Real Estate Price Prediction Project

HarvardX: PH125.9x Data Science: Capstone: Sudha Kankipati

2020-06-17

Introduction

Real estate price prediction: it solves the problem of predicting house prices for house buyers and house sellers.

A house value is more than location and square footage. An educated party would want to know all aspects that give a house its value. Like age of the house, distance to the nearest MRT station, number of near by convenience stores, its location

We will be applying machine learning techniques that go beyond standard linear regression.

Data Preparation

Install Packages

```r
if(!require(caret))install.packages("caret", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
if(!require(data.table))install.packages("data.table", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```r
if(!require(rattle))install.packages("rattle", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: rattle

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
if(!require(magrittr))install.packages("magrittr", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: magrittr
```

```r
if(!require(Hmisc))install.packages("Hmisc", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: Hmisc

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(Hmisc, quietly=TRUE)
if(!require(reshape))install.packages("reshape", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: reshape

##
## Attaching package: 'reshape'

## The following object is masked from 'package:data.table':
##
##     melt
```

```r
if(!require(arules))install.packages("arules", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: arules

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:reshape':
##
##     expand

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
if(!require(rpart))install.packages("rpart", repos ="http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```r
if(!require(dataMaid)) install.packages("dataMaid", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dataMaid

##
## Attaching package: 'dataMaid'

## The following object is masked from 'package:Hmisc':
##
##     summarize
```

```r
library(dataMaid)
if(!require(ggplot2))install.packages("ggplot2", repos ="http://cran.us.r-project.org")

building <- TRUE
scoring  <- ! building
```

A pre-defined value is used to reset the random seed

so that results are repeatable.

```
crv$seed <- 42
```

Load a dataset from file.

dataset is present at https://www.kaggle.com/quantbruce/real-estate-price-prediction/datasets_88705_204267_Real estate.csv

```
fname         <- "file:///C:/DataScience/datasets_88705_204267_Real estate.csv"
```

Creating Dataset from file

```
crs$dataset <- read.csv(fname,
                        na.strings=c(".", "NA", "", "?"),
                        strip.white=TRUE, encoding="UTF-8")
```

Build the train/validate/test datasets.

nobs=414 train=290 validate=62 test=62

```
set.seed(crv$seed)
```

Number of observations

```
crs$nobs <- nrow(crs$dataset)
crs$nobs
```

```
## [1] 414
```

Creating training set

```
crs$train <- sample(crs$nobs, 0.7*crs$nobs)
```

Creating validation set

```
crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
  crs$validate
```

Creating testing set

```
crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
  crs$test
```

Data Cleaning

- The following variable selections have been noted.
- We ignore transaction date

- we use No as identity

```r
crs$input     <- c("X2.house.age",
                   "X3.distance.to.the.nearest.MRT.station",
                   "X4.number.of.convenience.stores", "X5.latitude",
                   "X6.longitude")

crs$numeric   <- c("X2.house.age",
                   "X3.distance.to.the.nearest.MRT.station",
                   "X4.number.of.convenience.stores", "X5.latitude",
                   "X6.longitude")

crs$categoric <- NULL

crs$target    <- "Y.house.price.of.unit.area"
crs$risk      <- NULL
crs$ident     <- "No"
crs$ignore    <- "X1.transaction.date"
crs$weights   <- NULL
```

Data Exploration

The 'Hmisc' package provides the 'contents ,describe' function. ## Summary of the dataset.

```r
contents(crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)])
```

```
##
## Data frame:crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)]    289 observations and 6 variabl
##
##
##                                         Storage
## X2.house.age                             double
## X3.distance.to.the.nearest.MRT.station   double
## X4.number.of.convenience.stores         integer
## X5.latitude                              double
## X6.longitude                             double
## Y.house.price.of.unit.area               double
```

```r
summary(crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)])
```

```
##   X2.house.age   X3.distance.to.the.nearest.MRT.station
##  Min.   : 0.00   Min.   :  23.38
##  1st Qu.: 9.10   1st Qu.: 289.32
##  Median :15.90   Median : 490.35
##  Mean   :17.46   Mean   :1102.79
##  3rd Qu.:26.80   3rd Qu.:1559.83
##  Max.   :43.80   Max.   :6488.02
##  X4.number.of.convenience.stores  X5.latitude     X6.longitude
##  Min.   : 0.000                  Min.   :24.93   Min.   :121.5
##  1st Qu.: 1.000                  1st Qu.:24.96   1st Qu.:121.5
##  Median : 4.000                  Median :24.97   Median :121.5
##  Mean   : 4.014                  Mean   :24.97   Mean   :121.5
##  3rd Qu.: 6.000                  3rd Qu.:24.98   3rd Qu.:121.5
##  Max.   :10.000                  Max.   :25.01   Max.   :121.6
##  Y.house.price.of.unit.area
##  Min.   :  7.60
##  1st Qu.: 26.50
##  Median : 39.30
```

```
##   Mean   : 37.88
##   3rd Qu.: 46.60
##   Max.   :117.50
```

Generating a description of the dataset.

```
describe(crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)])
```

```
## crs$dataset[crs$train, c(crs$input, crs$risk, crs$target)]
##
##  6  Variables      289  Observations
## ---------------------------------------------------------------------------
## X2.house.age
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0      194        1    17.46    12.74     1.10     3.58
##      .25      .50      .75      .90      .95
##     9.10    15.90    26.80    34.52    37.54
##
## lowest :  0.0  1.1  1.5  1.7  1.9, highest: 39.8 40.1 40.9 42.7 43.8
## ---------------------------------------------------------------------------
## X3.distance.to.the.nearest.MRT.station
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0      198        1     1103     1234    90.46   157.33
##      .25      .50      .75      .90      .95
##   289.32   490.34  1559.83  3079.57  4082.01
##
## lowest :   23.38284   49.66105   56.47425   57.58945   82.88643
## highest: 4527.68700 4605.74900 5512.03800 6306.15300 6488.02100
## ---------------------------------------------------------------------------
## X4.number.of.convenience.stores
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0       11    0.986    4.014    3.449        0        0
##      .25      .50      .75      .90      .95
##        1        4        6        8        9
##
## lowest :  0  1  2  3  4, highest:  6  7  8  9 10
##
## Value          0     1     2     3     4     5     6     7     8     9    10
## Frequency     52    33    16    33    19    44    22    26    18    18     8
## Proportion 0.180 0.114 0.055 0.114 0.066 0.152 0.076 0.090 0.062 0.062 0.028
## ---------------------------------------------------------------------------
## X5.latitude
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0      188        1    24.97  0.01384    24.94    24.95
##      .25      .50      .75      .90      .95
##    24.96    24.97    24.98    24.98    24.98
##
## lowest : 24.93207 24.93293 24.93885 24.94155 24.94297
## highest: 24.99006 24.99176 24.99800 25.00115 25.01459
## ---------------------------------------------------------------------------
## X6.longitude
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0      180        1    121.5  0.01626    121.5    121.5
##      .25      .50      .75      .90      .95
##    121.5    121.5    121.5    121.5    121.5
##
```

```
## lowest : 121.4735 121.4752 121.4846 121.4954 121.4958
## highest: 121.5539 121.5548 121.5596 121.5617 121.5663
## ---------------------------------------------------------------------------
## Y.house.price.of.unit.area
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      289        0      212        1    37.88    15.66    15.72    20.90
##      .25      .50      .75      .90      .95
##    26.50    39.30    46.60    55.02    59.56
##
## lowest :   7.6  11.2  12.8  12.9  13.0, highest:  69.7  71.0  73.6  78.3 117.5
## ---------------------------------------------------------------------------
```

Data Visualization

- Displaying histogram plots for the selected variables. ### Generating histogram plot for X2.house.age

```
p01 <- crs %>%
  with(dataset[train,]) %>%
  dplyr::select(X2.house.age) %>%
  ggplot2::ggplot(ggplot2::aes(x=X2.house.age)) +
  ggplot2::geom_density(lty=3) +
  ggplot2::ggtitle("Distribution of X2.house.age (sample)") +
  ggplot2::labs(y="Density")
```

Generating histogram plot for X3.distance.to.the.nearest.MRT.station

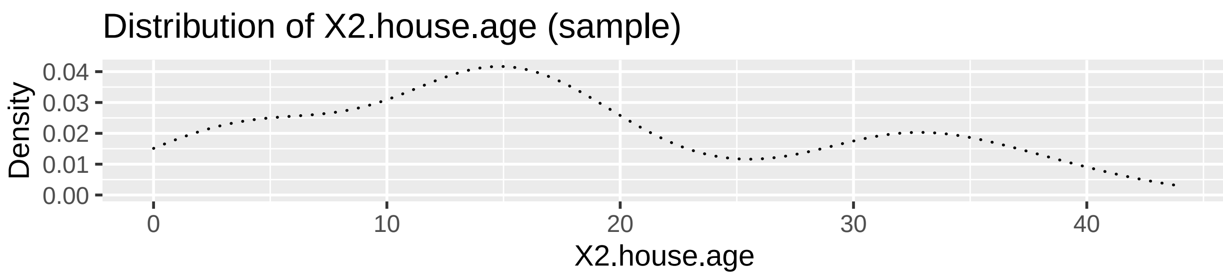
```
p02 <- crs %>%
  with(dataset[train,]) %>%
  dplyr::select(X3.distance.to.the.nearest.MRT.station) %>%
  ggplot2::ggplot(ggplot2::aes(x=X3.distance.to.the.nearest.MRT.station)) +
  ggplot2::geom_density(lty=3) +
  ggplot2::ggtitle("Distribution of X3.distance.to.the.nearest.MRT.station (sample)") +
  ggplot2::labs(y="Density")
```

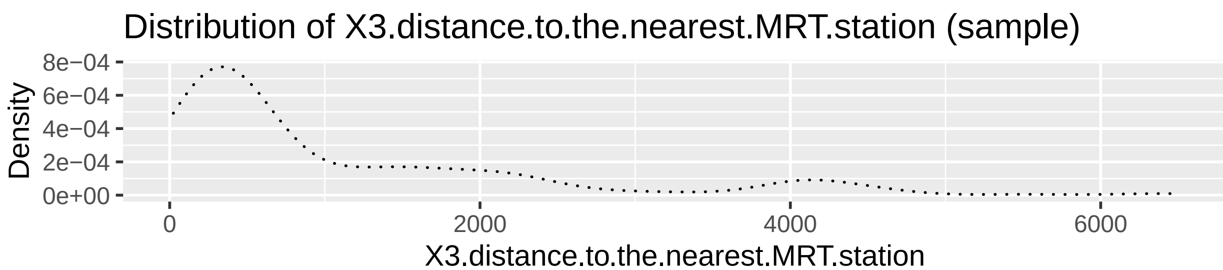
Generating histogram plot for X4.number.of.convenience.stores

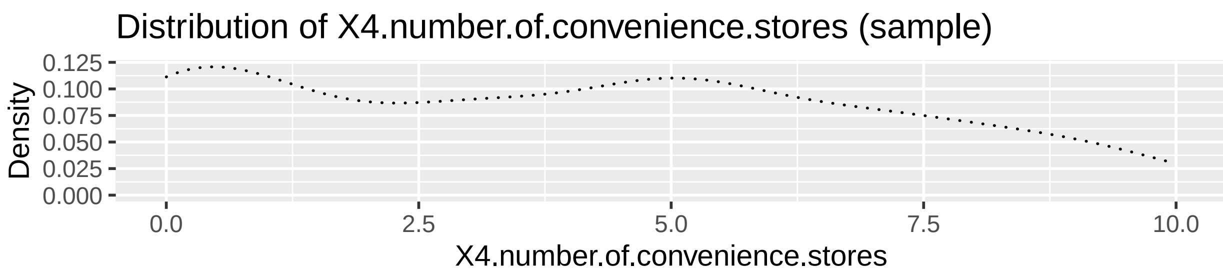
```
p03 <- crs %>%
  with(dataset[train,]) %>%
  dplyr::select(X4.number.of.convenience.stores) %>%
  ggplot2::ggplot(ggplot2::aes(x=X4.number.of.convenience.stores)) +
  ggplot2::geom_density(lty=3) +
  ggplot2::ggtitle("Distribution of X4.number.of.convenience.stores (sample)") +
  ggplot2::labs(y="Density")
```

Displaying the plots.

```
gridExtra::grid.arrange(p01, p02, p03)
```







Insights

- The first thing we notice is that price of houses nearest to MRT station are more.
- The second thing we notice is that price of houses with 0 to 1 or 4 to 5 convenience stores near by are more.

Methods / Analysis

Clustering

A cluster analysis will identify groups within a dataset. The KMeans clustering algorithm will search for K clusters (which you specify). The resulting K clusters are represented by the mean or average values of each of the variables.

By default KMeans only works with numeric variables. ### KMeans * Reseting the random number seed to obtain the same results each time.

```
set.seed(crv$seed)
```

- Generating a kmeans cluster of size 10.

```
crs$kmeans <- kmeans(sapply(na.omit(crs$dataset[crs$train, crs$numeric]), rescaler, "range"), 10)
```

- Report on the cluster characteristics.
- Cluster sizes:

```
paste(crs$kmeans$size, collapse=' ')
```

```
## [1] "41 36 30 19 47 18 26 5 18 49"
```

- Data means:

```
colMeans(sapply(na.omit(crs$dataset[crs$train, crs$numeric]), rescaler, "range"))
```

```
##                          X2.house.age X3.distance.to.the.nearest.MRT.station
##                             0.3985243                              0.1669702
##        X4.number.of.convenience.stores                            X5.latitude
##                             0.4013841                              0.4466725
##                           X6.longitude
##                             0.6370726
```

- Cluster centers:

```
crs$kmeans$centers
```

```
##    X2.house.age X3.distance.to.the.nearest.MRT.station
## 1    0.32782047                             0.31880268
## 2    0.08707509                             0.03046333
## 3    0.78409437                             0.04429593
## 4    0.36325403                             0.04175362
## 5    0.25026717                             0.14371437
## 6    0.75951294                             0.06670442
## 7    0.44520548                             0.67572084
## 8    0.72146119                             0.17854078
## 9    0.77447996                             0.12792863
## 10   0.27788650                             0.06622810
##    X4.number.of.convenience.stores X5.latitude X6.longitude
## 1                       0.27560976   0.4261702    0.4393479
## 2                       0.77222222   0.4848151    0.7336011
```

```
## 3                          0.80333333   0.5341937    0.7237258
## 4                          0.73157895   0.4840358    0.7424378
## 5                          0.06170213   0.4226854    0.7132271
## 6                          0.51111111   0.4980880    0.7173100
## 7                          0.01923077   0.1522941    0.2785621
## 8                          0.00000000   0.2091857    0.6111279
## 9                          0.17222222   0.5238730    0.6331560
## 10                         0.47346939   0.5239274    0.7294829
```

- Within cluster sum of squares:

```
crs$kmeans$withinss
```

```
##  [1] 1.8618190345 1.2282816760 0.6917138826 0.5257203677 3.1928393135
##  [6] 0.5070401247 1.2267361719 0.0009575018 0.8887540745 1.7426147025
```

Hierarchical Cluster

Generating a hierarchical cluster from the numeric data.

```
crs$dataset[crs$train, crs$numeric] %>%
  amap::hclusterpar(method="euclidean", link="ward", nbproc=1) ->
  crs$hclust
```

Association Rule Analysis

Association analysis identifies relationships or affinities between observations and/or between variables. These relationships are then expressed as a collection of association rules. The approach has been particularly successful in mining very large transaction databases. It is also often referred to as basket (as in shopping basket) analysis.

The 'arules' package provides the 'arules' function.

Generating a transactions dataset.

```
crs$transactions <- as(split(crs$dataset[crs$train, crs$target],
                             crs$dataset[crs$train, crs$ident]),
                       "transactions")
```

Generating the association rules.

```
crs$apriori <- apriori(crs$transactions, parameter = list(support=0.100, confidence=0.100, minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5     0.1      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 28
##
```

```
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[212 item(s), 289 transaction(s)] done [0.00s].
## sorting and recoding items ... [0 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Summary the resulting rule set.

```
generateAprioriSummary(crs$apriori)
```

```
## [1] "Number of Rules: 0 \n\n"
```

Building Models

Decision Tree

The 'rpart' package provides the 'rpart' function.

- Reset the random number seed to obtain the same results each time.

```
set.seed(crv$seed)
```

Building the Decision Tree model.

```
crs$rpart <- rpart(Y.house.price.of.unit.area ~ .,
                   data=crs$dataset[crs$train, c(crs$input, crs$target)],
                   method="anova",
                   parms=list(split="information"),
                   control=rpart.control(usesurrogate=0,
                                         maxsurrogate=0),
                   model=TRUE)
```

Generating a textual view of the Decision Tree model.

```
print(crs$rpart)
```

```
## n= 289
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 289 57607.2100 37.87855
##    2) X3.distance.to.the.nearest.MRT.station>=981.5777 101  5678.0340 24.46337
##      4) X5.latitude< 24.98363 91  3059.8040 22.98022
##        8) X3.distance.to.the.nearest.MRT.station>=4007.266 23   266.2661 16.91304 *
##        9) X3.distance.to.the.nearest.MRT.station< 4007.266 68  1660.5290 25.03235 *
##      5) X5.latitude>=24.98363 10   596.4640 37.96000 *
##    3) X3.distance.to.the.nearest.MRT.station< 981.5777 188 23987.3700 45.08564
##      6) X2.house.age>=11.7 122  9203.9050 41.03197
##       12) X5.latitude< 24.96412 10   832.8640 29.56000 *
##       13) X5.latitude>=24.96412 112  6937.4760 42.05625
##         26) X6.longitude< 121.5403 58  2130.8860 38.92414 *
##         27) X6.longitude>=121.5403 54  3626.4680 45.42037
##           54) X5.latitude< 24.97702 28  1167.2270 42.07857 *
##           55) X5.latitude>=24.97702 26  1809.8000 49.01923 *
##      7) X2.house.age< 11.7 66  9073.0100 52.57879
##       14) X5.latitude< 24.97425 26  1254.8450 47.54615 *
##       15) X5.latitude>=24.97425 40  6731.6200 55.85000
```

```
##         30) X3.distance.to.the.nearest.MRT.station>=385.8173 11   332.0764 46.01818 *
##         31) X3.distance.to.the.nearest.MRT.station< 385.8173 29  4932.9080 59.57931 *
```

```
printcp(crs$rpart)
```

```
##
## Regression tree:
## rpart(formula = Y.house.price.of.unit.area ~ ., data = crs$dataset[crs$train,
##     c(crs$input, crs$target)], method = "anova", model = TRUE,
##     parms = list(split = "information"), control = rpart.control(usesurrogate = 0,
##         maxsurrogate = 0))
##
## Variables actually used in tree construction:
## [1] X2.house.age
## [2] X3.distance.to.the.nearest.MRT.station
## [3] X5.latitude
## [4] X6.longitude
##
## Root node error: 57607/289 = 199.33
##
## n= 289
##
##         CP nsplit rel error  xerror     xstd
## 1 0.485040      0   1.00000 1.00253 0.125658
## 2 0.099127      1   0.51496 0.53421 0.100918
## 3 0.035096      2   0.41583 0.46430 0.107386
## 4 0.024885      3   0.38074 0.44941 0.107047
## 5 0.022160      4   0.35585 0.46312 0.108090
## 6 0.020486      6   0.31153 0.44129 0.099009
## 7 0.019668      7   0.29105 0.42576 0.098907
## 8 0.011274      8   0.27138 0.40864 0.113024
## 9 0.010000      9   0.26010 0.39788 0.112110
```

```
cat("\n")
```

Building a Random Forest model using the traditional approach.

```
set.seed(crv$seed)

crs$rf <- randomForest::randomForest(Y.house.price.of.unit.area ~ .,
                                     data=crs$dataset[crs$train, c(crs$input, crs$target)],
                                     ntree=500,
                                     mtry=2,
                                     importance=TRUE,
                                     na.action=randomForest::na.roughfix,
                                     replace=FALSE)
```

Generating textual output of the 'Random Forest' model.

```
crs$rf
```

```
##
## Call:
##  randomForest(formula = Y.house.price.of.unit.area ~ ., data = crs$dataset[crs$train,      c(crs$input,
##                Type of random forest: regression
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 62.9085
##                     % Var explained: 68.44
```

Listing the importance of the variables.

```
rn <- crs$rf %>%
  randomForest::importance() %>%
  round(2)
rn[order(rn[,1], decreasing=TRUE),]
```

```
##                                        %IncMSE IncNodePurity
## X3.distance.to.the.nearest.MRT.station   29.27      11467.38
## X5.latitude                              26.90       7955.66
## X6.longitude                             20.58       5854.17
## X2.house.age                             19.18       4616.45
## X4.number.of.convenience.stores          15.37       3348.28
```

Linear Regression model (LM)

Building a Regression model.

```
crs$glm <- lm(Y.house.price.of.unit.area ~ ., data=crs$dataset[crs$train,c(crs$input, crs$target)])
```

Generating a textual view of the Linear model.

```
print(summary(crs$glm))
```

```
##
## Call:
## lm(formula = Y.house.price.of.unit.area ~ ., data = crs$dataset[crs$train,
##     c(crs$input, crs$target)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.848  -5.334  -1.043   4.694  76.402
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           -1.112e+04  7.844e+03  -1.418    0.157
## X2.house.age                          -2.888e-01  4.920e-02  -5.871 1.21e-08
## X3.distance.to.the.nearest.MRT.station -3.613e-03  8.818e-04  -4.097 5.46e-05
## X4.number.of.convenience.stores        1.138e+00  2.289e-01   4.970 1.16e-06
## X5.latitude                            3.029e+02  5.529e+01   5.478 9.52e-08
## X6.longitude                           2.965e+01  6.213e+01   0.477    0.634
##
## (Intercept)
## X2.house.age                          ***
## X3.distance.to.the.nearest.MRT.station ***
## X4.number.of.convenience.stores       ***
## X5.latitude                           ***
## X6.longitude
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 9.339 on 283 degrees of freedom
## Multiple R-squared:  0.5715, Adjusted R-squared:  0.5639
## F-statistic: 75.49 on 5 and 283 DF,  p-value: < 2.2e-16
```

```
cat('==== ANOVA ====')
```

```
## ==== ANOVA ====
```

```
print(anova(crs$glm))
```

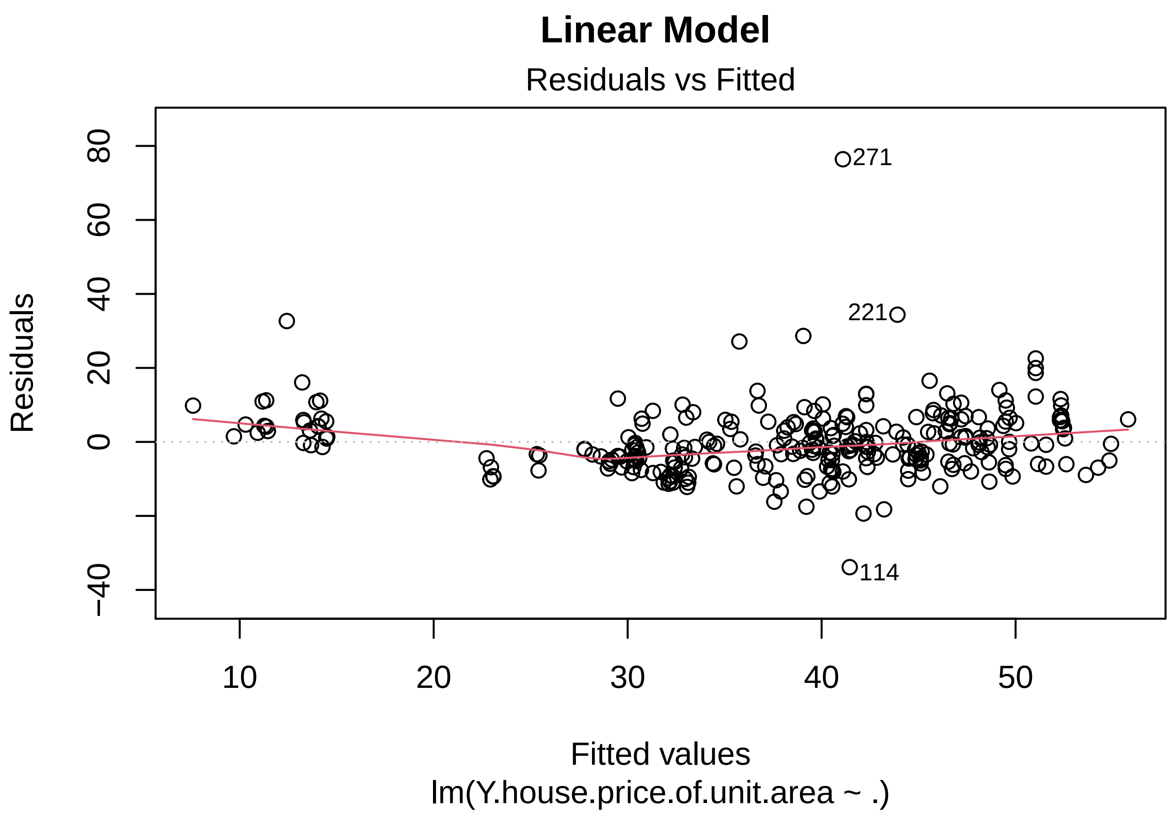
```
## Analysis of Variance Table
##
## Response: Y.house.price.of.unit.area
##                                         Df  Sum Sq Mean Sq  F value    Pr(>F)
## X2.house.age                             1  2926.6  2926.6  33.5526 1.844e-08
## X3.distance.to.the.nearest.MRT.station   1 24523.7 24523.7 281.1523 < 2.2e-16
## X4.number.of.convenience.stores          1  2851.1  2851.1  32.6869 2.746e-08
## X5.latitude                              1  2601.0  2601.0  29.8189 1.039e-07
## X6.longitude                             1    19.9    19.9   0.2278    0.6335
## Residuals                              283 24684.9    87.2
##
## X2.house.age                           ***
## X3.distance.to.the.nearest.MRT.station ***
## X4.number.of.convenience.stores        ***
## X5.latitude                            ***
## X6.longitude
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

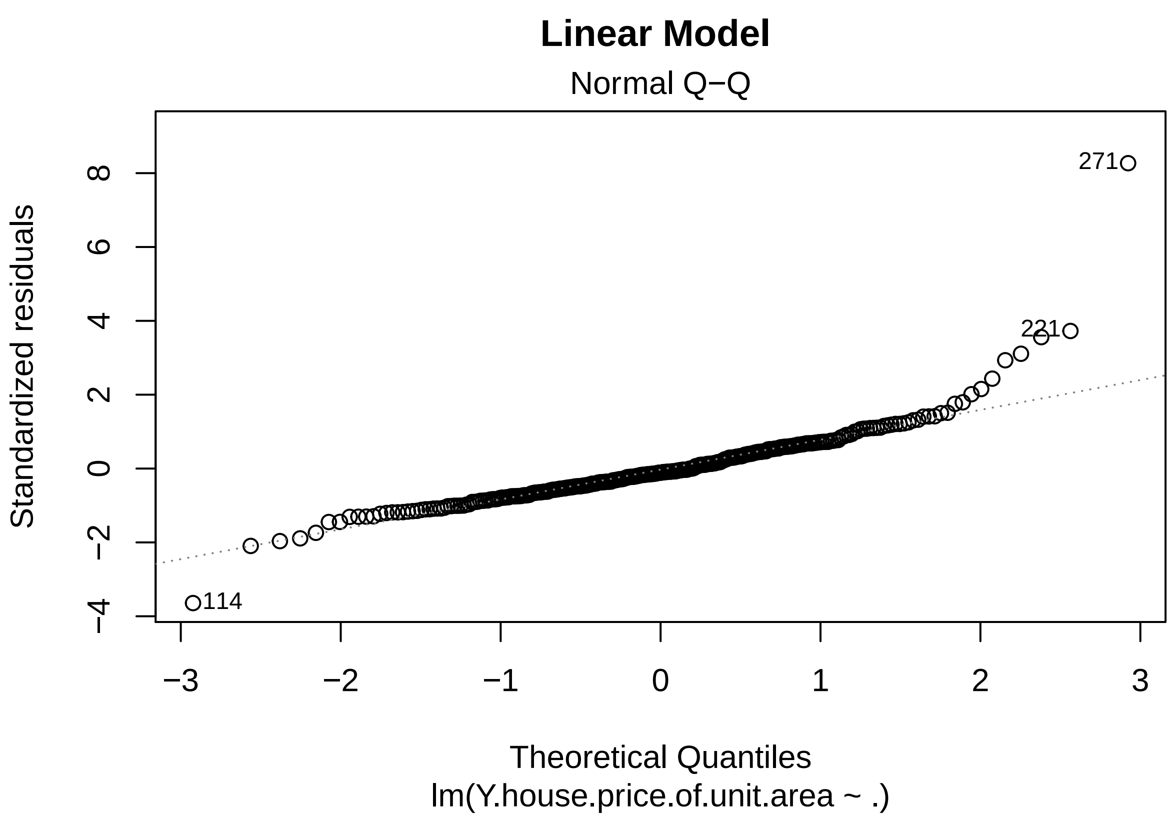
```
print(" ")
```

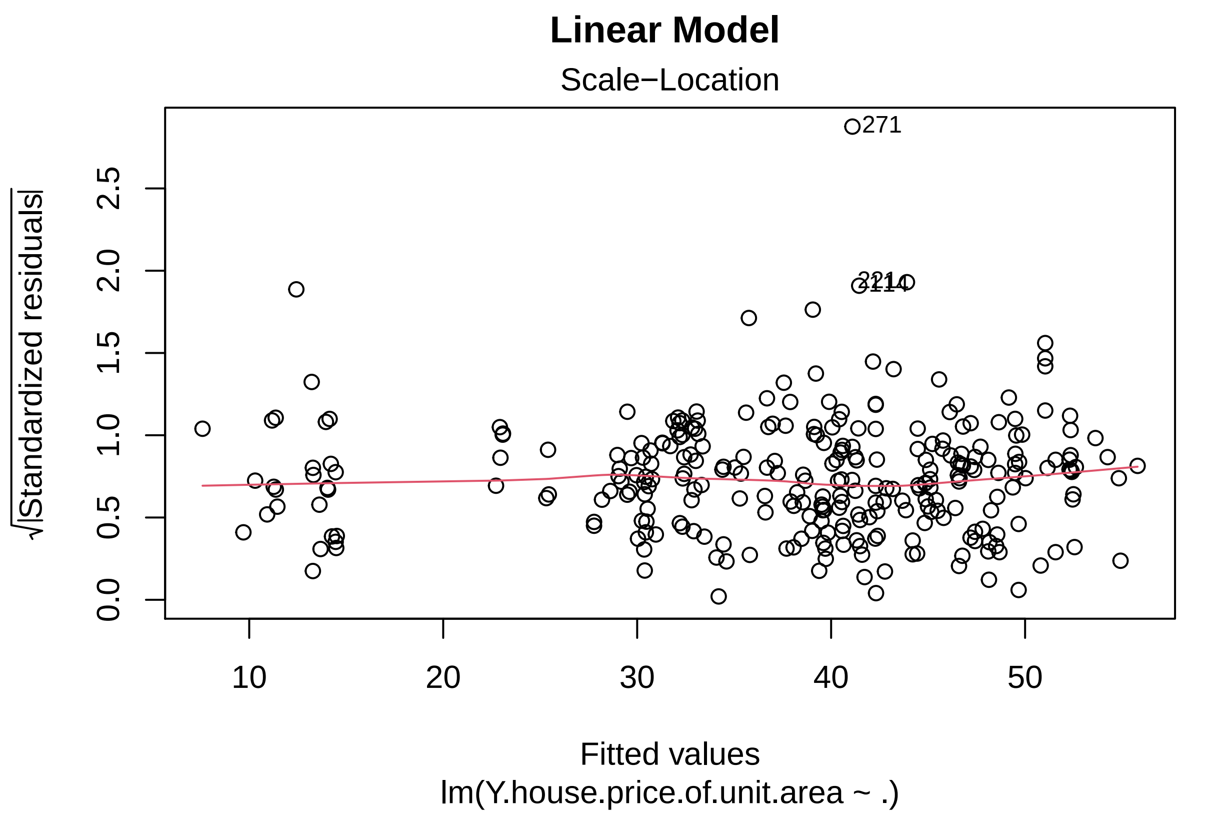
```
## [1] " "
```

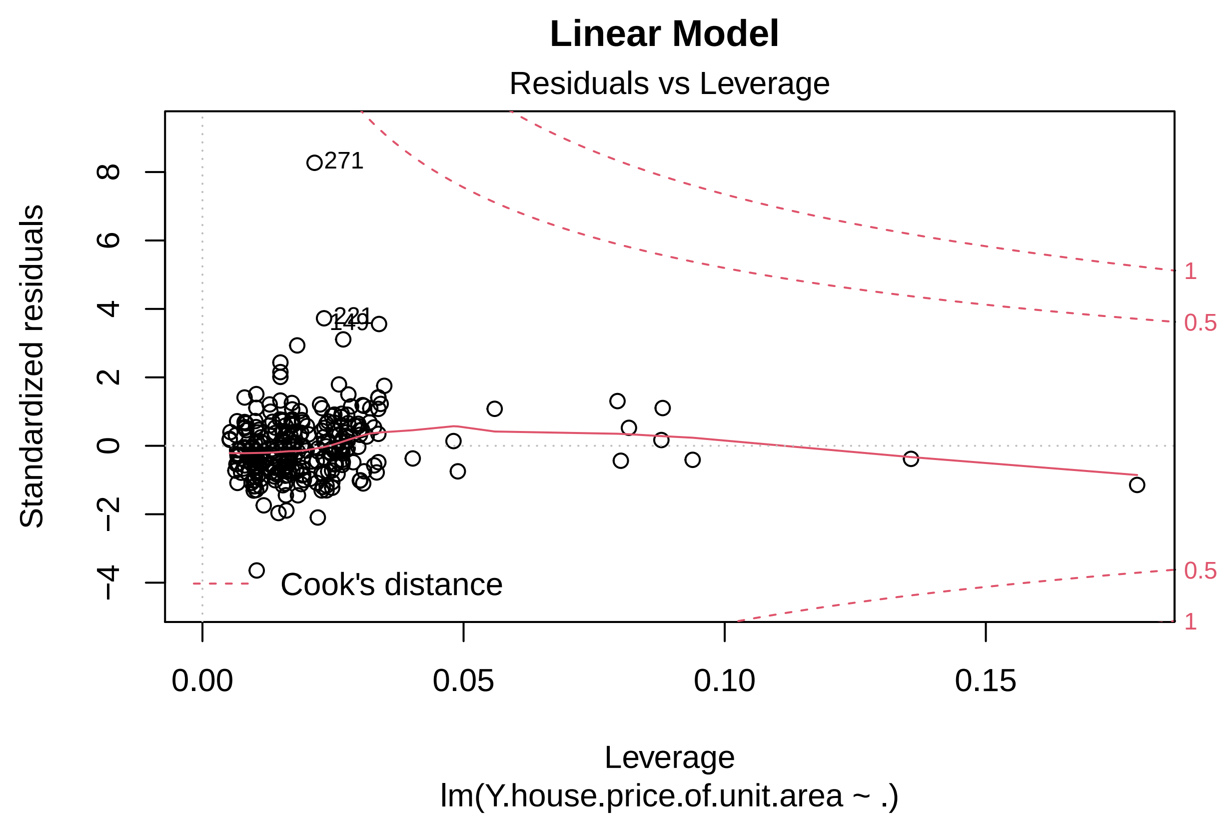
Plot the model evaluation.

```
ttl <- genPlotTitleCmd("Linear Model",crs$dataname,vector=TRUE)
plot(crs$glm, main=ttl[1])
```


Linear Model
Residuals vs Fitted
Residuals
Fitted values
lm(Y.house.price.of.unit.area ~ .)
271
221
114
−40
0
20
40
60
80
10
20
30
40
50


Linear Model
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Y.house.price.of.unit.area ~ .)
271
221
114
−4
−2
0
2
4
6
8
−3
−2
−1
0
1
2
3


Linear Model
Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
10
20
30
40
50
271
Fitted values
lm(Y.house.price.of.unit.area ~ .)


Linear Model
Residuals vs Leverage
Standardized residuals
−4
−2
0
2
4
6
8
0.00
0.05
0.10
0.15
271
Cook's distance
1
0.5
0.5
1
Leverage
lm(Y.house.price.of.unit.area ~ .)

Regression model (GLM)

Building a Regression model.

```
crs$glm <- glm(Y.house.price.of.unit.area ~ .,
           data=crs$dataset[crs$train,c(crs$input, crs$target)],
           family=gaussian(identity))
```

Generate a textual view of the Linear model.

```
print(summary(crs$glm))
```

```
##
## Call:
## glm(formula = Y.house.price.of.unit.area ~ ., family = gaussian(identity),
##     data = crs$dataset[crs$train, c(crs$input, crs$target)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -33.848   -5.334   -1.043    4.694   76.402
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           -1.112e+04  7.844e+03  -1.418    0.157
## X2.house.age                          -2.888e-01  4.920e-02  -5.871 1.21e-08
## X3.distance.to.the.nearest.MRT.station -3.613e-03  8.818e-04  -4.097 5.46e-05
## X4.number.of.convenience.stores         1.138e+00  2.289e-01   4.970 1.16e-06
## X5.latitude                             3.029e+02  5.529e+01   5.478 9.52e-08
## X6.longitude                            2.965e+01  6.213e+01   0.477    0.634
##
## (Intercept)
## X2.house.age                          ***
## X3.distance.to.the.nearest.MRT.station ***
## X4.number.of.convenience.stores       ***
## X5.latitude                           ***
## X6.longitude
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 87.22571)
##
##     Null deviance: 57607  on 288  degrees of freedom
## Residual deviance: 24685  on 283  degrees of freedom
## AIC: 2119.5
##
## Number of Fisher Scoring iterations: 2
```

```
cat('==== ANOVA ====')
```

```
## ==== ANOVA ====
```

```
print(anova(crs$glm))
```

```
## Analysis of Deviance Table
##
## Model: gaussian, link: identity
##
## Response: Y.house.price.of.unit.area
```

```
##
## Terms added sequentially (first to last)
##
##
##                                          Df Deviance Resid. Df Resid. Dev
## NULL                                                       288      57607
## X2.house.age                              1   2926.6       287      54681
## X3.distance.to.the.nearest.MRT.station    1  24523.7       286      30157
## X4.number.of.convenience.stores           1   2851.1       285      27306
## X5.latitude                               1   2601.0       284      24705
## X6.longitude                              1     19.9       283      24685
```

```r
print(" ")
```

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

Evaluating model performance on the validation dataset.

Predicted Versus Observed

The Predicted Versus Observed plot is relevant for regression models (predicting a continuous value rather than a discrete value). It will display the predicted values against the observed values, as the name suggests!

Two lines are also plotted, one being a linear fit to the actual points, and the other being the perfect fit, if the predicted values were the same as the actual observations.

The Pseudo R-Squared is a measure that tries to mimic the R-Squared. It is calculated as the square of the correlation between the predicted and observed values. The closer to 1, the better.

RPART : Generate a Predicted v Observed plot for Decision Tree model

on datasets_88705_204267_Real estate.csv [validate].

```r
crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input, crs$target)])
```

- Obtain the observed output for the dataset.

```r
obs <- subset(crs$dataset[crs$validate, c(crs$input, crs$target)], select=crs$target)
```

- Handle in case categoric target treated as numeric.

```r
obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(Y.house.price.of.unit.area=obs)
rownames(obs) <- obs.rownames
```

- Combine the observed values with the predicted.

```r
fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))
```

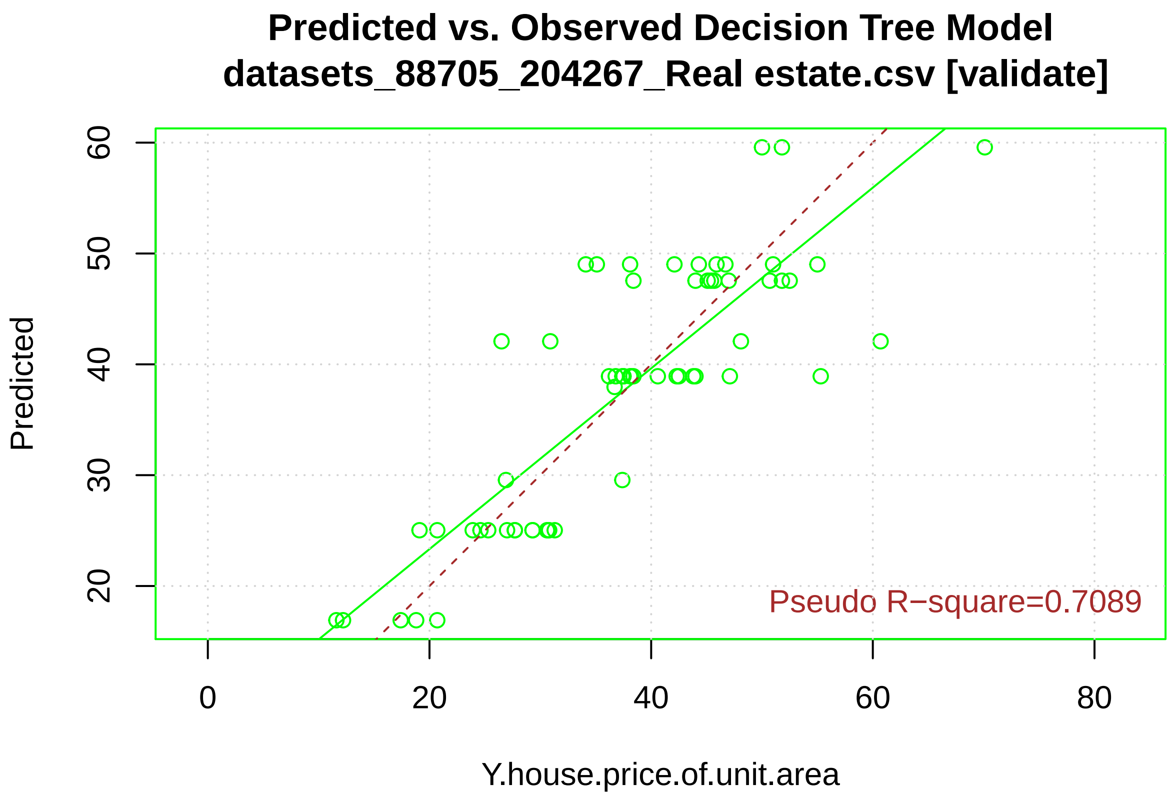
- Obtain the pseudo R2 - a correlation.

```r
fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)
dtRsqr <- fitcorr
```

- Plot settings for the true points and best fit.

```r
op <- par(c(lty="solid", col="green"))
# Display the observed (X) versus predicted (Y) points.
plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="Y.house.price.of.unit.area", ylab="Predicted")
# Generate a simple linear fit between predicted and observed.
prline <- lm(fitpoints[,2] ~ fitpoints[,1])
```

```
abline(prline) # Add the linear fit to the plot.
# Add a diagonal representing perfect correlation.
par(c(lty="dashed", col="brown"))
abline(0, 1)
# Include a pseudo R-square on the plot
legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")
# Add a title and grid to the plot.
title(main="Predicted vs. Observed Decision Tree Model
 datasets_88705_204267_Real estate.csv [validate]")
grid()
```



RF: Generate a Predicted v Observed plot for rf model

on datasets_88705_204267_Real estate.csv [validate].

```
crs$pr <- predict(crs$rf, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]))
```

- Obtain the observed output for the dataset.

```
obs <- subset(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]), select=crs$target)
```

- Handle in case categoric target treated as numeric.

```
obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(Y.house.price.of.unit.area=obs)
rownames(obs) <- obs.rownames
```

- Combine the observed values with the predicted.

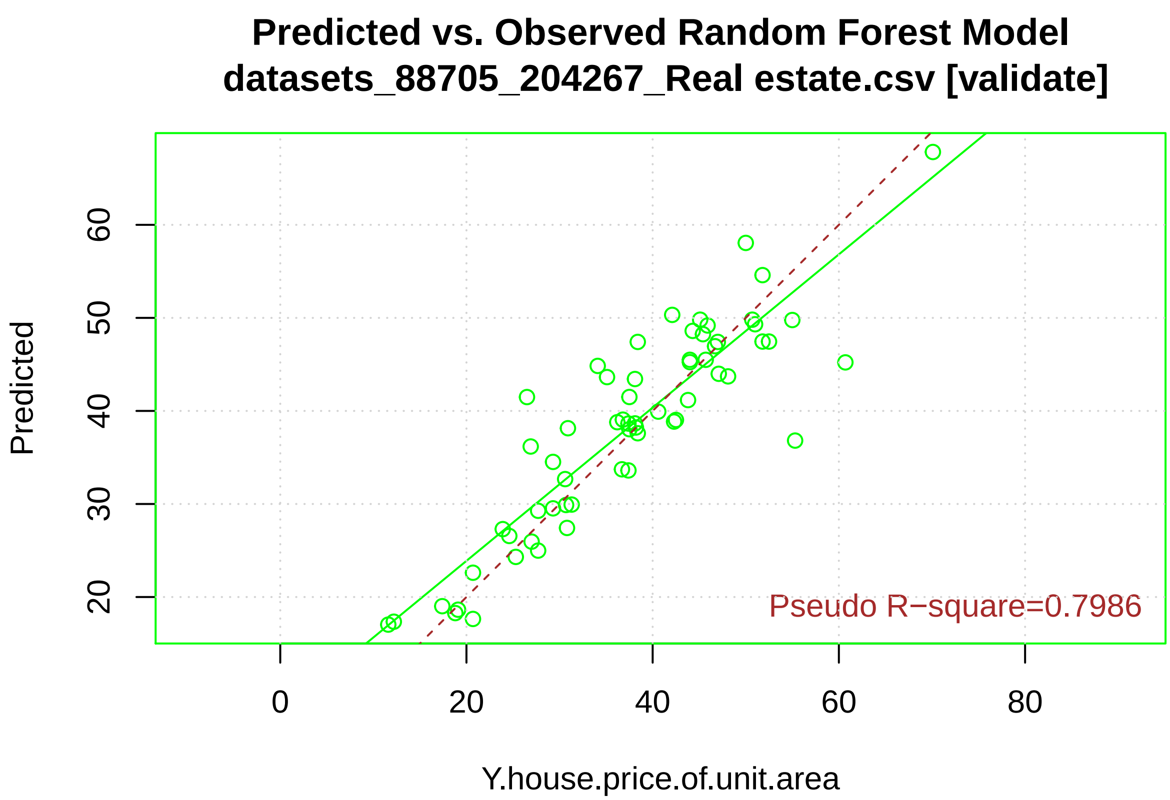
```
fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))
```

- Obtain the pseudo R2 - a correlation.

```
fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)
rfRsqr <- fitcorr
```

- Plot settings for the true points and best fit.

```
op <- par(c(lty="solid", col="green"))
# Display the observed (X) versus predicted (Y) points.
plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="Y.house.price.of.unit.area", ylab="Predicted")
# Generate a simple linear fit between predicted and observed.
prline <- lm(fitpoints[,2] ~ fitpoints[,1])
 # Add the linear fit to the plot.
abline(prline)
# Add a diagonal representing perfect correlation.
par(c(lty="dashed", col="brown"))
abline(0, 1)
# Include a pseudo R-square on the plot
legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")
# Add a title and grid to the plot.
title(main="Predicted vs. Observed Random Forest Model
 datasets_88705_204267_Real estate.csv [validate]")
grid()
```



GLM: Generate a Predicted v Observed plot for glm model

on datasets_88705_204267_Real estate.csv [validate].

```r
crs$pr <- predict(crs$glm,
                  type    = "response",
                  newdata = crs$dataset[crs$validate, c(crs$input, crs$target)])
```

- Obtain the observed output for the dataset.

```r
obs <- subset(crs$dataset[crs$validate, c(crs$input, crs$target)], select=crs$target)
```

- Handle in case categoric target treated as numeric.

```r
obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(Y.house.price.of.unit.area=obs)
rownames(obs) <- obs.rownames
```

- Combine the observed values with the predicted.

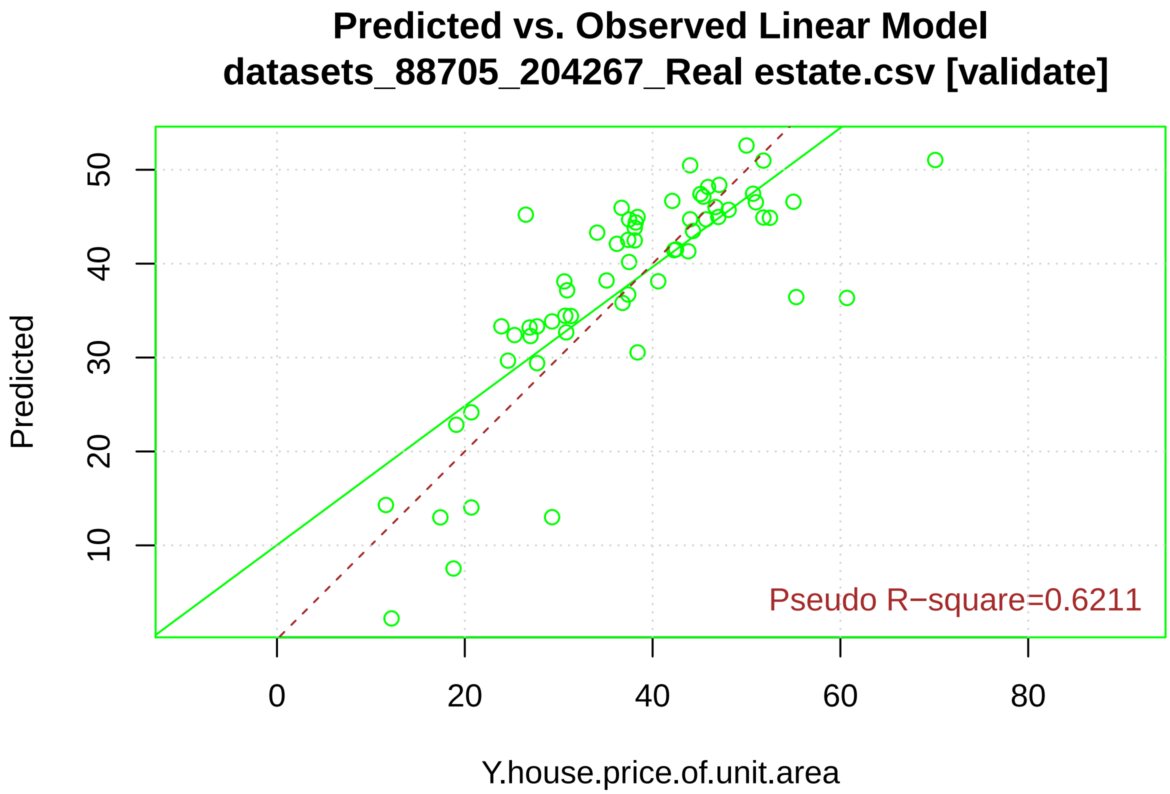
```r
fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))
```

- Obtain the pseudo R2 - a correlation.

```r
fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)
lmRsqr <- fitcorr
```

- Plot settings for the true points and best fit.

```r
op <- par(c(lty="solid", col="green"))
# Display the observed (X) versus predicted (Y) points.
plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="Y.house.price.of.unit.area", ylab="Predicted")
prline <- lm(fitpoints[,2] ~ fitpoints[,1])
 # Add the linear fit to the plot.
abline(prline)
 # Add a diagonal representing perfect correlation.
par(c(lty="dashed", col="brown"))
abline(0, 1)
 # Include a pseudo R-square on the plot
legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")
# Add a title and grid to the plot.
title(main="Predicted vs. Observed Linear Model
 datasets_88705_204267_Real estate.csv [validate]")
grid()
```


Predicted vs. Observed Linear Model
datasets_88705_204267_Real estate.csv [validate]
Predicted
Y.house.price.of.unit.area
Pseudo R−square=0.6211
0
20
40
60
80
10
20
30
40
50

Conclusion

- Based on the pseudo R-square results of the 3 models used
- Linear Model 0.6211
- Random Forest Model 0.7986
- Decision Tree Model 0.7089

```
## [1] "Linear Model has least pseudo R-square of 0.6211 over the other 2 models"
```

```
## [1] "Random Forest Model has higher pseudo R-square of 0.7986 over the other 2 models"
```

Report generation information:

- created by: Sudha Kankipati
- Report creation date: Wed Jun 17 2020
- R version 4.0.0 (2020-04-24).
- Platform: x86_64-w64-mingw32/x64 (64-bit)(Windows 10 x64 (build 18363)).
- Placed files for this project in https://github.com/DrSudhaK/RealEstatePricePredictionProject.git
- Dataset is located at https://www.kaggle.com/quantbruce/real-estate-price-prediction?select=Real+estate.csv click on the download button "datasets_88705_204267_Real estate.csv" will be downloaded.