HarvardX: PH125.9x Final Project

By Philip J Brown, pjbMit@pjb3.com

6/16/2019

HarvardX Data Science Capstone Class PH125.9x (2T2018)

- Student: Philip Brown
- email: Phil@pjb3.com
- github: https://github.com/pjbMit

RealML - *A Real Estate Machine Learning Project*

This is RealML, a real estate machine learning project and report created by Philip J Brown (pjbMit@pjb3.com) as the final capstone project for the Data Science Certificate program offered by HarvardX: PH125.9x from edx.org.

Part 1) EXECUTIVE SUMMARY

The goal of this project is to utilize data analysis and modelling skills to a create machine learning engine and this report as the final exercise in completing the 9 course Data Science Certificate program offered by HarvardX through edx.org.

For my project I chose to use machine learning techniques to build a *RealML,a real estate sales price prediction engine.* More specifically, I wanted to answer this question:

> **Can I reasonably predict the resale price of residential condominum and single family real estate within a five mile radius of *Fairlington Villages (link)*, the condominum development in Arlington Virginia that I call home?**

Through this project, I am able to demonstrate examples of data identificaton and acquisition, data wrangling and cleansing, data analysis, modeling and machine learning techniques, data presentation and data visualization and report generation and presentation. The project was built by acquiring and analyzing more than 20,000 reports of current real estate sales within the stated five mile radius for residential properties that sold for at least $5,000 but less than $1,000,000.

After some research I was able to locate and curate live data for this project, so the basis for this report is real and impactful – at least it is to me as home owner in this area. * :-)

The results of this analysis were very encouraging, and are included in the **results** section and the **conclusion** section, which are the last two sections in this report.

The mission was to locate, curate, wrangle and cleanse real data, and use it build a prediction engine that tries to predict sales prices so as to optimize the model for a low Root Mean Square Error (RMSE), defined as

$$RMSE = \sqrt{\Sigma_{i=1}^{n} \frac{(actual_i - predicted_i)^2}{N}}$$

This project is intended to highlight some of the skills acquired throughout the courses in this program. All programming was done in R Code using RStudio on MacBook Pro. After acquiring and processing the data, the real work began!

In addition to this **executive summary**, this report also includes a **methods and analysis section**, a **results section** and a **conclusion section**.

Key files for this project have been uploaded and stored on my git hub page at github.com/pjbMit/real_estate_project. The three main files for this project are listed below, and can be viewed on git hub – The file names are also links:

- real_ml_script.R *(link)*
- real_ml_report.Rmd *(link)*
- real_ml_report.pdf *(link)*

Additionally, a gzip'd version of the data file is on github at:

- realml_data_file.github.json.gz *(link)*

Part 2) METHODS AND ANALYSIS

The project was created in the RStudio environment using Rstudio Version 1.1.442 on a Macintosh; Intel Mac OS X 10_14_5

R version 3.5.1 (2018-07-02)
nickname Feather Spray

All code was written in R and executed in RStudio.

Here are the methods and techniques used.

Data was downloaded from AttomData.com, a commercial data provider, using their RESTful API and an apikey that is needed in order to get data. Sales data was queried from their API, and results were downloaded 10,000 rows at a time.

The data was then filtered to remove property types that aren't residential condos or homes.

Part 3) RESULTS

After downloaing the raw data, we had sales information covering the following sales date range:

```
factorData %>% summarize(newest_sale=max(saledate),oldest_sale=min(saledate))
```

```
##   newest_sale oldest_sale
## 1  2019-05-06  2012-01-03
```

We found additional filter criteria to help us cleanse the data. For example, shown below is the data after filtering to show just the property types and subtypes of interest, which had the effect of removing commercial sales, industrial sales, and other data that is out of scope for this project.

```
#Show "Resale' for "SFR" and "CONDOMINIUM
factorData %>%
    filter(transtype == "Resale", proptype %in% c("SFR","CONDOMINIUM")) %>%
    group_by(state,propsubtype,proptype) %>%
    summarize(num_sales=n()) %>%
    select(state,propsubtype,proptype,num_sales) %>%
    xtabs(num_sales ~ proptype + propsubtype, data=.) %>%
    ftable()
```

```
##             propsubtype HOUSE RESIDENTIAL
## proptype
## CONDOMINIUM               864       11511
## SFR                     10516        5134
```

```
factorData %>%
    summarize(num_sales=n())
```

```
##   num_sales
## 1     28025
```

```
factorData %>% group_by(transtype) %>% summarize(num=n())
```

```
## # A tibble: 1 x 2
##   transtype   num
##   <fct>     <int>
## 1 Resale    28025
```

After examining the data, we saw that a little data-cleansing house keeping was in order. We found that about 10% of the data had zero listed for bedrooms, yet those units had about 1400 sqft on average (mean), thus the zero bedrooms was clearly an error. We removed these rows along with two unneeded colums.

```
#Look for zero bedrooms, and determine their mean sqft.
factorData %>% filter(beds==0) %>% summarize(num=n(),mean(sqft))
```

```
##    num mean(sqft)
## 1 3923   1455.371
```

```
#remove zero bedroom errors, and remove unneeded columns.
myData <- factorData %>% filter(beds != 0) %>%
            select(-transtype,-propsubtype)
```

After cleansing the data, here's a grouped summary showing the data by year and property type:

```
##             year_sold 2012 2013 2014 2015 2016 2017 2018 2019
## proptype
## CONDOMINIUM            970 1305 1303 2429 2341 1852 1897  278
## SFR                   1549 1727 1661 3136 2993 2117 2048  419
```

```
## [1] 28025
```

```
## [1] 2452
```

```
## [1] 2452   13
```

Now it's time to look at some the individual data attributes.

```
##       beds           baths            sqft        yearbuilt
##  Min.   :1.000   Min.   :1.000   Min.   : 510   Min.   :   0
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 900   1st Qu.:1945
##  Median :2.000   Median :2.000   Median :1150   Median :1961
##  Mean   :2.468   Mean   :2.078   Mean   :1328   Mean   :1964
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:1568   3rd Qu.:1986
##  Max.   :8.000   Max.   :7.000   Max.   :5656   Max.   :2017
##
##       lat             lon            saledate              zip
##  Min.   :38.77   Min.   :-77.19   Min.   :2018-06-02   22204  :318
##  1st Qu.:38.84   1st Qu.:-77.12   1st Qu.:2018-07-30   22201  :263
##  Median :38.86   Median :-77.10   Median :2018-10-05   22206  :192
##  Mean   :38.86   Mean   :-77.10   Mean   :2018-10-17   22203  :159
##  3rd Qu.:38.88   3rd Qu.:-77.08   3rd Qu.:2018-12-21   22207  :127
##  Max.   :38.91   Max.   :-77.01   Max.   :2019-05-06   22310  :127
##                                                        (Other):777
##            city      state            proptype
##  Alexandria  : 394   DC:  70   CONDOMINIUM:1055
```

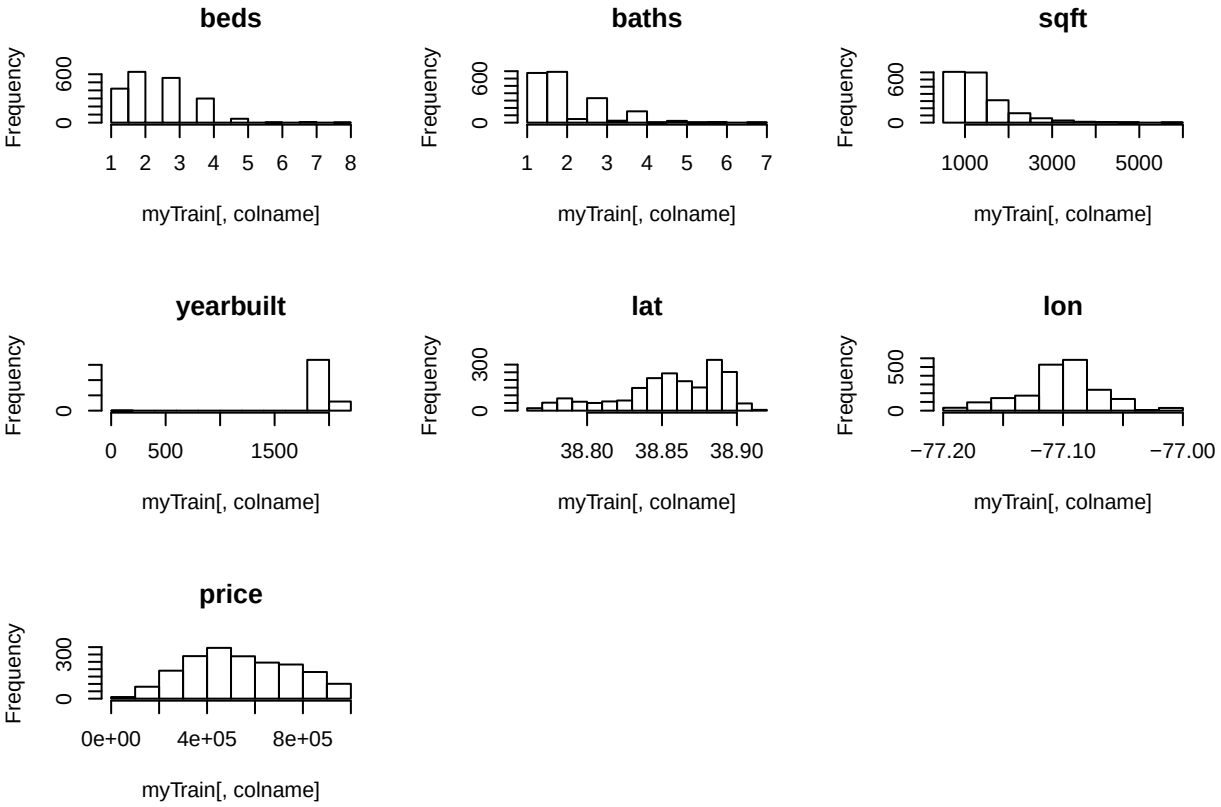
```
##  Annandale   :  31   VA:1893   SFR        : 908
##  Arlington   :1348
##  ARLINGTON   :   0
##  Falls Church: 112
##  Springfield :   8
##  Washington  :  70
##                        addr          price
##  1024 N UTAH ST APT 423  :   2   Min.   :  30000
##  10 HALLEY PL SE APT 303 :   1   1st Qu.: 379000
##  100 N TRENTON ST # 100-1:   1   Median : 525000
##  1000 20TH ST S          :   1   Mean   : 542073
##  1000 26TH ST S          :   1   3rd Qu.: 715000
##  1000 N KENTUCKY ST      :   1   Max.   :1000000
##  (Other)                 :1956
```

```
##      beds     baths      sqft yearbuilt       lat       lon  saledate
## "integer" "numeric" "integer" "integer" "numeric" "numeric"    "Date"
##       zip      city     state  proptype      addr     price
##  "factor"  "factor"  "factor"  "factor"  "factor" "integer"
```

```
##       beds           baths            sqft        yearbuilt
##  Min.   :1.000   Min.   :1.000   Min.   : 510   Min.   :   0
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 900   1st Qu.:1945
##  Median :2.000   Median :2.000   Median :1150   Median :1961
##  Mean   :2.468   Mean   :2.078   Mean   :1328   Mean   :1964
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:1568   3rd Qu.:1986
##  Max.   :8.000   Max.   :7.000   Max.   :5656   Max.   :2017
##
##       lat             lon            saledate               zip
##  Min.   :38.77   Min.   :-77.19   Min.   :2018-06-02   22204  :318
##  1st Qu.:38.84   1st Qu.:-77.12   1st Qu.:2018-07-30   22201  :263
##  Median :38.86   Median :-77.10   Median :2018-10-05   22206  :192
##  Mean   :38.86   Mean   :-77.10   Mean   :2018-10-17   22203  :159
##  3rd Qu.:38.88   3rd Qu.:-77.08   3rd Qu.:2018-12-21   22207  :127
##  Max.   :38.91   Max.   :-77.01   Max.   :2019-05-06   22310  :127
##                                                        (Other):777
##            city      state            proptype
##  Alexandria  : 394   DC:  70   CONDOMINIUM:1055
##  Annandale   :  31   VA:1893   SFR        : 908
##  Arlington   :1348
##  ARLINGTON   :   0
##  Falls Church: 112
##  Springfield :   8
##  Washington  :  70
##                        addr          price
##  1024 N UTAH ST APT 423  :   2   Min.   :  30000
##  10 HALLEY PL SE APT 303 :   1   1st Qu.: 379000
##  100 N TRENTON ST # 100-1:   1   Median : 525000
##  1000 20TH ST S          :   1   Mean   : 542073
##  1000 26TH ST S          :   1   3rd Qu.: 715000
##  1000 N KENTUCKY ST      :   1   Max.   :1000000
##  (Other)                 :1956
```

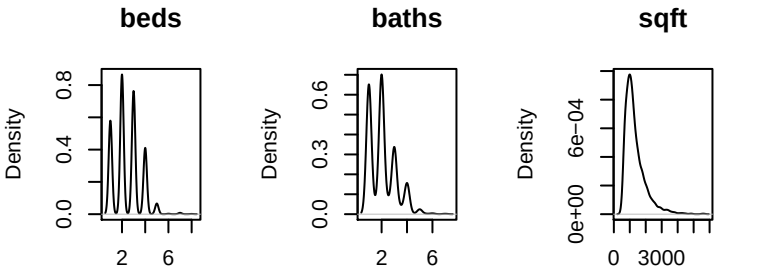
```
##          beds                baths               sqft
## breaks   Numeric,15          Numeric,13          Integer,12
```

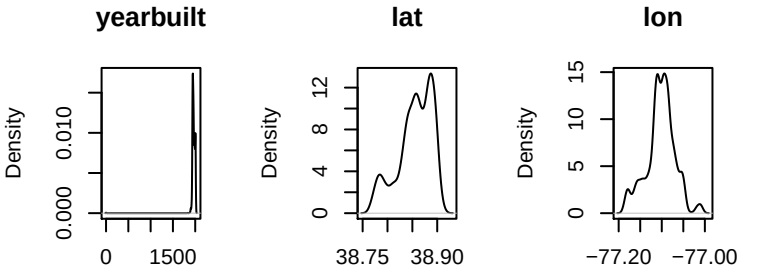
```
## counts   Integer,14           Integer,12           Integer,11
## density  Numeric,14           Numeric,12           Numeric,11
## mids     Numeric,14           Numeric,12           Numeric,11
## xname    "myTrain[, colname]" "myTrain[, colname]" "myTrain[, colname]"
## equidist TRUE                 TRUE                 TRUE
##          yearbuilt            lat                  lon
## breaks   Numeric,12           Numeric,17           Numeric,11
## counts   Integer,11           Integer,16           Integer,10
## density  Numeric,11           Numeric,16           Numeric,10
## mids     Numeric,11           Numeric,16           Numeric,10
## xname    "myTrain[, colname]" "myTrain[, colname]" "myTrain[, colname]"
## equidist TRUE                 TRUE                 TRUE
##          price
## breaks   Numeric,11
## counts   Integer,10
## density  Numeric,10
## mids     Numeric,10
## xname    "myTrain[, colname]"
## equidist TRUE
```



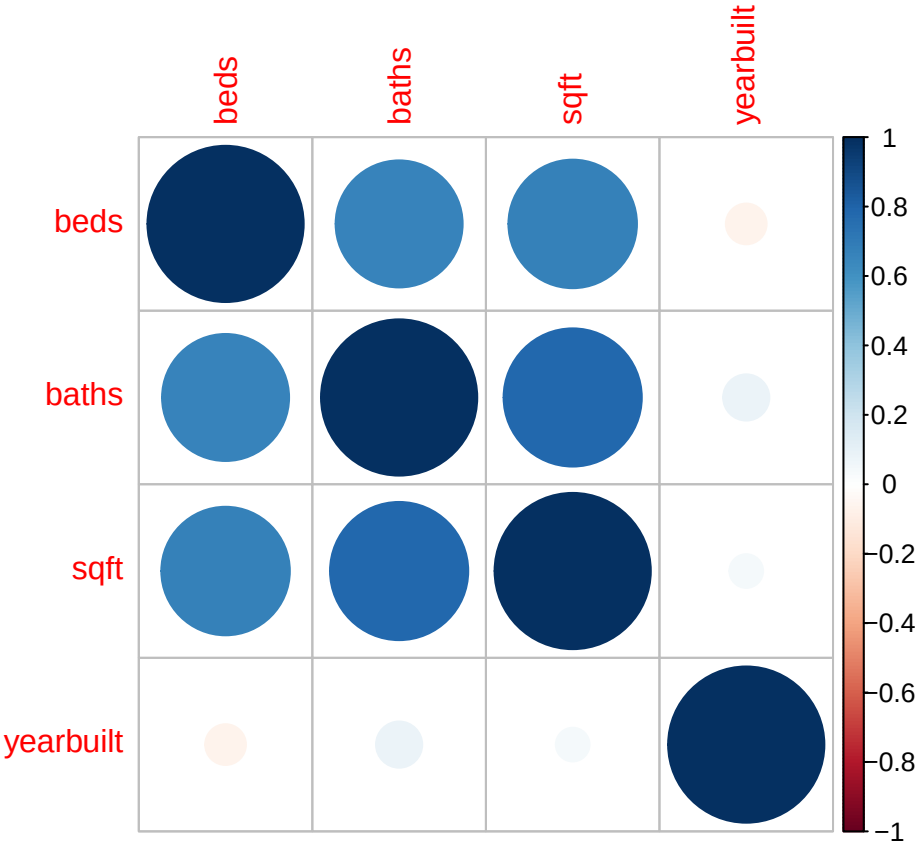
The bar plots are a little jagged. Lets get a different perspective by smoothing out the graphs using a density plot. This helps us better visualize the data to see if we have a binomial distribution, or other anomally.

```
par(mfrow=c(2,3))
for(i in c(1:6)) {
    plot(density(myTrain[,i]), main=names(myTrain)[i])
}
```





Next, we looked the remaining columns to see how they correlated, to see if we can remove any columns that are highly correlated.



The data is highly correlated. We use the calculation below to determine which attribute, if any, we should consider removing to see if our models improve... and the answer turns out to be baths. But for now, we

opted not to remove baths from the model.

```
set.seed(2020)
cutoff <- 0.70
correlations <- cor(myTrain[,c("beds","baths","sqft","yearbuilt")])
highlyCorrelated <- findCorrelation(correlations, cutoff=cutoff)
for (value in highlyCorrelated) {
    print(names(myTrain)[value])
}
```

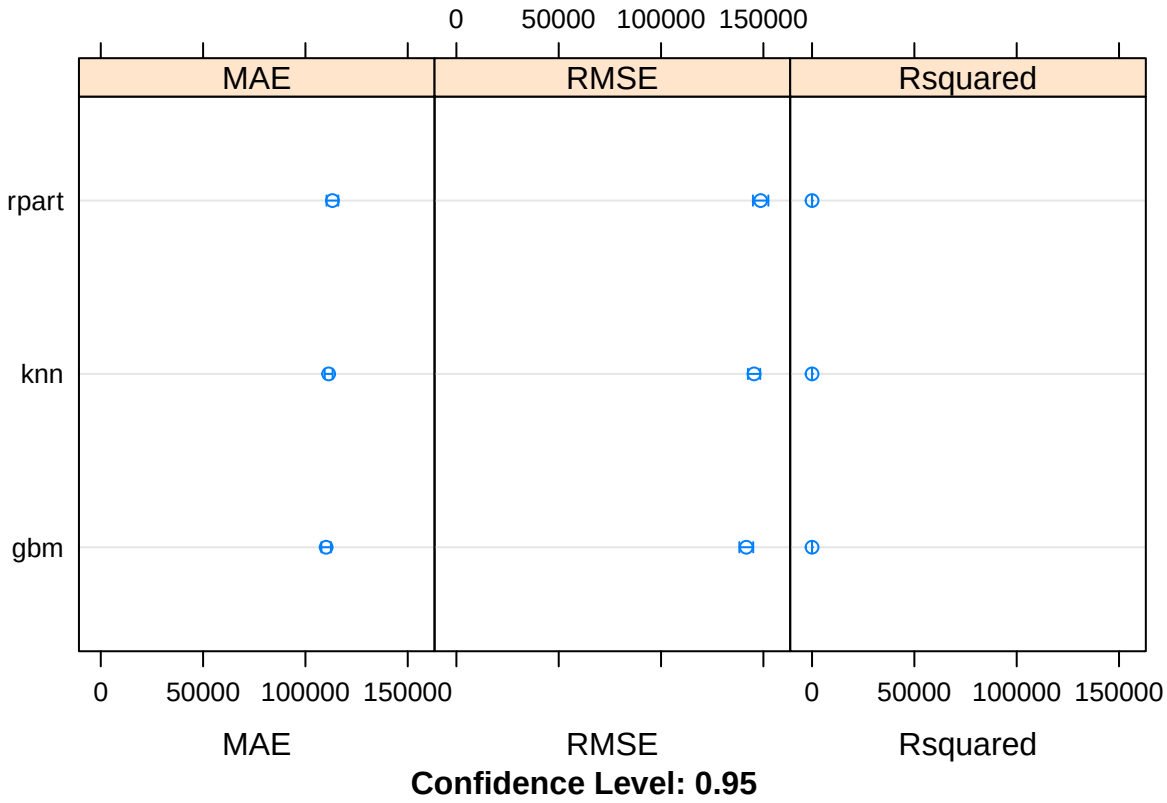
```
## [1] "baths"
```

Modelling

Now it's time to explore our data models. Let's try a mix of non-linear models, and see how we do. Note that we enabled parallel processing in the caret package by loading the **doMC* package and library.

Here are the models from our first run.

```
##
## Call:
## summary.resamples(object = results)
##
## Models: knn, rpart, gbm
## Number of resamples: 30
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## knn   101947.96 106820.6 112511.8 111346.4 114584.1 121547.5    0
## rpart  97243.74 107957.2 113773.1 113198.4 119292.5 128387.7    0
## gbm   100681.44 106503.1 108852.6 110119.5 112562.0 124161.1    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## knn   130482.4 139565.3 145296.0 145443.9 149278.5 165034.8    0
## rpart 125640.6 142110.8 148207.9 148660.1 155899.8 168538.0    0
## gbm   128222.2 135788.6 140608.0 141647.8 144628.3 164723.1    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn   0.4264019 0.5234677 0.5485585 0.5507051 0.5821683 0.6521425    0
## rpart 0.4129235 0.5018945 0.5383389 0.5354760 0.5823529 0.6790664    0
## gbm   0.4167143 0.5477817 0.5763397 0.5712591 0.6080254 0.6602776    0
```



And here are the results of the output. # `{r models_compare1, echo=TRUE, include=TRUE, eval=TRUE, cache=FALSE} #`

TODO * Data was ... * After these models were evaluated, we looked at variability, and attempted to add genre to models using several standard models available through the **caret package** and applied techniques such as cross-validation. While we examined these models, and made multiple attempts to improve the results, none of the techniques tried improved upon the best results that were previously used.

– General approach:

For many approaches, I first tried working on a very small data set, just to get the code working, then I re-ran the on a medium sized data set, and then when I was satisfied, then I processed the full training set.

Similarly, initially I did NOT do full cross-validation, but once the model was built and the code was working, I enabled cross validation and other ML techniques.

Additionally, being sensitive to computation times, I wrote code and used global variables to enable saving daa and objects containing intermediate results as files on the local file system. By changing the values of these logial variable from TRUE to FALSE, or vice-versa, I was able to re-run code without having to repeat some of the more lengthy processing or repeatedly downloading and cleansing the same data.

–set up

Set up libraries and enable multi-core processing for some of the operations used by the caret package. Because I have an 8 core processor, for calcuations that can utilize the parallel processing features, this script runs *__substantially__ faster.

Part 4) CONCLUSION

The model results were promising, as can be seen by the output from rmse_results.

I discovered a library and options to set to enable multi-core parallel processing for some of the algorithms in the **caret** package, and this technique helped tremendously, as I was able to span 8 R-sessions that ran in parallel to process some of the algorithms.

Ultimately, the best results that I obtained were a **RMSE of TODO** which was obtained ffrom the final model. This was deemed satisfactory based on goals and scope of this project. Of course, if you plan to move nearby, please do your own due dilligence before purchasing a home – while I wanted to choose an impactful and relevant project, this project was created primarily for didactic purposes.

(See the output below which shows the best results obtained.)

```
## [1] "Thanks for checking this out!  pjbMit@pjb3.com  :-)"
```