Term Project

Housing Affordability Data System

Name: Dasun Wellawalage

Course: DSC 630

Working Method: Independent

Dataset: American Housing Survey: Housing Affordability Data System

Data Source: https://www.huduser.gov/portal/datasets/hads/hads.html

Data Origin: USA Federal Government

Executive Summary

Housing affordability is one of the biggest problems an average American has to solve during his lifetime. In fact, for most people, it accounts for the biggest recurring payment made every month whether it's a rented property or owned. The Housing Affordability Data System (HADS) is a set of housing-unit level datasets that measures the affordability of housing units and the housing cost burdens of households, relative to area median incomes, poverty level incomes, and Fair Market Rents.

This project aims to model the amount of housing cost burden, a family would be willing to take based on their household income, age of the head of household, monthly expenses and many other factors. Interestingly, we found out that the housing burden, a family is willing to take, is not directly proportional to the family income but rather based on other factors as well. This analysis was conducted on a dataset that includes more than 50000 housing data records from 1985 to 2009. As a result, any changes in the housing market over the last decade will not be reflected here, although most of the factors would have remained the same.

Two different algorithms were used for testing. Initial testing using didn't yield good results but the subsequent testing using a different method gave us great results with a very high accuracy level. The results were conclusive enough to determine that the household income, housing cost, and the age of the head of household data is enough to build a model that can accurately predict the housing cost burden a family is likely to bear.

Background

The Housing Affordability Data System (HADS) is a set of housing-unit level datasets that measures the affordability of housing units and the housing cost burdens of households, relative to area median incomes, poverty level incomes, and Fair Market Rents. This data set provides housing analysts with consistent measures of affordability and burdens over a long period. The dataset is based on the American Housing Survey (AHS) national files from 1985 through 2009 and the metropolitan files from 2002 through 2009. The purpose of this project is to help families make more informed decisions about their future home based on their income and other relevant factors.

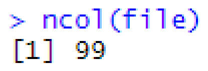
Problem Statement

Housing affordability is one of the biggest problems an average American has to solve during his lifetime. In fact, for most people, it accounts for the biggest payment made every month whether it's a rented property or owned. This is one of the key considerations when people to decide live in a particular state or an area. Depending on the income and other personal requirements one might choose to live in a slightly more expensive area whereas another person might choose much less expensive area and accept inconveniences such a long commute or not having easy access to essential services. This analysis aims to figure out the housing cost you can or should afford based on your income, expenses, and other specific factors. I will be using standard statistical methods such as regression and predictive analysis to derive useful insight from this dataset.

Methods

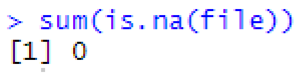
All the computations were performed using a Jupyter Notebook on a 16 core Dell Server.

Checking the number of variables available in the dataset.



```
> ncol(file)
[1] 99
```

Checking for missing values



```
> sum(is.na(file))
[1] 0
```

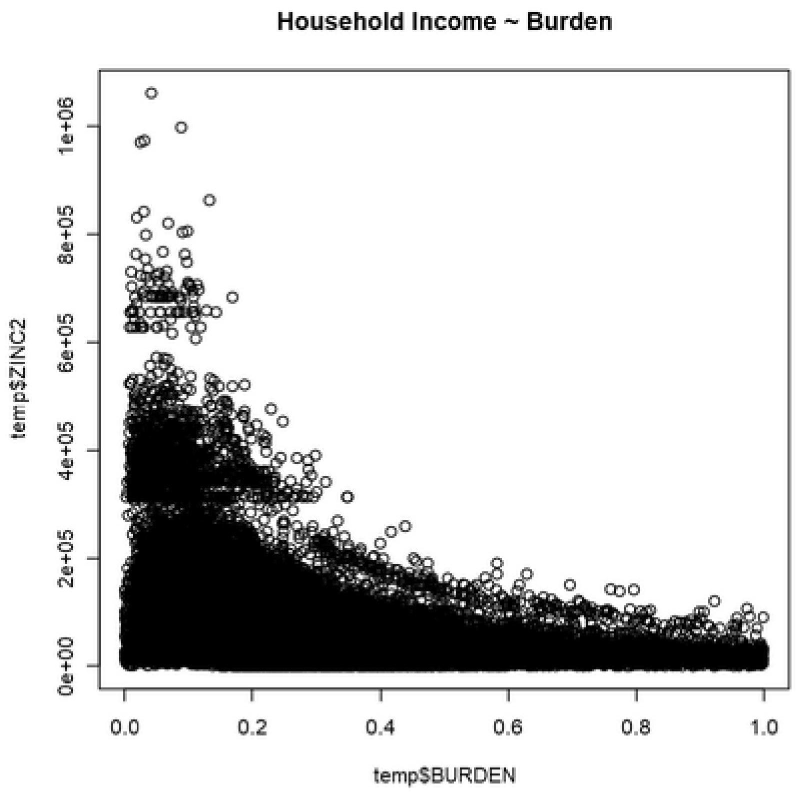
Although there are 99 variables in the dataset, most of them are somewhat similar cost values. Here are the descriptions for some of the important variables.

ZINC2	Household Income
ZSMHC	Monthly housing costs
TOTSAL	Total Wage Income
LMED	Area median income (average)
GLMED	Growth-adjusted median income
AGE1	Age of head of household
APLMED	Median Income Adjusted for # of Persons
TENURE	Owner/renter status of unit
TYPE	Structure Type
UTILITY	Monthly utility cost

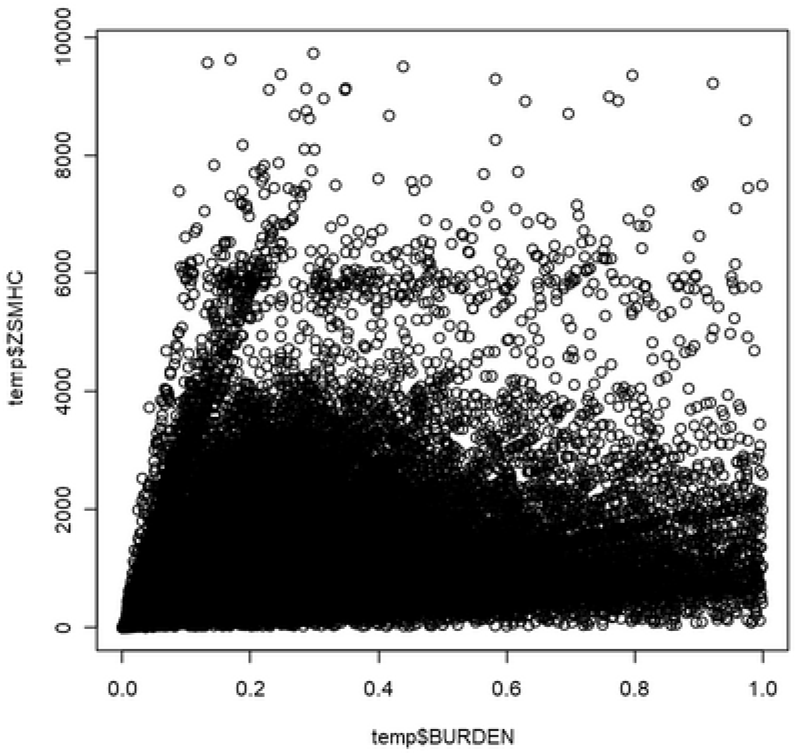
OTHERCOST	Insurance, condo, land rent, other mobile home fees
BEDRMS	# of bedrooms in unit
BUILT	Year unit was built
BURDEN	Housing cost as a fraction of income

Checking for linear relationships between BURDEN and a few other variables that are most likely to have an impact.

```
temp <- data %>% filter(BURDEN > 0 & BURDEN < 1)
scatter.smooth(x=temp$BURDEN, y=temp$ZINC2, main="Household Income ~ Burden")
```

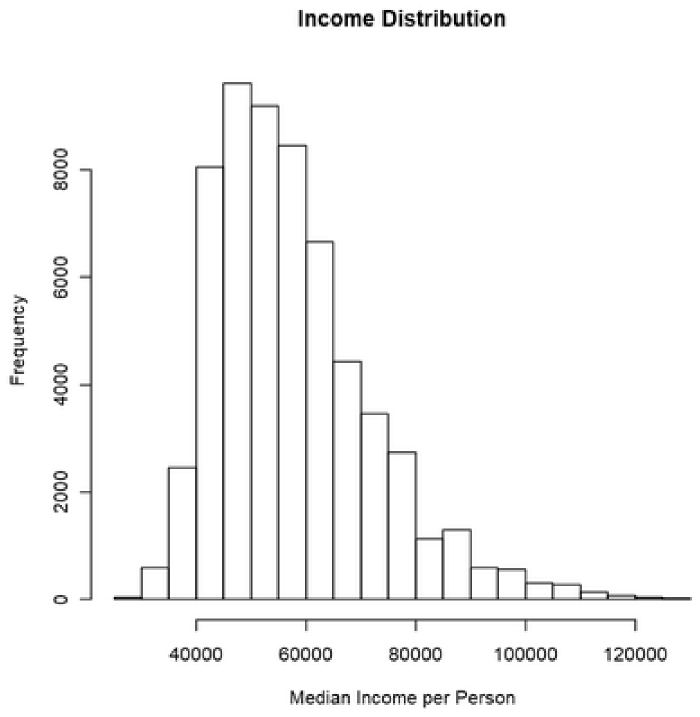


Here is the result for Housing cost vs Burden.

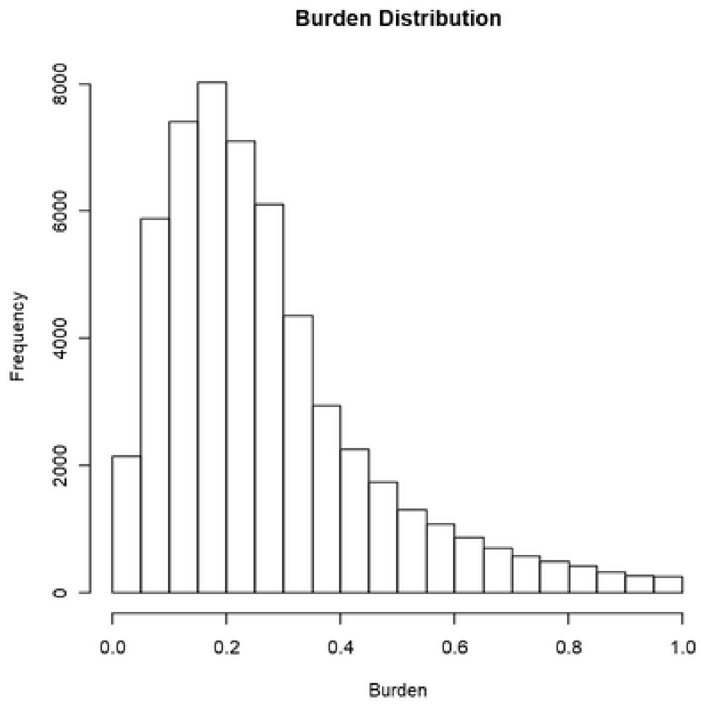


As you can see none of these seem to have a linear relationship with BURDEN. However, if we check the distribution pattern for these variables, all of them seem to be positively skewed with an almost identical distribution pattern.

```r
temp <- data %>% filter(25000 < APLMED) %>% filter(APLMED < 130000)
hist(temp$APLMED, main="Income Distribution", xlab="Median Income per Person")
```



```r
temp <- data %>% filter(BURDEN > 0 & BURDEN < 1)
hist(temp$BURDEN, main="Burden Distribution", xlab="Burden")
```



We are using the CARET package to train a model to predict BURDEN based on the household income, monthly housing cost, and head of household age.

Selecting a subset of data for our training purposes.

```r
tempdata <- data %>% filter(BURDEN > 0 & BURDEN < 1) %>% filter(AGE1 > 0)

var <- c("ZINC2", "ZSMHC", "AGE1")
temp <- tempdata[var]

temp$BURDEN = round(tempdata$BURDEN * 100, 0)
head(temp)
```

```
A data.frame: 6 × 4

    ZINC2  ZSMHC  AGE1  BURDEN
    <int>  <int>  <int>  <dbl>
1   18021    533    82      35
2  122961    487    50       5
3   27974   1405    53      60
4   32220    279    67      10
5   96874    759    26       9
6   14987    695    56      56
```

Since none of the variables showed any useful relationship with BURDEN, after omitting similar variables, I selected ZINC2, ZSMHC, and AGE1 variables arbitrarily which seemed to have a potential impact on BURDEN.

For the initial effort, I decided to use a K Nearest Neighbor method. Here is the related code. Creating the training vs testing data split as 80% and 20% respectively.

```r
TrainSet <- createDataPartition(y = temp$BURDEN,p = 0.8,list = FALSE)
training <- temp[TrainSet,]
testing <- temp[-TrainSet,]
```

Creating the “traincontrol”

```r
set.seed(200)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
```

The function “traincontrol” generates parameters that further control how models are created. It is used for controlling training parameters like resampling, number of folds, iteration etc. The user can change the metric used to determine the best settings. By default, RMSE, R^2, and the mean absolute error (MAE) are computed for regression while accuracy and Kappa are computed for classification. If none of these parameters are satisfactory, the user can also compute custom performance metrics. The “trainControl” function has an argument called “summaryFunction” that specifies a function for computing performance.

Following this, I tried a Random Forest model without Pre-processing, first.

```r
rfFit <- train(BURDEN ~ ., data = training, method = "rf", trControl = ctrl, tuneLength = 20)
```

In CARET package, data transformations can be used either as a standalone transform or during the training. In my case I used the “preprocess” function with the “train” function. In my next attempt, I tried the same Random Forest model with pre-processing enabled.

```r
rfFit <- train(BURDEN ~ ., data = training, method = "rf", trControl = ctrl, preProcess = c("center","scale"),
               tuneLength = 20)
```

I used “center” and “scale” transforms specifically. “center” transform calculates the mean for an attribute and subtracts it from each value while “scale” transform calculates the standard deviation for an attribute and divides each value by that standard deviation.

Finally, I tried a Random Forest model with the same parameters but with extra variables.

```
var <- c("ZINC2", "ZSMHC", "AGE1", "TENURE", "TYPE", "UTILITY", "OTHERCOST")
temp <- tempdata[var]
temp$BURDEN = round(tempdata$BURDEN * 100, 0)
```

For better performance, parallel processing was enabled before running the ML models. I was using a Dell server with two 8 core sockets. I decided to use 14 cores out of the 16 available for the model training.

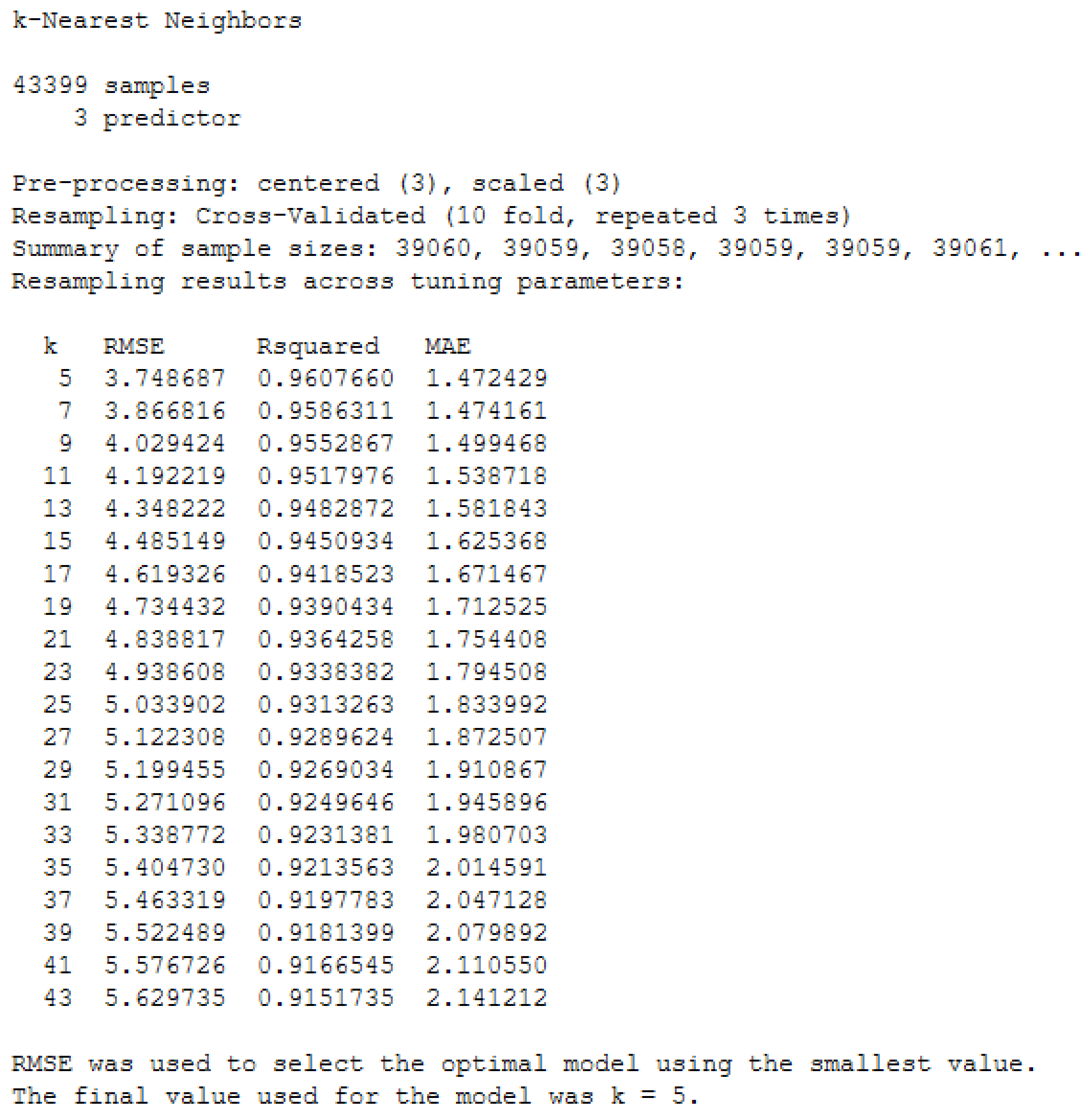
```
library(doParallel)
cl <- makePSOCKcluster(14)
registerDoParallel(cl)
```

```
Loading required package: foreach

Loading required package: iterators

Loading required package: parallel
```

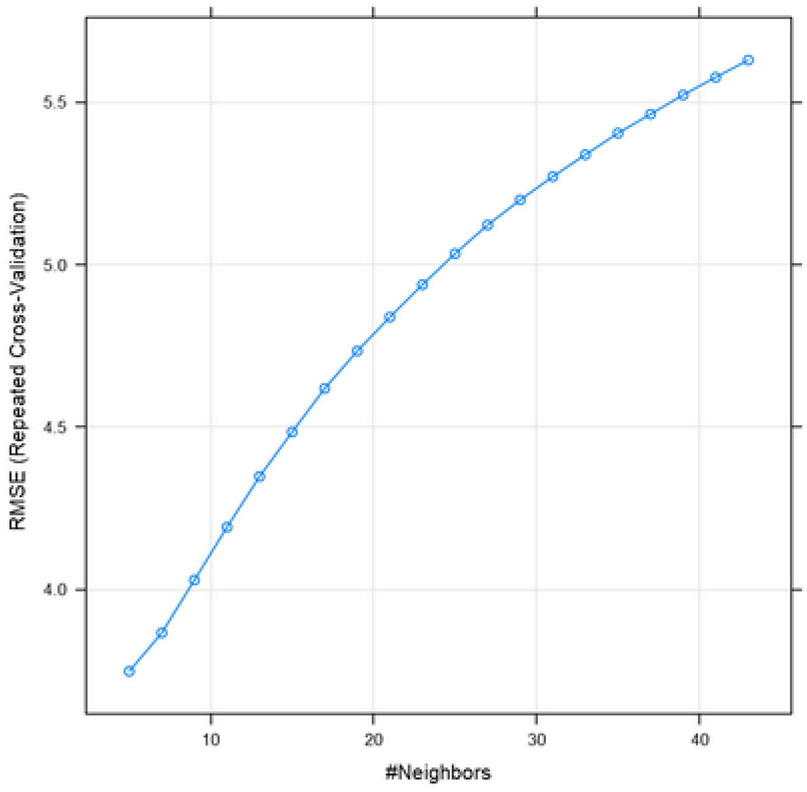
Let’s explore the results of these models in the next section.

Results

k-Nearest Neighbors with 3 predictors

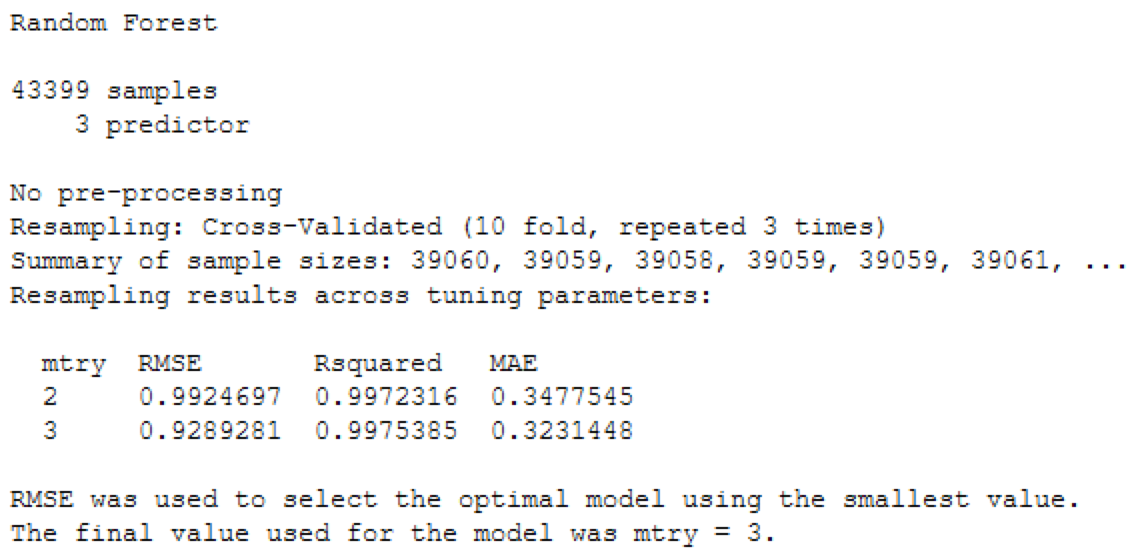


```
k-Nearest Neighbors

43399 samples
    3 predictor

Pre-processing: centered (3), scaled (3)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 39060, 39059, 39058, 39059, 39059, 39061, ...
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   5  3.748687  0.9607660  1.472429
   7  3.866816  0.9586311  1.474161
   9  4.029424  0.9552867  1.499468
  11  4.192219  0.9517976  1.538718
  13  4.348222  0.9482872  1.581843
  15  4.485149  0.9450934  1.625368
  17  4.619326  0.9418523  1.671467
  19  4.734432  0.9390434  1.712525
  21  4.838817  0.9364258  1.754408
  23  4.938608  0.9338382  1.794508
  25  5.033902  0.9313263  1.833992
  27  5.122308  0.9289624  1.872507
  29  5.199455  0.9269034  1.910867
  31  5.271096  0.9249646  1.945896
  33  5.338772  0.9231381  1.980703
  35  5.404730  0.9213563  2.014591
  37  5.463319  0.9197783  2.047128
  39  5.522489  0.9181399  2.079892
  41  5.576726  0.9166545  2.110550
  43  5.629735  0.9151735  2.141212

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.
```



As you can see from the above plot and the RMSE values, the results are best at the K value of 5 and it worsens as the K value increases. This K value corresponds to the number of neighbors to be considered in the calculation when predicting new values. Although the R squared (variance of dependent variable explained by the independent variable) is 96%, this model still doesn't seem to improve with the addition of neighbors. Even at best there is a 3.7 error margin considering BURDEN has been transformed to a value in 0 to 100 scale. Next, let's check a Random Forest model.

Random Forest with 3 predictors (without pre-processing)

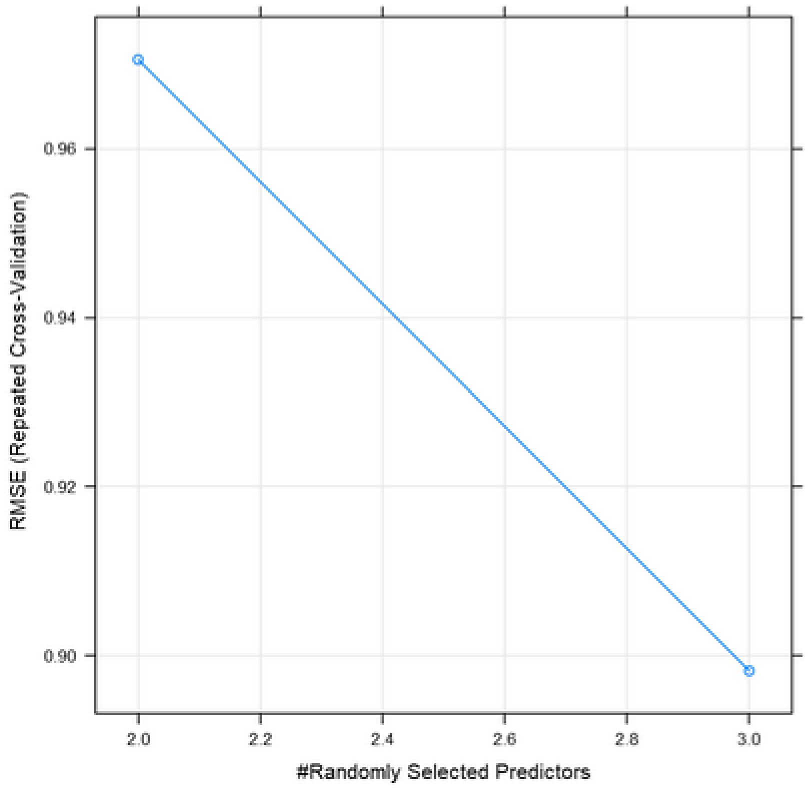


```
Random Forest

43399 samples
    3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 39060, 39059, 39058, 39059, 39059, 39061, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  2     0.9924697  0.9972316  0.3477545
  3     0.9289281  0.9975385  0.3231448

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
```

This model shows a clear improvement over the KNN model. The model has picked 3 as the optimal number of variables to randomly sample as candidates at each split. Using 3 variables it has achieved an error margin (RMSE) of 0.92 with a 99% R squared value. This is a significant improvement from KNN. However, I specifically chose to omit pre-processing here. Transformations applied during the pre-processing stage can have huge impact on the accuracy of the final models. Next, I ran the same Random Forest model with the same dataset and the same number of variables, but with 2 transformation methods (centering & scaling) to pre-process data.

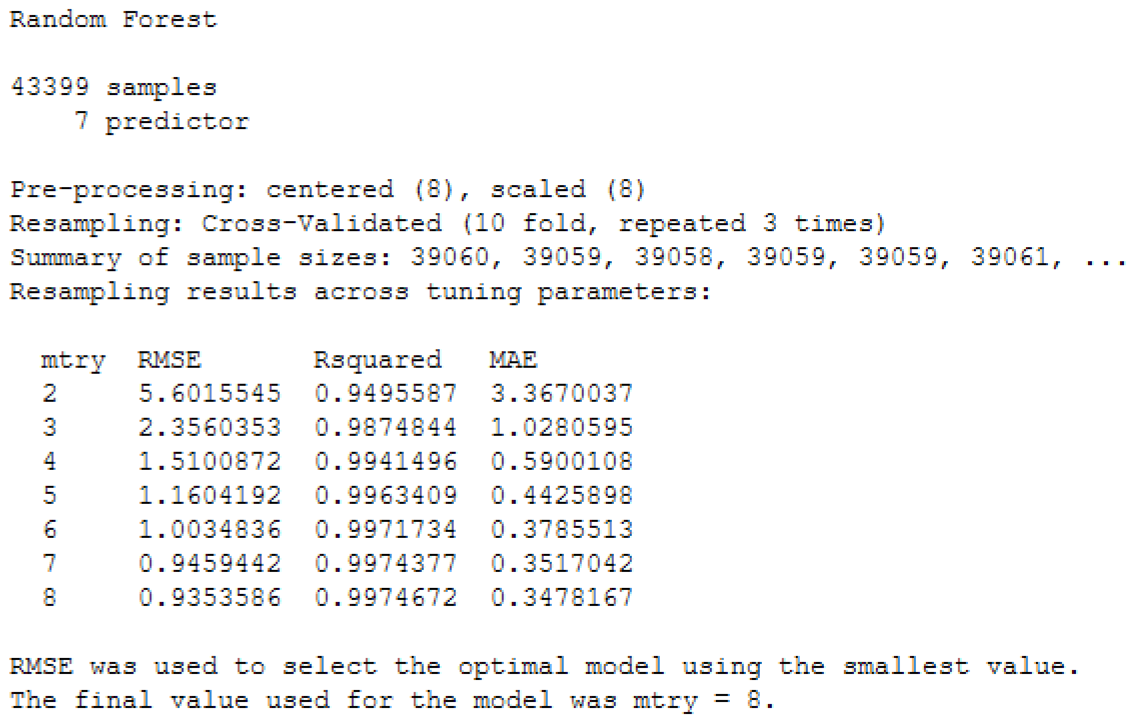
Random Forest with 3 predictors (with pre-processing)

```
Random Forest

43399 samples
    3 predictor

Pre-processing: centered (3), scaled (3)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 39060, 39059, 39058, 39059, 39059, 39061, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  2     0.9705719  0.9973463  0.3524440
  3     0.8981926  0.9976822  0.3284541

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
```

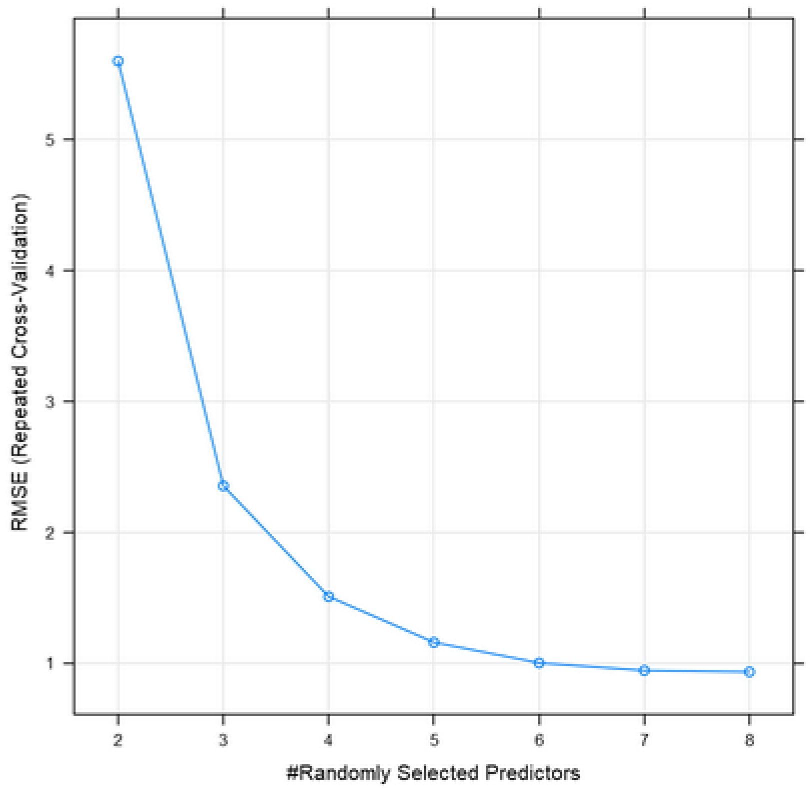


As you can see, we achieved some marginal improvement over our previous model. This new model works best with 3 variables randomly sampled as candidates at each split and it has an RMSE as low as 0.89 and R squared remains at 99%. Finally, I tried the same RF model on the same dataset, but with 4 additional variables which are "TENURE", "TYPE", "UTILITY", and "OTHERCOST". The TENURE means whether the property is owned or rented. TYPE refers to the structural type and the other two are self-explanatory.

Random Forest with 7 predictors (with pre-processing)



```
Random Forest

43399 samples
    7 predictor

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 39060, 39059, 39058, 39059, 39059, 39061, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  2     5.6015545  0.9495587  3.3670037
  3     2.3560353  0.9874844  1.0280595
  4     1.5100872  0.9941496  0.5900108
  5     1.1604192  0.9963409  0.4425898
  6     1.0034836  0.9971734  0.3785513
  7     0.9459442  0.9974377  0.3517042
  8     0.9353586  0.9974672  0.3478167

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 8.
```



This model works best with 8 variables randomly sampled as candidates at each split and it has an RMSE of 0.93 and R squared remains at 99%.

Conclusion

As we can see, adding more variables to the model didn't really improve the results. Random Forest models clearly had a higher accuracy level over the KNN model. One area for improvement would be the pre-processing stage. I tried centering and scaling but there are many other transformation methods that can be tested. Another area for improvement would be the variable selection. Although increasing the number of independent variables didn't produce better results, we can still try different combinations of variables, or even a derived variable if that helps improve the accuracy of the model. Overall, the Random Forest model with 3 independent variables and pre-processing enabled, produced the best results.

Acknowledgement

For the past 12 weeks, it has been a challenging and interesting journey through the predictive analysis course. This project is the combined result of learning outcome by all the readings and assignments completed throughout this course. I take this opportunity to thank our professor Ashley Kern for carefully preparing this challenging and informative syllabus while keeping it interesting. She continued to support us through out the program whenever we were in doubt. I would also like to thank the users who dedicate their personal time to answer questions published on sites like Kaggle, and RPubs, which have been an immense help when looking for answers that seem impossible to find without some extra help.

Finally, I would like to thank my almost 40 weeks pregnant wife who were patient enough while I spent a lot of time working on my assignments and projects. This wouldn't have been possible without her patience and support.

References

Caret Package (Mar 2019). Parallel Processing. Retrieved May 26, 2020, from https://topepo.github.io/caret/visualizations.html

Caret Package (Mar 2019). Model Training and Tuning. Retrieved May 26, 2020, from https://topepo.github.io/caret/model-training-and-tuning.html#control

Kaggle (2017). KNN with Caret. Retrieved May 26, 2020, from https://www.kaggle.com/rumasinha/knn-with-caret

Kaggle (2017). Picking the best model with Caret. Retrieved May 26, 2020, from https://www.kaggle.com/rtatman/picking-the-best-model-with-caret

Machine Learning Mastery (Sep 2014). Tuning Machine Learning Models Using the Caret R Package. Retrieved May 26, 2020, from https://machinelearningmastery.com/tuning-machine-learning-models-using-the-caret-r-package/

Machine Learning Mastery (Feb 2016). Get Your Data Ready For Machine Learning in R with Pre-Processing. Retrieved May 26, 2020, from https://machinelearningmastery.com/pre-process-your-dataset-in-r/

RPubs (May 2018). Caret Practice. Retrieved May 26, 2020, from https://rpubs.com/phamdinhkhanh/389752

RPubs (Apr 2014). kNN Using caret R package. Retrieved May 26, 2020, from https://rpubs.com/njvijay/16444