MGSC310 - Zillow Algorithm Final

Jesse Aseoff, Spencer Carlson, Sindhu Iyengar

8/21/2020

Background

Zillow, the leading online real estate marketplace, is well known for their creation of the "Zestimate" - the companies unique formula to predict home value. The formula is comprised of a multitude of factors including expected variables like location, square footage, number of bedrooms/bathrooms, etc. It also takes into consideration more complex factors like market fluctuation, traffic, and school districts.

Stan Humphries, the Chief Analytics Officer at Zillow said, "So many people visited Zillow on our first day to check out their home values that the website crashed. Up until that point, computer-generated estimates of home values were like the Wizard of Oz – hidden behind a curtain where only mortgage lenders or appraisers could access them."

Over 10 years after the creation of the Zestimate, people are still just as fascinated. A couple of years ago, Zillow released a competition with a $1,000,000 top prize to the team that could create the most accurate "Zestimate". The winning team managed to produce a "Zestimate" model with a lower error rate than the original!

Our Task

Using our knowledge of descriptive and predictive analytics, our team (Alpha) has spent the last few weeks analyzing and modeling our own simplified "Zestimate". Our project utilizes many of the main takeaways from the course including: data cleaning, data visualization, supervised learning (linear regression, decision trees), and unsupervised learning (K-means, PCA).

The Data

The Data we used for the project is the properties_2017.csv file provided from kaggle.com. The dataset is comprised of all the properties in 3 counties (Los Angeles County, Orange County, Ventura County) and their corresponding home features. The raw dataset included 58 variables and nearly 3 million observations.

We decided the best way to maximize accuracy in our models and reduce computational expenses would be to significantly cut down the data set even after cleaning. We modeled from a random sample of 20,000 observations for EACH of the three counties.

The variables we chose to include in our final model are: location, square footage, bedrooms, bathrooms, year built, pool (yes or no), and hot tub or spa (yes or no). We are using these variables to predict the tax assessment value (taxvaluedollarcnt) of properties, because market value is not provided in the data set and there is no one fits all formula to convert any of the variables that are included. The tax assessment value is typically similar to the market value of a property.

Data Cleaning

All of the variables are numeric and integer values. Some of the categorical variables use a numbering system (referenced in a data key) for the different levels.

Establish y variable and delete NA's from predictor variable column:

```
data_cleaner = data_2017_raw[!is.na(data_2017_raw$taxvaluedollarcnt),]
```

This line of code removed about 3,000 observations from the data set. These removed values were NA's in the tax value dollar count column - our target variable. It would be useless to try and fill in these values during the cleaning process because after all, this is the objective of our project.

Total NAs in data:

```
sum_nas = sum(is.na(data_cleaner))
sum_nas
```

```
## [1] 83410640
```

There are 83 million NAs in the dataset. This means that we need to look further and see if there are any columns we can remove.

NAs per column:

We used a for loop (in back end only) to iterate through each variable and return the number of NAs found. There are 29 columns with over 1.5 million NAs - a cause for concern considering that we are working with a dataset of just under 3 million. Many columns also appear to be repetitive - meaning that there are other variables in the data set that provide similar or identical value. One assumption we will be making is that 0s in the pool and hot tub columns means there is no pool. We believe this to be reasonable because of the way the data is set up.

We've created an in depth spreadsheet to explain the logic behind removing or including each variable in the raw data set (see appendix).

Z-score manipulation:

```
zscores = scale(data_test2)
zscores = as.data.frame(zscores)
```

We standardized a version of the cleaned data set in order to remove outliars. The criteria for an outliar was a z-score greater than 3 or less than -3. We chose 3 standard deviations because statistically (assumed normality) 99.7% of the data falls within 3 standard deviations.

Convert outliars to NA/remove outliars:

```
sum(is.na(data_test2))
```

```
## [1] 130183
```

```
data_clean = na.omit(data_test2)
```

We lost about 130,000 observations from cleaning out the outliars. This is okay in the context of our much larger data set.

Convert longitude and latitude variables:

```
data_clean <- data_clean %>%
mutate(Longitude = longitude / 1000000) %>%
mutate(Latitude = latitude / 1000000)
```

The longitude/latitude variables were originally multiplied by 1,000,000, so we converted to the standard notation in new variables.

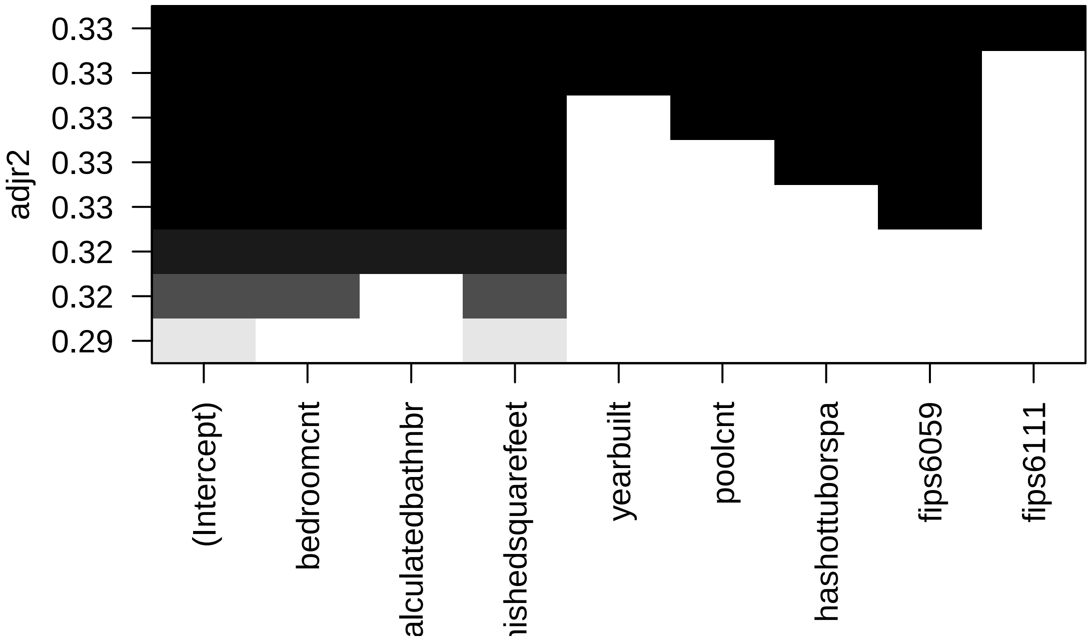
Clean data types:

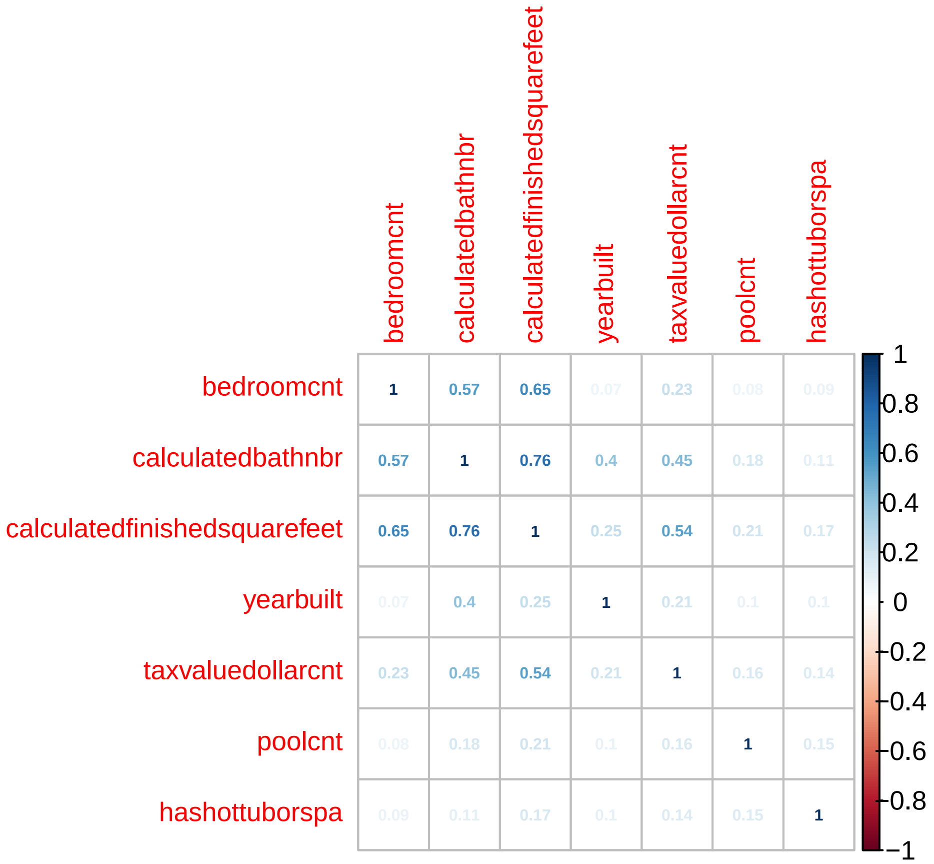
```
data_types = sapply(data_clean, class)
data_types
```

```
##                    bedroomcnt            calculatedbathnbr
##                     "numeric"                    "numeric"
## calculatedfinishedsquarefeet                    yearbuilt
##                     "numeric"                    "numeric"
##             taxvaluedollarcnt                      poolcnt
##                     "numeric"                    "numeric"
##                hashottuborspa                         fips
##                     "numeric"                     "factor"
##                     Longitude                     Latitude
##                     "numeric"                    "numeric"
```

These are the variables and their accompanying class which will be used in our modeling process.

Linear Relationship Visualizations:





	bedroomcnt	calculatedbathnbr	calculatedfinishedsquarefeet	yearbuilt	taxvaluedollarcnt	poolcnt	hashottuborspa
bedroomcnt	1	0.57	0.65	0.07	0.23	0.08	0.09
calculatedbathnbr	0.57	1	0.76	0.4	0.45	0.18	0.11
calculatedfinishedsquarefeet	0.65	0.76	1	0.25	0.54	0.21	0.17
yearbuilt	0.07	0.4	0.25	1	0.21	0.1	0.1
taxvaluedollarcnt	0.23	0.45	0.54	0.21	1	0.16	0.14
poolcnt	0.08	0.18	0.21	0.1	0.16	1	0.15
hashottuborspa	0.09	0.11	0.17	0.1	0.14	0.15	1

The correlation matrix and regsubsets plot help give us a sense of what's driving tax assessment value from a linear standpoint. We can immediately see that square footage is driving this initial linear model (makes sense from logical standpoint). Since location hasn't been taken into account yet, our relatively low r squared value of 0.33 is justifiable. We can reasonably predict that the incorporation of location will drastically improve our error rate.

Split and sample data:

```r
#CREATE 3 datasets for clustering

split_up_dataframe = split(data_clean, data_clean$fips)

#these are the three clean datasets (full - not sampled yet)
data_LA = as.data.frame(split_up_dataframe[1])
data_OC = as.data.frame(split_up_dataframe[2])
data_VC = as.data.frame(split_up_dataframe[3])

#samples created for each county
data_LA_sample = data_LA[sample(nrow(data_LA), 20000), ]
data_OC_sample = data_OC[sample(nrow(data_OC), 20000), ]
data_VC_sample = data_VC[sample(nrow(data_VC), 20000), ]

#remove the fips variable
data_LA_sample = data_LA_sample[,-8]
data_OC_sample = data_OC_sample[,-8]
data_VC_sample = data_VC_sample[,-8]
```

The above code outlines our process of splitting the data set based on fips code (the 3 different fips codes represent the 3 counties seen in the data - LA, Orange, Ventura). After splitting, we then cut each down randomly to 20,000 observations to allow for more sophisticated modeling without the computational expenses of a much larger set.

Although we will not do it in this markdown file, we wrote our 3 county sample data sets in order to use for county specific modeling.