburban) has on a house's price.

```r
## model 2: location effect
location_effect <- train %>%
  group_by(location_encoded) %>%
  summarize(location_effect_val = mean(Price - mu) )

test<- left_join(test, location_effect, by = "location_encoded")

model2_rmse<- RMSE(test$Price, mu + test$location_effect_val)

rmse_table<- rmse_table %>% rbind(c("Model 2", round(model2_rmse,5)))
rmse_table

##     Model         RMSE
## 1 Model 1 276534.16478
## 2 Model 2 276153.43161
```

Model 3: Regularized Location Effect The regularized location effect model builds on model 2 but incorporates regularization to account for the fact that overfitting may occur.

```r
## model 3: regularized location effect
location_effect_reg<- function(alpha){
  location_effect <- train %>%
    group_by(location_encoded) %>%
    summarize(location_effect_val_reg = sum(Price - mu)/(n() + alpha))

  temp<- left_join(test, location_effect, by = "location_encoded")

  model3_rmse<- RMSE(temp$Price, mu + temp$location_effect_val_reg)
  return(model3_rmse)
}
```

```r
alphas<- seq(0, 50, by = 1)
reg_results<- sapply(alphas, location_effect_reg)
best_alpha<- alphas[which.min(reg_results)]
model3_rmse<- reg_results[which.min(reg_results)]

rmse_table<- rmse_table %>% rbind(c("Model 3", round(model3_rmse,5)))
rmse_table
```

```
##     Model         RMSE
## 1 Model 1 276534.16478
## 2 Model 2 276153.43161
## 3 Model 3 276153.43161
```

Model 4: Location and Bedroom Effect The location and bedroom effect model takes into account both the effect that a location (ie. Downtown, Rural, Urban, Suburban) and that the number of bedrooms has on a house's price.

```r
## model 4: location and bedroom effect
bed_effect <- train %>% left_join(location_effect, by = "location_encoded")
%>%
group_by(Bedrooms) %>%
  summarize(bed_effect_val = mean(Price - mu - location_effect_val))

test<- left_join(test, bed_effect, by = "Bedrooms")

model4_rmse<- RMSE(test$Price, mu +test$location_effect_val+
test$bed_effect_val)

rmse_table<- rmse_table %>% rbind(c("Model 4", round(model4_rmse,5)))
rmse_table
```

```
##     Model         RMSE
## 1 Model 1 276534.16478
## 2 Model 2 276153.43161
## 3 Model 3 276153.43161
## 4 Model 4    276594.97
```

Let's switch it up and try some linear regression models with the lm function in R.

Model 6: Linear Regression with only non-categorical variables. For simplicity, this model is a linear regression that takes into account the numerical columns in our dataset: area, number of bedrooms. number of bathrooms, number of floors, and year built. I will output both the RMSE and the R-squared.

```r
## model 6: linear regression with only non-categorical variables
fit<- lm(Price ~ Area + Bedrooms + Bathrooms + Floors + YearBuilt, data =
train)
fit_summary<- summary(fit)
```

```
# predict on test data
p_hat <- predict(fit, newdata = house_prices$test)
model6_rmse<- RMSE(p_hat, test$Price)

## Warning in pred - obs: longer object length is not a multiple of shorter
object length

fit_summary$r.squared

## [1] 0.007115854

rmse_table<- rmse_table %>% rbind(c("Model 6", round(model6_rmse,5)))
rmse_table

##     Model         RMSE
## 1 Model 1 276534.16478
## 2 Model 2 276153.43161
## 3 Model 3 276153.43161
## 4 Model 4    276594.97
## 5 Model 6 278036.05172
```

Model 7: Linear Regression with all variables. The last model I will perform is a linear regression that incorporates all variables in the dataset. Once again, I will output both the RMSE and the R-squared.

```
### model 7: linear regression with all variables
fit<- lm(Price ~., data = train)
fit_summary<- summary(fit)

# predict on test data
y_hat <- predict(fit, newdata = house_prices$test)
model7_rmse<- RMSE(y_hat, test$Price)

## Warning in pred - obs: longer object length is not a multiple of shorter
object length

fit_summary$r.squared

## [1] 0.008127687

rmse_table<- rmse_table %>% rbind(c("Model 7", round(model7_rmse,5)))
rmse_table

##     Model         RMSE
## 1 Model 1 276534.16478
## 2 Model 2 276153.43161
## 3 Model 3 276153.43161
## 4 Model 4    276594.97
## 5 Model 6 278036.05172
## 6 Model 7 278263.63622
```

Results Based on the above 7 models, we can see that the model that yields the lowest RMSE is model 2: the location effect with a final RMSE of 276153.43161.

```
#select smallest RMSE
rmse_table_winner<- rmse_table %>% arrange(RMSE) %>% head(1)
rmse_table_winner

##     Model          RMSE
## 1 Model 2 276153.43161
```

Conclusion Compared to the MovieLens project, this does not appear to be a very good RMSE. There are certain limitations that are not allowing us to get a very accurate prediction model, such as the size of the dataset and the fact that there are many other factors that could affect a houses price. For example, the specific city is a large factor, as a 2 bedroom in New York City is going to be much more expensive than a 2 bedroom in Scranton, New York. Future house pricing prediction systems should take that into account in order to get a lower RMSE.

References The dataset I used in this project is the House Price Prediction Dataset from Kaggle, which can be downloaded by going to the following website. https://www.kaggle.com/datasets/zafarali27/house-price-prediction-dataset/data