ECE-9603A Assignment 1: Forecasting

Joe Abley – jabley@uwo.ca

2018-10-10

Contents

Abstract

This paper is submitted for Assignment 1, ECE-9603A, Fall 2018, Western University Faculty of Engineering, Department of Electrical and Computer Engineering. It has been written in R Markdown[1]; the code used to produce the output included in this document is included with the document source[2].

The subject of this assignment is experimentation with different forecasting approaches and algorithms.

[1] https://rmarkdown.rstudio.com
[2] https://github.com/ableyjoe/uwo-mesc/tree/master/ECE-9603A-001-GF18/assignment1

Forecasting Problem

Given a dataset that describes various features of individual houses along with details of their sale, identify and test forecasting models that are able to predict the prices realised by the sale of houses based on an appropriate set of parameters.

Available Data

We make use of a dataset that describes house sales in Iowa, published on and retrieved from Kaggle as part of a Kaggle[3] competition entitled "House Prices: Advanced Regression Techniques"[4]. This data set was suggested in directions for this assignment.

Importing Data

Two datasets are provided, a training set and a test set:

```
train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

However, the `test` set does not contain a `SalePrice` column: the objective of the competition is to populate one with predicted values. The `train` set does, however.

```
summary(train$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

```
summary(test$SalePrice)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

In order to cross-validate the accuracy of the predictions used by different models it will be convenient to have a test set that includes a `SalePrice` column. We shall therefore discard the supplied test set and construct a replacement from the supplied train set. The train set will be reduced correspondingly in order to avoid contamination.

```
data <- train

# start again to ensure the intersection between test and train is null
rm(test)
rm(train)

sample <- sample.int(n = nrow(data), size = floor(0.2 * nrow(data)))
train <- data[-sample,]
test <- data[sample,]
```

Abridged Feature Engineering

```
dim(train)
```

[3] https://www.kaggle.com/
[4] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

```
## [1] 1168   81
```

There are 1168 rows in this dataset and 81 columns. Of those columns one is a numeric id and one is the sale price; the other 79 are parameters that describe each house, some of which are numeric variables and some of which are categories.

From the description provided with the source data, the numeric variables are as follows:

Variable Name	Description
`LotFrontage`	Linear feet of street connected to property
`LotArea`	Lot size in square feet
`MasVnrArea`	Masonry veneer area in square feet
`BsmtFinSF2`	Type 2 finished square feet
`BsmtUnfSF`	Unfinished square feet of basement area
`TotalBsmtSF`	Total square feet of basement area
`1stFlrSF`	First Floor square feet
`2ndFlrSF`	Second floor square feet
`LowQualFinSF`	Low quality finished square feet (all floors)
`GrLivArea`	Above grade (ground) living area square feet
`BsmtFullBath`	Basement full bathrooms
`BsmtHalfBath`	Basement half bathrooms
`FullBath`	Full bathrooms above grade
`HalfBath`	Half baths above grade
`Bedroom`	Bedrooms above grade (does NOT include basement bedrooms)
`Kitchen`	Kitchens above grade
`TotRmsAbvGrd`	Total rooms above grade (does not include bathrooms)
`Fireplaces`	Number of fireplaces
`GarageCars`	Size of garage in car capacity
`GarageArea`	Size of garage in square feet
`WoodDeckSF`	Wood deck area in square feet
`OpenPorchSF`	Open porch area in square feet
`EnclosedPorch`	Enclosed porch area in square feet
`3SsnPorch`	Three season porch area in square feet
`ScreenPorch`	Screen porch area in square feet
`PoolArea`	Pool area in square feet
`MiscVal`	$Value of miscellaneous feature

The stated purpose of this assignment is "to experiment with different models"" and its focus is "applying forecasting approaches and not on optimising models". In the spirit of that direction we will not complete a detailed feature analysis and instead will select a set of numeric samples that seem likely to be sufficiently representative to give some kind of correlation, based on general background knowledge gained buying and selling houses in places other than Iowa, because how different can people from Iowa be? A smaller set of features seems helpful.

We can construct a new variable `newBathrooms`, derived from the various other bathroom variables:

- `newBathrooms` (Total number of bathrooms, full and half, all levels) = `BsmtFullBath` + `BsmtHalfBath` + `FullBath` + `HalfBath`

```
train$newBathrooms = train$BsmtFullBath + train$BsmtHalfBath + train$FullBath + train$HalfBath
```

We can eliminate some anticipated redundancy by identifying variables that seem likely to be closely related, and arbitrarily choosing the one that seems most interesting.

- `LotFrontage` and `LotArea` both relate to the size of the lot, which seems pertinent. Retain `LotArea`.
- `BsmtFinSF2`, `BsmtUnfSF` and `TotalBsmtSF` all relate to the size of the basement. Retain `TotalBsmtSF`.

- `1stFlrSF`, `2ndFlrSF`, `LowQualFinSF` and `GrLivArea` all relate to the size of the rest of the house. Retain `GrLivArea`.
- `Bedroom`, `Kitchen` and `TotRmsAbvGrd` all relate to the number of rooms above the basement. Retain `TotRmsAbvGrd`.
- `GarageCars` and `GarageArea` both relate to the size of the garage. Retain `GarageArea`.

We can keep some variables as-is, because they seem harmless and potentially interesting:

- `Fireplaces`

We arbitrarily declare all remaining variables to be uninteresting. We take care to retain `SalePrice` which is our outcome/response variable.

```
interesting <- c("newBathrooms", "LotArea", "TotalBsmtSF", "GrLivArea", "TotRmsAbvGrd",
  "Fireplaces", "GarageArea", "SalePrice")
train <- train[, (names(train) %in% interesting)]
```

To avoid surprises, we check for variables that might have missing data. Fortunately we seem not to have any.

```
which(colSums(is.na(train)) > 0)
```

```
## named integer(0)
```

Our cauterised training data set now looks like this:

```
summary(train)
```

```
##     LotArea        TotalBsmtSF     GrLivArea     TotRmsAbvGrd
##  Min.   :  1300   Min.   :   0   Min.   : 438   Min.   : 3.000
##  1st Qu.:  7500   1st Qu.: 793   1st Qu.:1124   1st Qu.: 5.000
##  Median :  9352   Median : 980   Median :1449   Median : 6.000
##  Mean   : 10521   Mean   :1048   Mean   :1500   Mean   : 6.464
##  3rd Qu.: 11533   3rd Qu.:1274   3rd Qu.:1764   3rd Qu.: 7.000
##  Max.   :215245   Max.   :6110   Max.   :5642   Max.   :14.000
##    Fireplaces       GarageArea       SalePrice       newBathrooms
##  Min.   :0.0000   Min.   :   0.0   Min.   : 34900   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.: 334.5   1st Qu.:129500   1st Qu.:2.000
##  Median :1.0000   Median : 474.0   Median :162000   Median :2.000
##  Mean   :0.6113   Mean   : 470.3   Mean   :178864   Mean   :2.415
##  3rd Qu.:1.0000   3rd Qu.: 576.0   3rd Qu.:213000   3rd Qu.:3.000
##  Max.   :3.0000   Max.   :1418.0   Max.   :755000   Max.   :6.000
```

Finally, we transform and reduce our test data set in the same way, since keeping it the same seems less likely to be confusing.

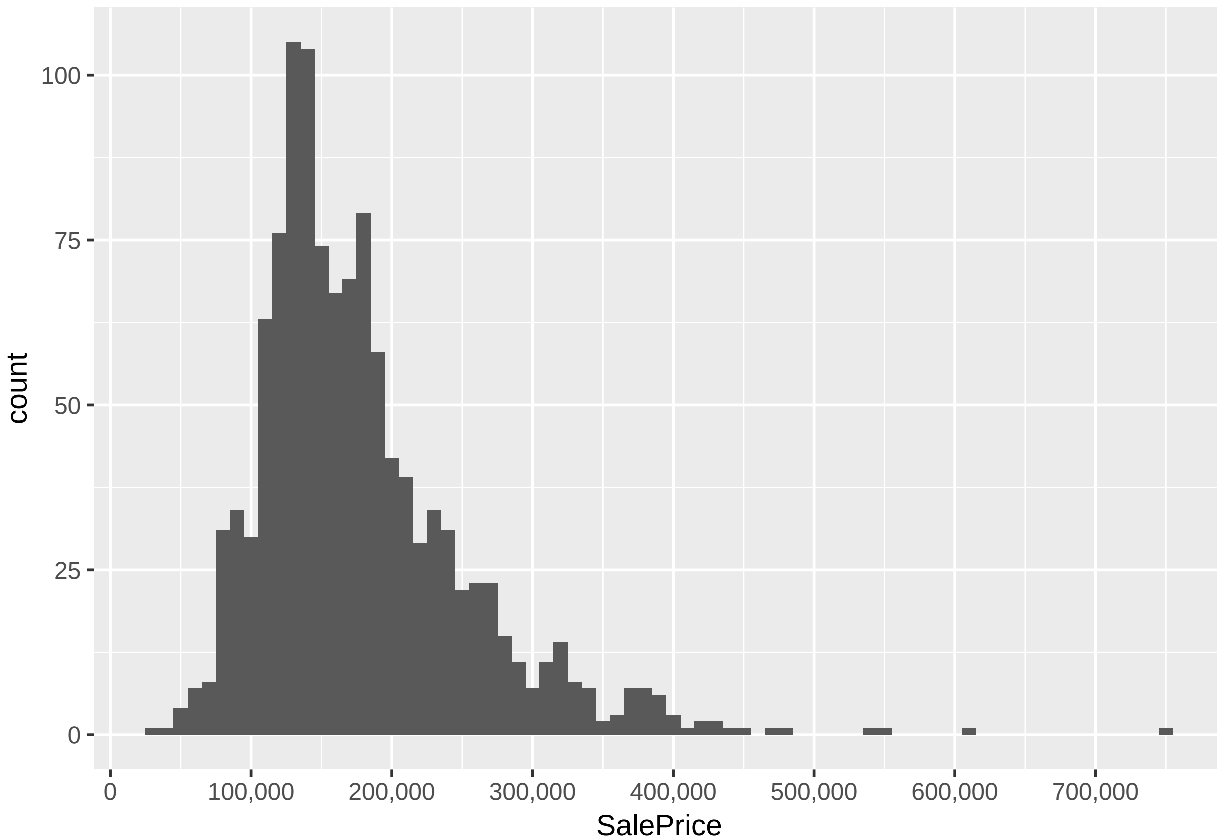
```
test$newBathrooms = test$BsmtFullBath + test$BsmtHalfBath + test$FullBath + test$HalfBath
test <- test[, (names(test) %in% interesting)]
```

Data Inspection

SalePrice

The distribution of sale prices is not symmetrical; there are more houses sold at lower prices and a long tail of expensive houses as is shown in the following histogram.

```
ggplot(data=train[!is.na(train$SalePrice),], aes(x = SalePrice)) +
  geom_histogram(binwidth = 10000) +
  scale_x_continuous(breaks = seq(0, 800000, by = 100000), labels = comma)
```



This distribution is observed to be skewed to the right, as can be confirmed numerically:

```
skewness(train$SalePrice)
```

```
## [1] 1.6522
```

The positive result confirms a skew to the right. We transform `SalePrice` data by replacing it with `log(SalePrice + 1)`:

```
train$SalePrice <- log(train$SalePrice + 1)
summary(train$SalePrice)
```

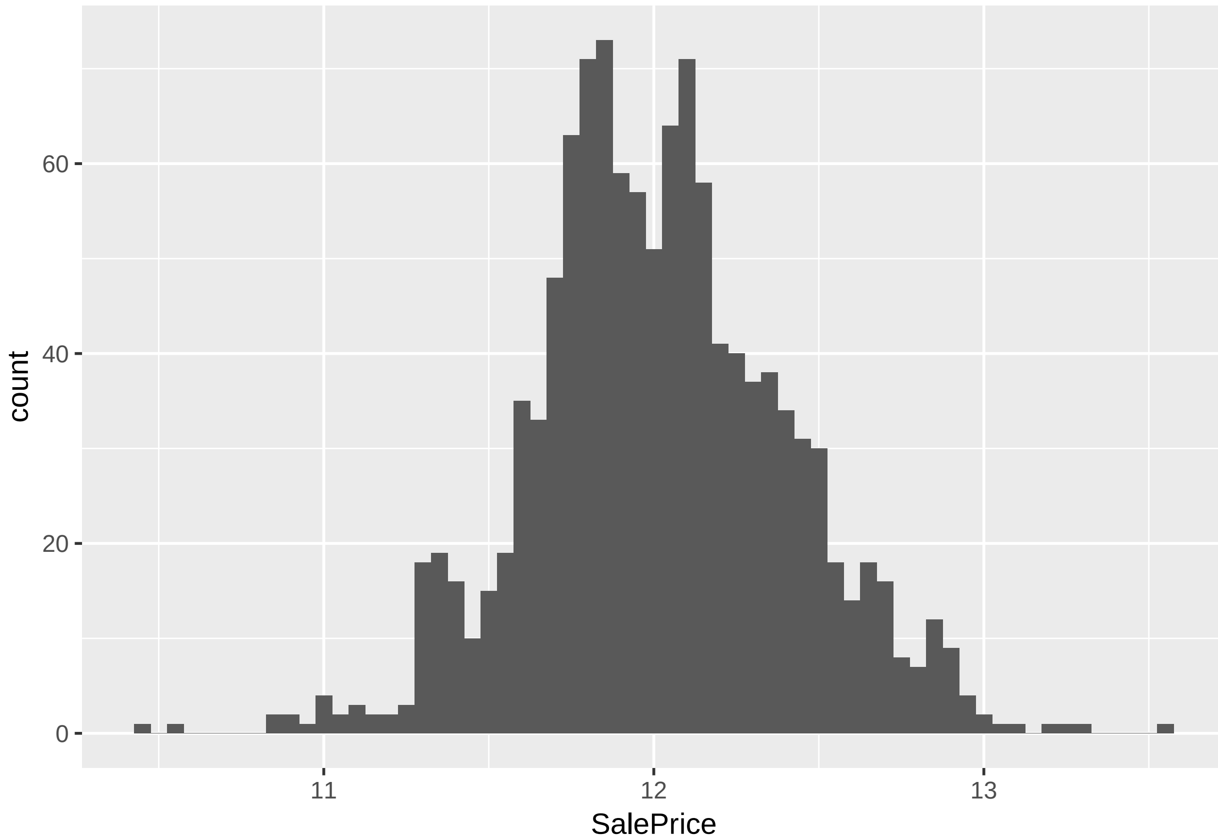
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.46   11.77   12.00   12.02   12.27   13.53
```

```
skewness(train$SalePrice)
```

```
## [1] 0.09345119
```

The skew is now much closer to zero, as can be confirmed visually:

```r
ggplot(data=train, aes(x = SalePrice)) +
  geom_histogram(binwidth = 0.05) +
  scale_x_continuous(breaks = seq(0, 20, by = 1), labels = comma)
```



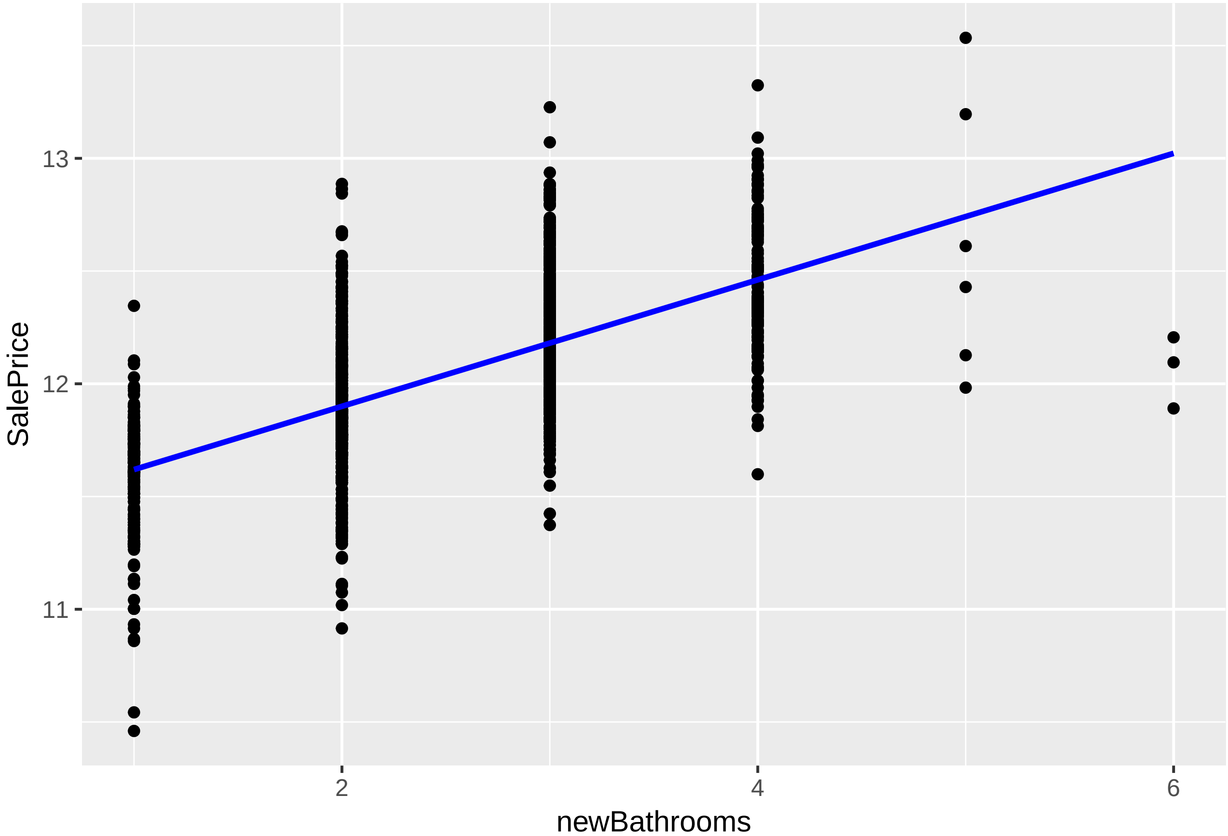
We apply the same logarithmic transform to the test set:

```r
test$SalePrice <- log(test$SalePrice + 1)
```

newBathrooms

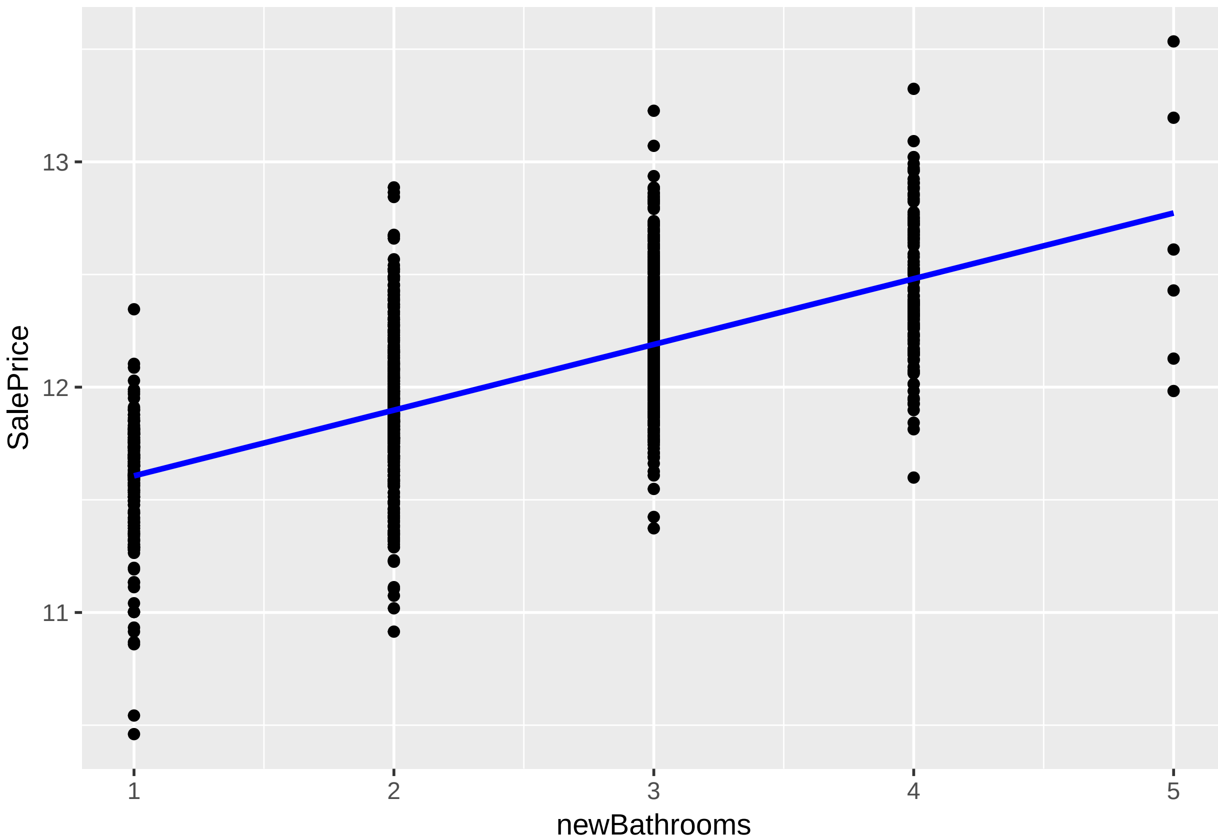
The total number of bathrooms seems to correlate to `SalePrice`, although there are a small number of outliers that suggest that at some point you really don't get much value from adding more toilets.

```
ggplot(data=train, aes(x=newBathrooms, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



The houses with six bathrooms don't seem to fit a linear relationship very well. Since they represent a tiny minority of the observations they will be eliminated as outliers.

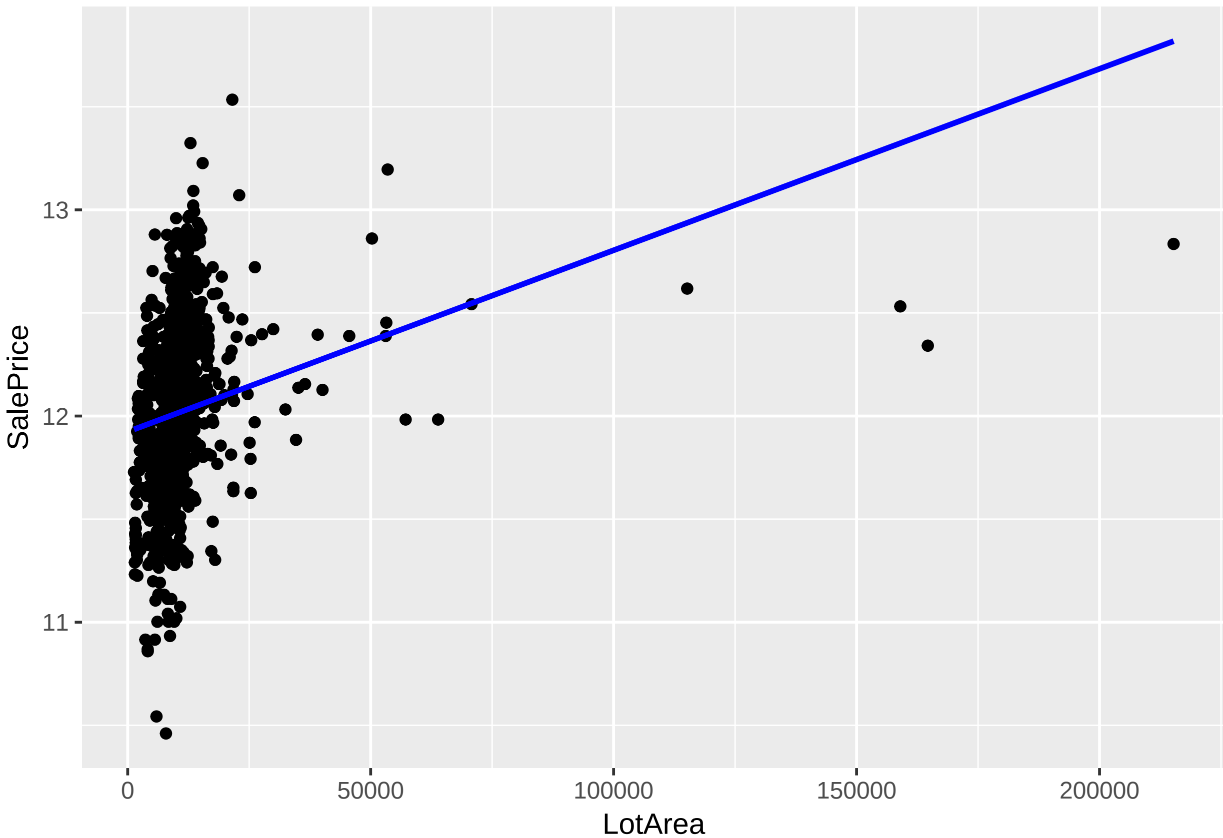
```
train <- train[train$newBathrooms < 6, ]

ggplot(data=train, aes(x=newBathrooms, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```


13
12
11
SalePrice
1
2
3
4
5
newBathrooms

LotArea

For many houses there seems to be a strong correlation between `LotArea` and `SalePrice`. As with the tentative toilet hypothesis, however, it seems possible that the size of the lot beyond a certain point just starts to seem more annoying to mow.

```
ggplot(data=train, aes(x=LotArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```

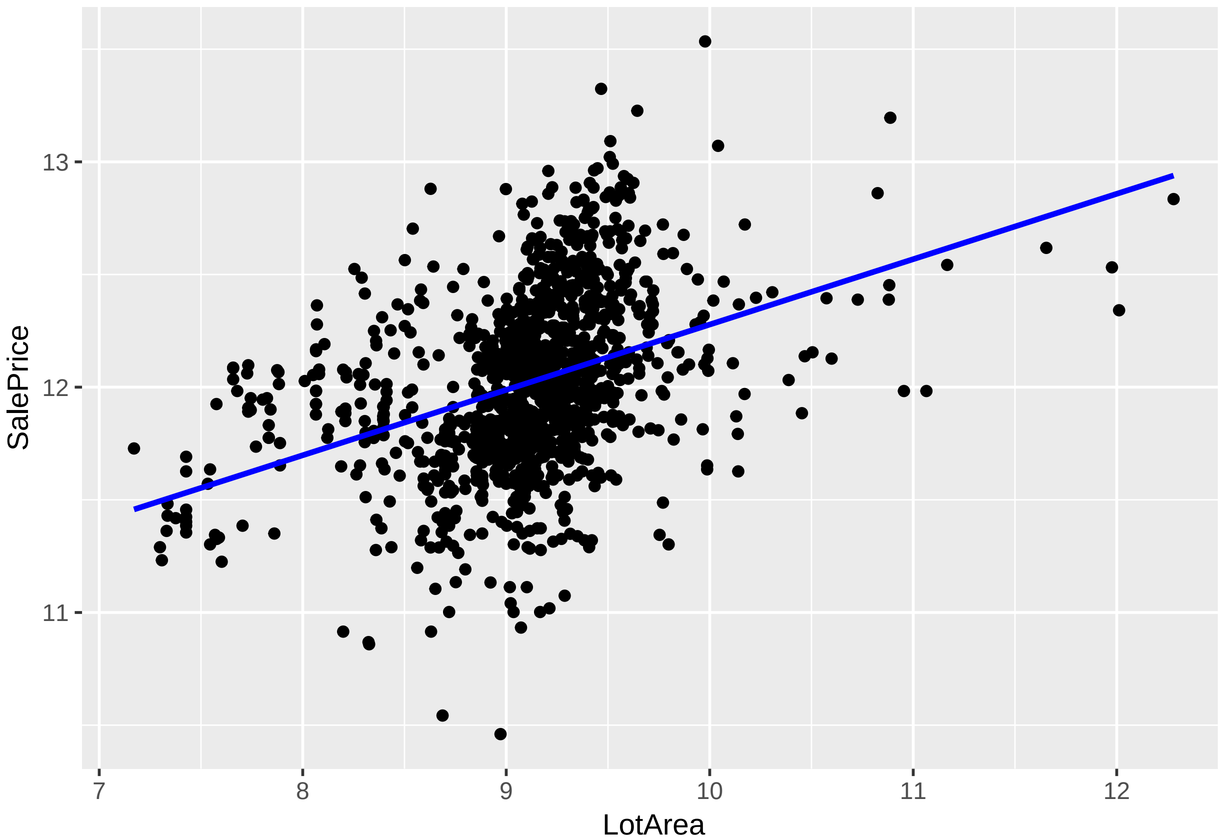


We will try to transform the `LotArea` variable as `log(LotArea)` to see whether that provides a more convincing linear relationship.

```
train$LotArea <- log(train$LotArea + 1)
summary(train$LotArea)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.171   8.923   9.143   9.098   9.352  12.280
```

```
ggplot(data=train, aes(x=LotArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



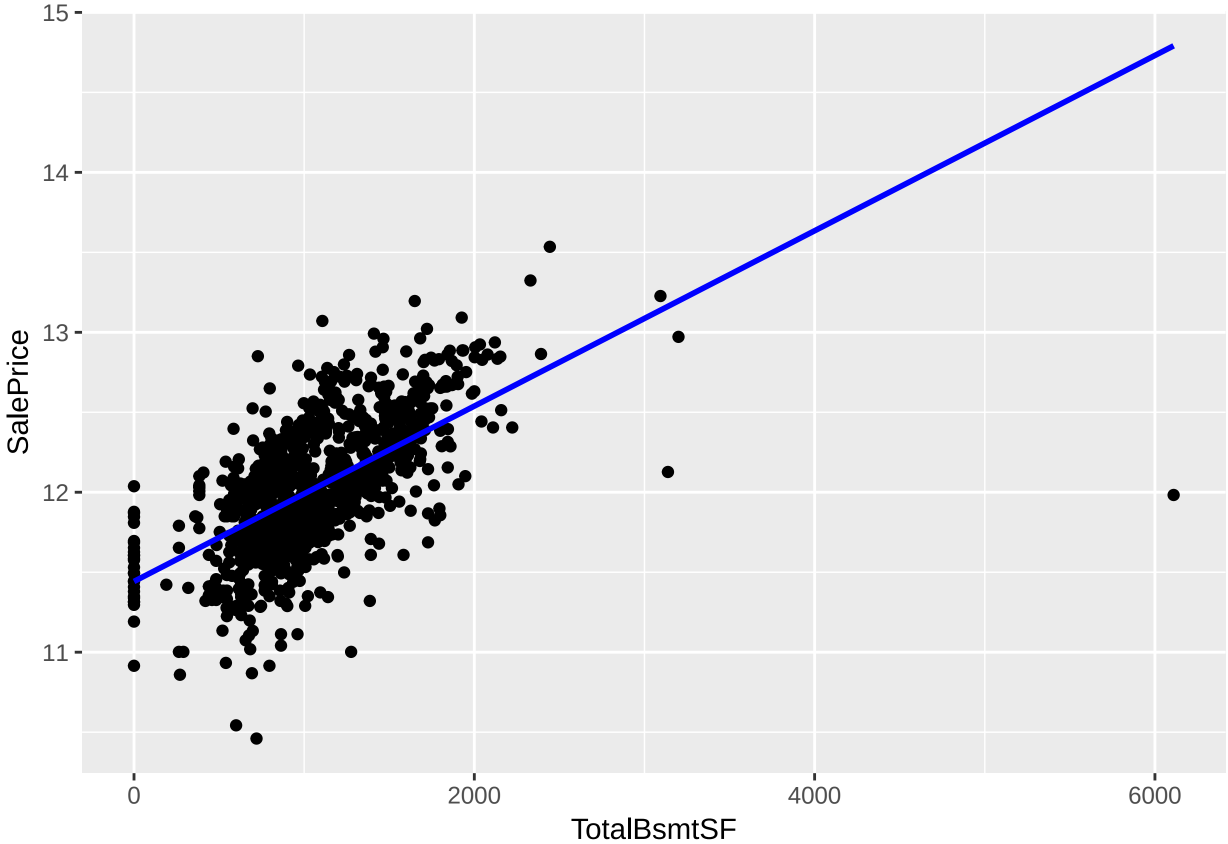
This looks slightly more convincing, although it does not show strong correlation. We will apply the same logarithmic transform to the test set.

```
test$LotArea <- log(test$LotArea + 1)
```

TotalBsmtSF

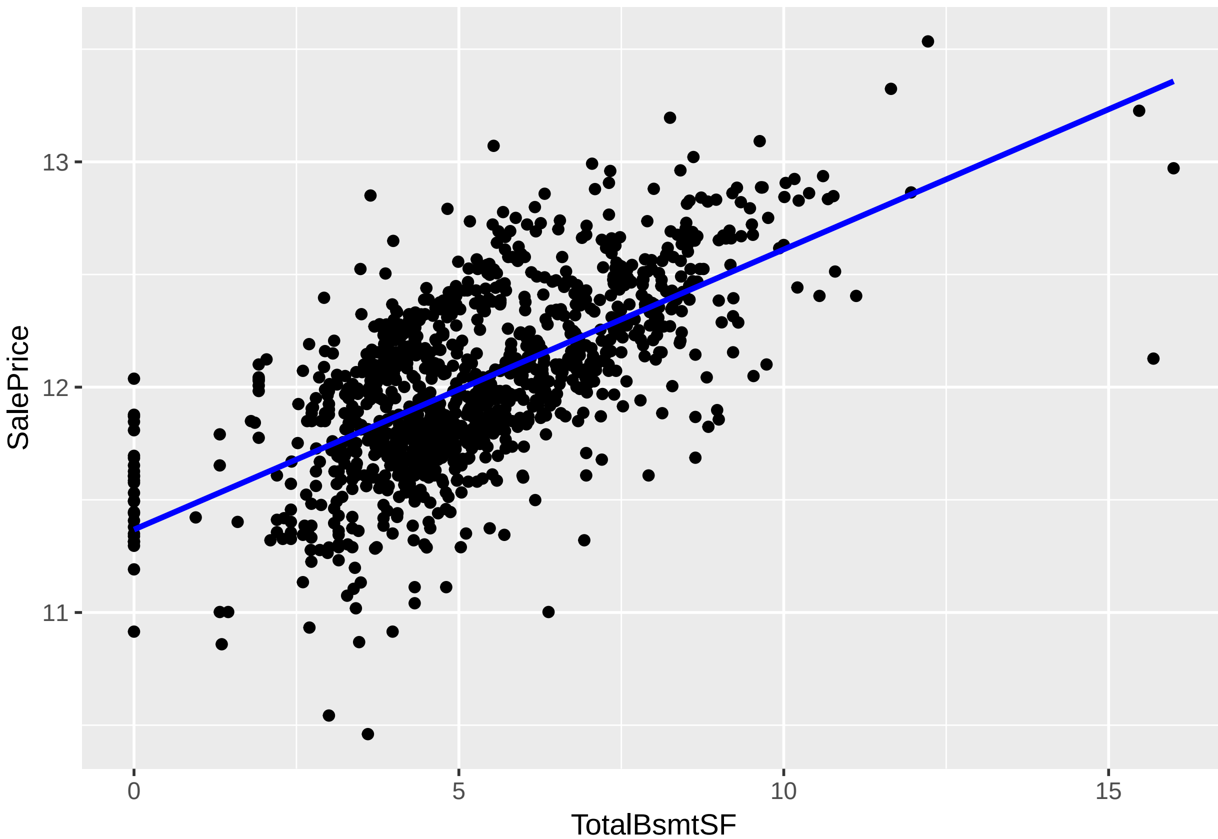
A positive correlation is observed between `TotalBsmtSF` and `SalePrice`, with just a single outlier that we might imagine corresponds to a basement that is over-large for an unsavoury reason.

```
ggplot(data=train, aes(x=TotalBsmtSF, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks = seq(0, 20, by=1), labels = comma)
```



We remove the property with the lurking, sub-grade menace, and also scale the variable to bring it into the same order of magnitude as the other variables considered so far:

```
train <- train[train$TotalBsmtSF < 4000, ]
train$TotalBsmtSF <- (train$TotalBsmtSF / 200)

ggplot(data=train, aes(x=TotalBsmtSF, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks = seq(0, 20, by=1), labels = comma)
```



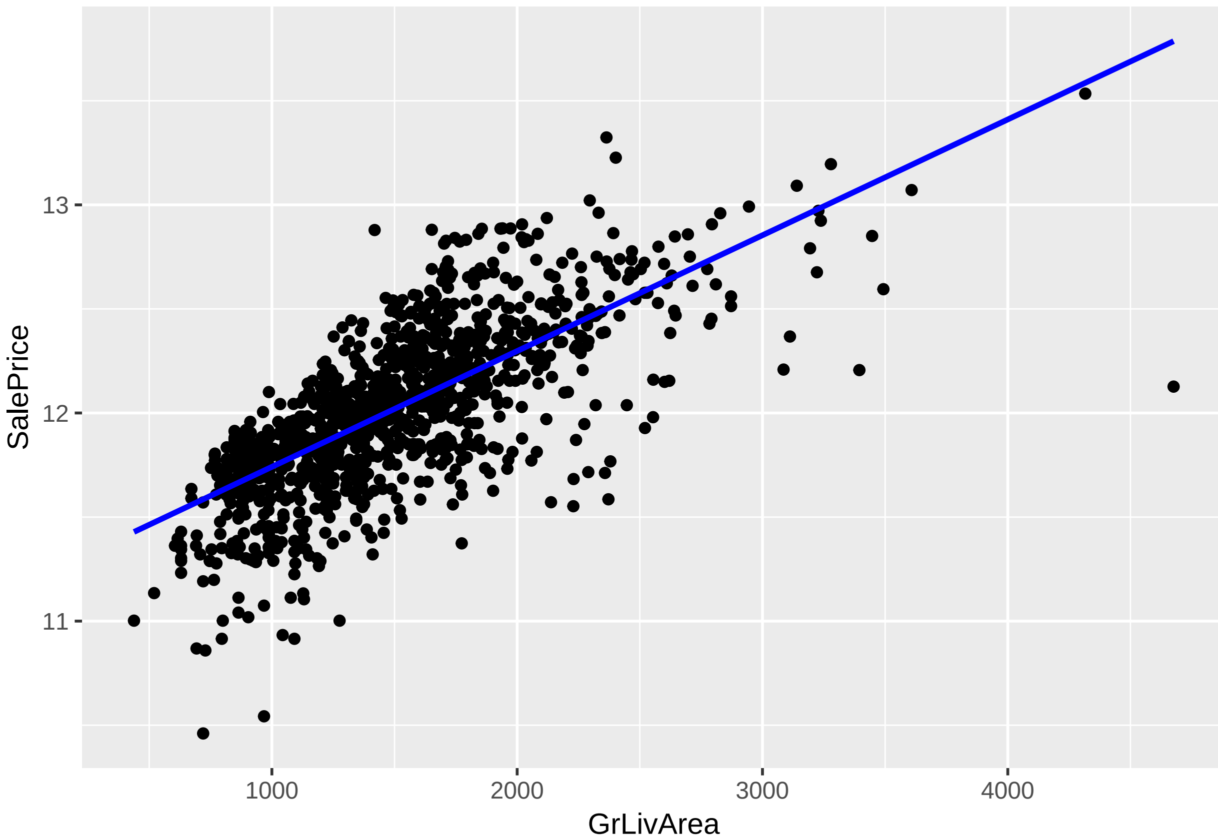
We scale the corresponding variable in the test set in the same manner.

```
test$TotalBsmtSF <- (test$TotalBsmtSF / 200)
```

GrLivArea

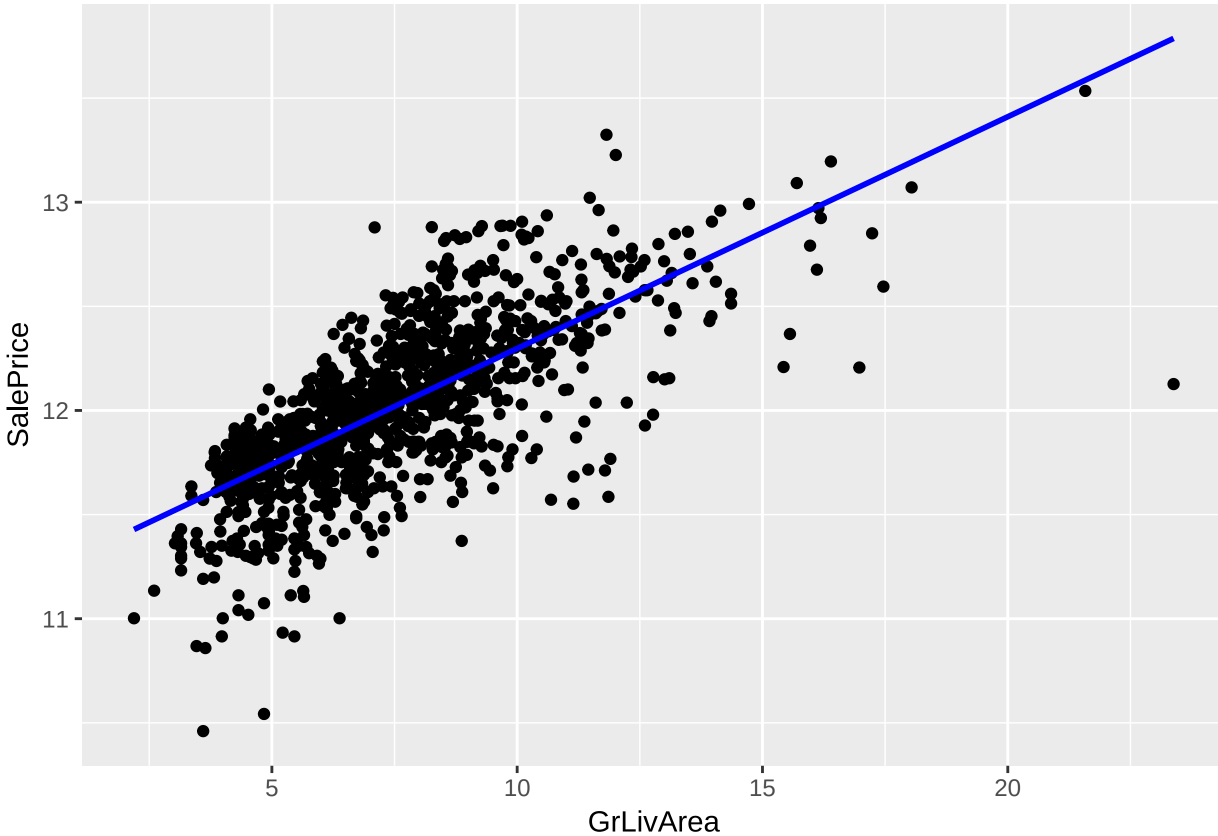
There is a strong linear correlation observed between `GrLivArea` and `SalePrice`. The properties with a living area over 4,000 square feet seem to be outliers, but we don't expect them to exert too much influence over the model so we'll pretend we didn't notice.

```
ggplot(data=train, aes(x=GrLivArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



We scale the values in the training set to bring them into the same order of magnitude as the other variables:

```
train$GrLivArea = (train$GrLivArea / 200)

ggplot(data=train, aes(x=GrLivArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



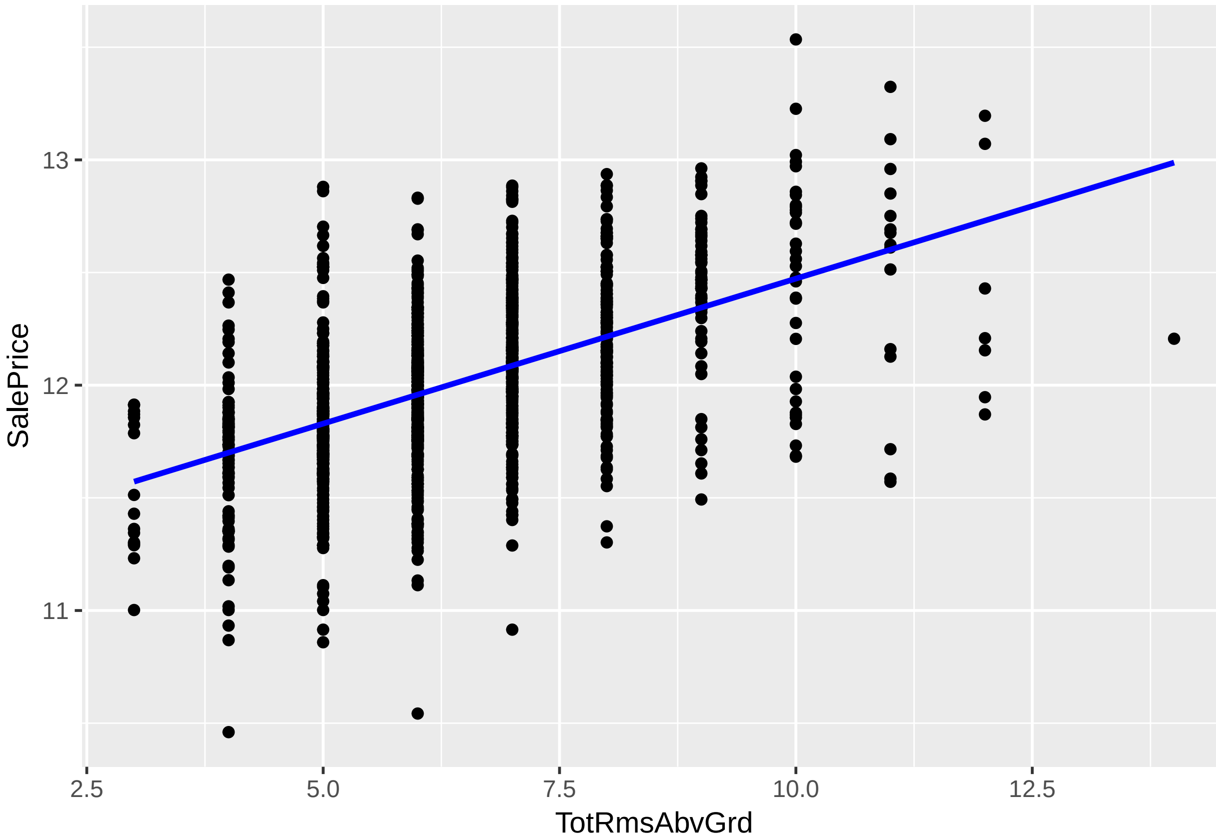
We apply the same scaling transformation to the test set:

```
test$GrLivArea = (test$GrLivArea / 200)
```

TotRmsAbvGrd

There is a strong linear correlation observed between `TotRmsAbvGrd` and `SalePrice`.

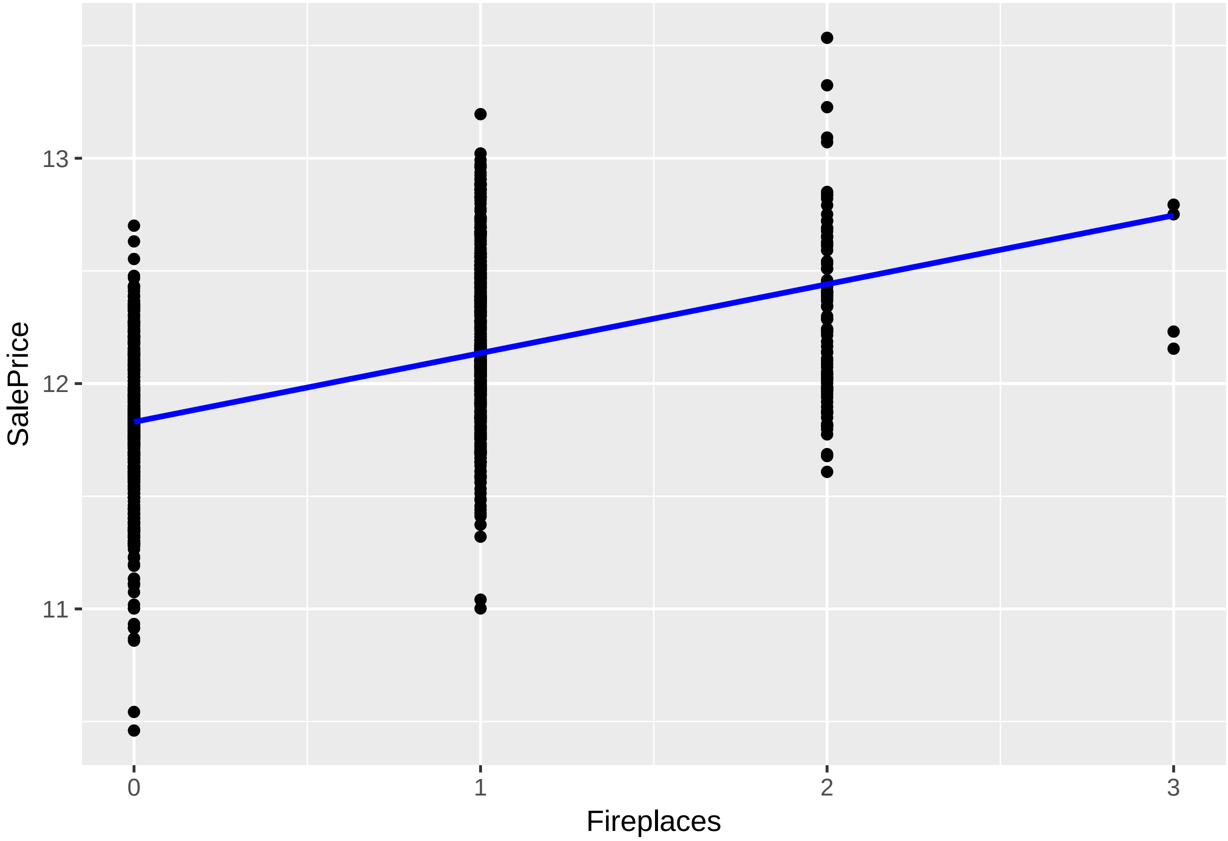
```
ggplot(data=train, aes(x=TotRmsAbvGrd, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



Fireplaces

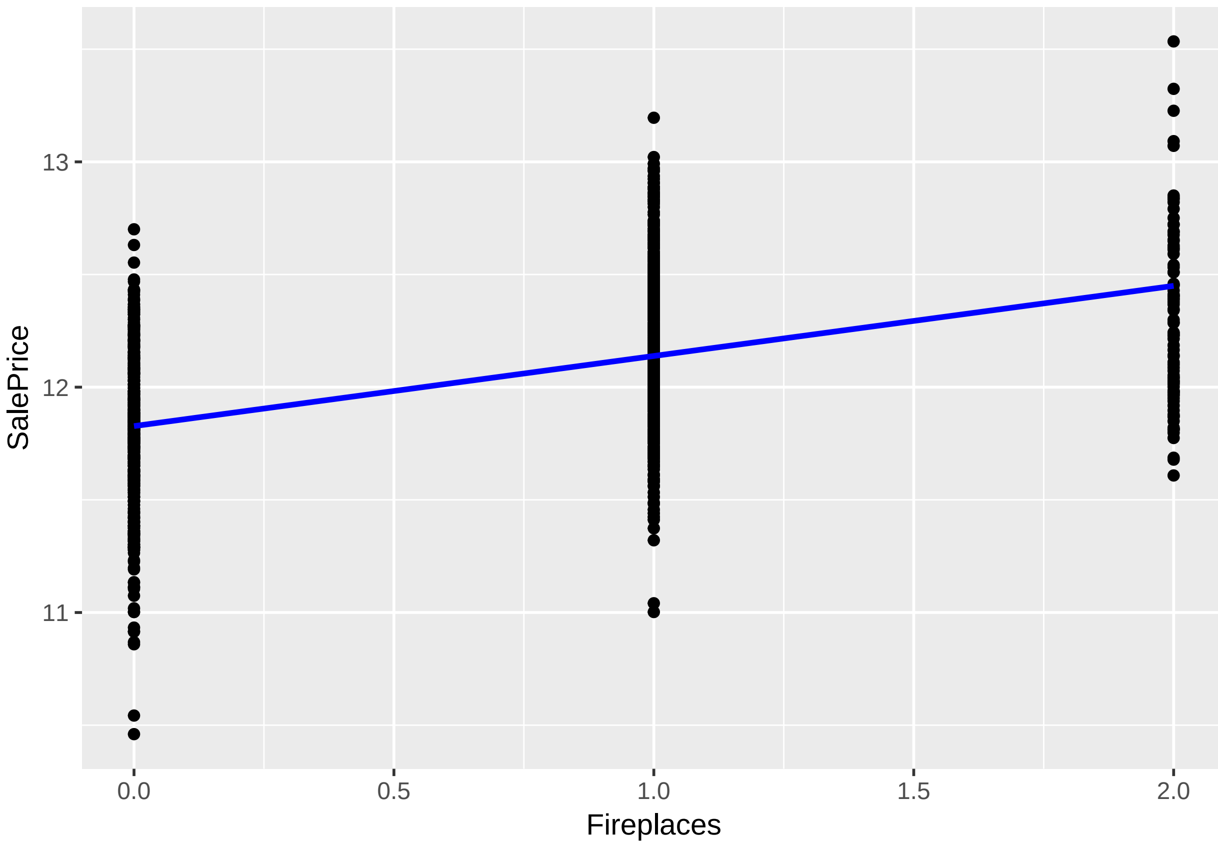
We observe a plausible correlation between the number of fireplaces and the sale price, although properties with three fireplaces seem to be outliers. People in Iowa like to burn things, but not *that* much.

```
ggplot(data=train, aes(x=Fireplaces, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



We shall remove the pyromaniac palaces from the training set:

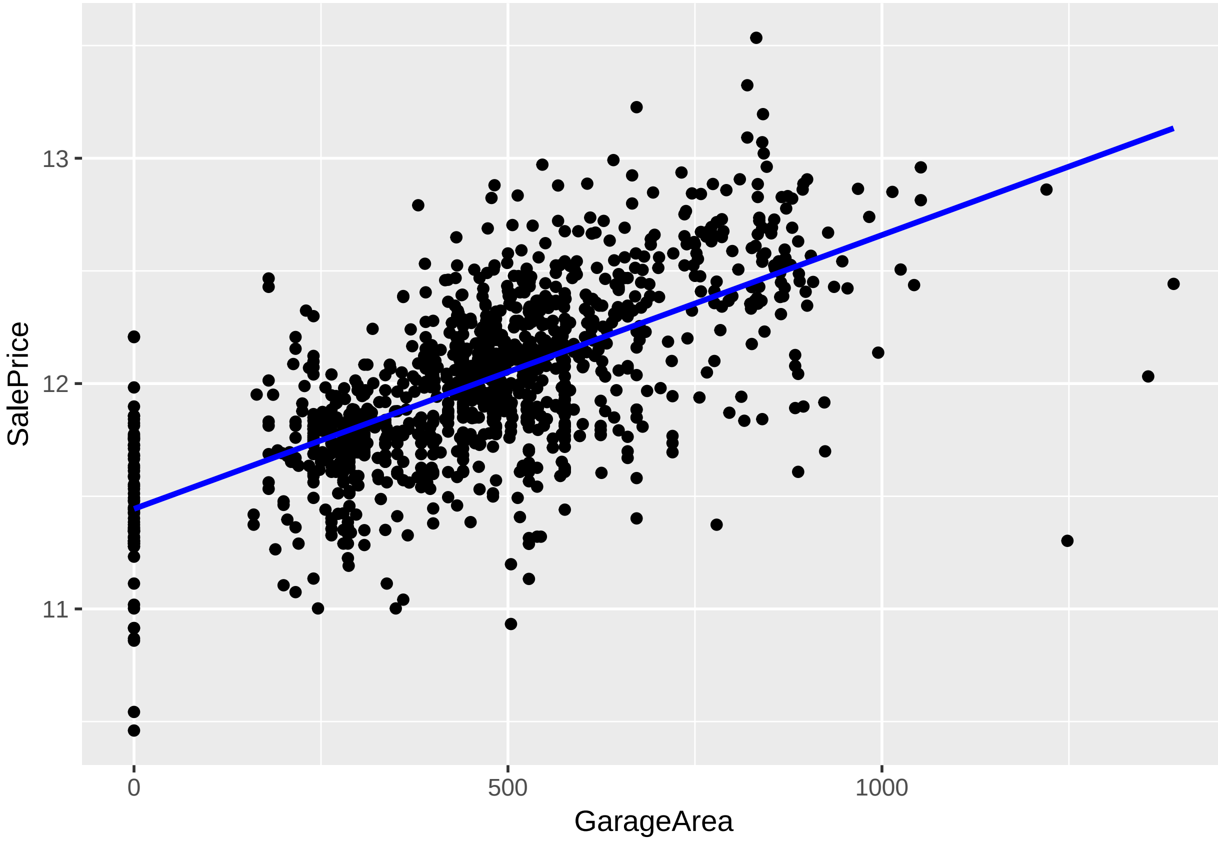
```
train <- train[train$Fireplaces < 3, ]

ggplot(data=train, aes(x=Fireplaces, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```


13
12
11
SalePrice
0.0
0.5
1.0
1.5
2.0
Fireplaces

GarageArea

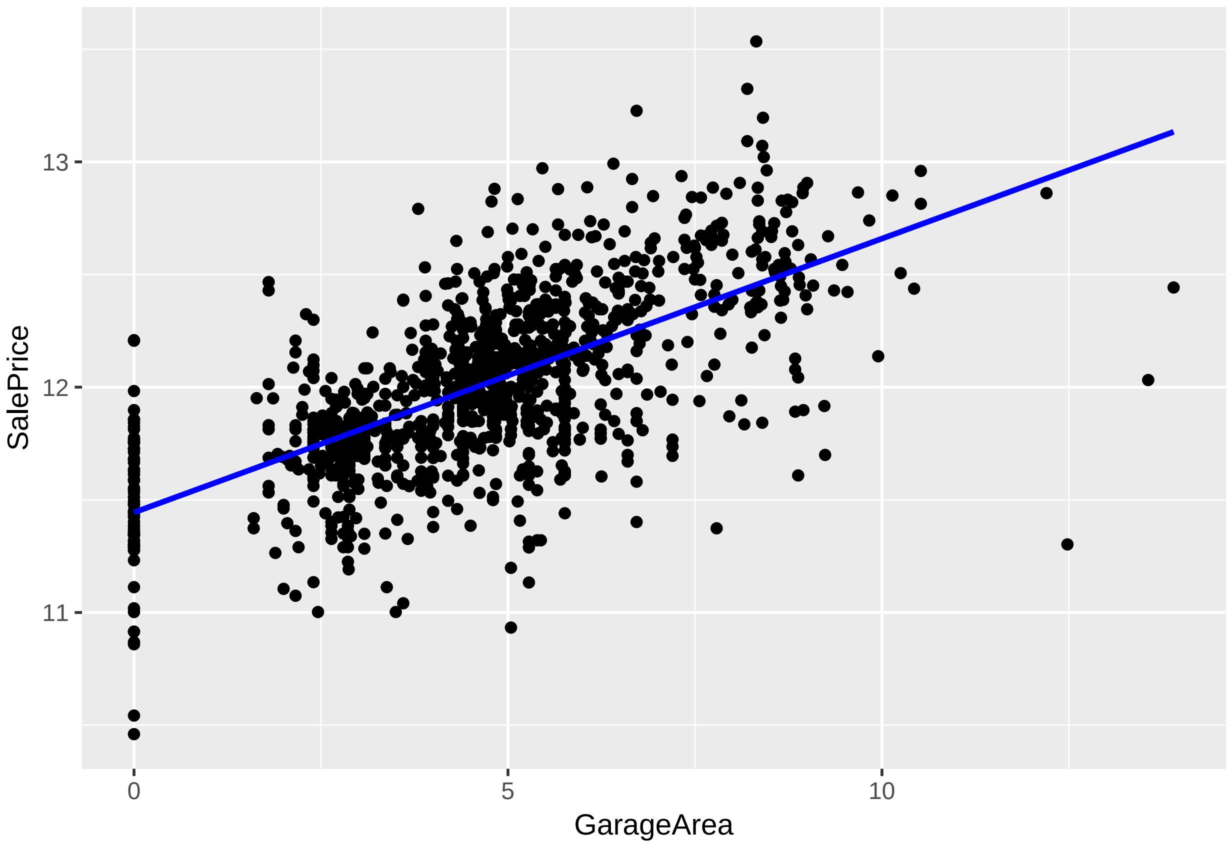
Properties with more garage space seem to command higher prices. The extremely large garages seem to have a lower impact on price, but there are not so many batcaves that we expect them to cause trouble.

```
ggplot(data=train, aes(x=GarageArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



We scale the values down into the same order of magnitude as the others:

```
train$GarageArea = (train$GarageArea / 100)

ggplot(data=train, aes(x=GarageArea, y=SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, colour = "blue", aes(group = 1)) +
  scale_y_continuous(breaks= seq(0, 20, by=1), labels = comma)
```



We apply the same transform to the test set:

```
test$GarageArea = (test$GarageArea / 100)
```

Selected Algorithms

Multivariate Regression

Multivariate linear regression is a method of supervised regression, used to predict a numerical outcome from a set of observations. In this exercise we have identified seven features (`LotArea`, `TotalBsmtSF`, `GrLivArea`, `TotRmsAbvGrd`, `Fireplaces`, `GarageArea` and `newBathrooms`); we have eliminated a small number of outliers and applied a logarithmic transform to some variables with the result that each of those features is observed to have a (different) linear relationship with `SalePrice`. Consequently, we will build and test a multivariate regression model with no further transforms and assess its goodness of fit.

```
modelLR <- lm(SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
  TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, train)
summary(modelLR)
```

```
##
## Call:
## lm(formula = SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
##     TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.69117 -0.08561  0.02233  0.11509  0.64572
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.689696   0.101211 105.618  < 2e-16 ***
## LotArea       0.025158   0.011871   2.119   0.0343 *
## TotalBsmtSF   0.054865   0.003289  16.682  < 2e-16 ***
## GrLivArea     0.045114   0.004686   9.628  < 2e-16 ***
## TotRmsAbvGrd -0.007894   0.006152  -1.283   0.1997
## Fireplaces    0.066671   0.010325   6.457 1.57e-10 ***
## GarageArea    0.045799   0.003228  14.187  < 2e-16 ***
## newBathrooms  0.112259   0.008135  13.800  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1878 on 1152 degrees of freedom
## Multiple R-squared:  0.7711, Adjusted R-squared:  0.7697
## F-statistic: 554.3 on 7 and 1152 DF,  p-value: < 2.2e-16
```

```
predictLR <- predict(modelLR, test)

summary(test$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.47   11.81   12.01   12.05   12.29   13.52
```

```
summary(predictLR)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.08   11.79   12.04   12.07   12.27   13.59
```

```
# rmse
RMSE <- function(x, y) {
  a <- sqrt(mean((x - y)^2))
```

```r
  return(a)
}

RMSE(test$SalePrice, predictLR)
```

```
## [1] 0.2021288
```

Support Vector Regression

Support Vector Regression (SVR) is another method of supervised regression. SVR is an adaptation of Support Vector Machines for function estimation, and is built around analogous hyperparameters, of which we are principally concerned with the soft margin loss setting ϵ, an acceptable error in the resulting regression model. We make no attempt to tune the default parameters in the model used here.

```r
modelSVR <- svm(SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
  TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, train)
summary(modelSVR)
```

```
##
## Call:
## svm(formula = SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
##     TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1428571
##     epsilon:  0.1
##
##
## Number of Support Vectors:  865
```

```r
predictSVR <- predict(modelSVR, test)

summary(test$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.47   11.81   12.01   12.05   12.29   13.52
```

```r
summary(predictSVR)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.36   11.81   12.07   12.08   12.30   12.99
```

```r
RMSE(test$SalePrice, predictSVR)
```

```
## [1] 0.2018896
```

Regression Trees

Decision trees attempt to classify an observation in the form of a target variable based on a set of input variables. They take the form of a directed graph where each interior node corresponds to a decision made on the basis of an input variable. A regression tree is a decision tree whose target variable is continuously variable, as is the case here. The training algorithm used by the library shown below uses recursive partitioning with an exit condition based on the target observation in the training set; the decision tree can then be used with a test dataset to produce predictions.

```
modelRT <- rpart(SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
  TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train, method = "anova")
summary(modelRT)
```

```
## Call:
## rpart(formula = SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
##     TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train,
##     method = "anova")
##   n= 1160
##
##            CP nsplit rel error    xerror       xstd
## 1  0.39600505      0 1.0000000 1.0023453 0.04657561
## 2  0.10299019      1 0.6039950 0.6138342 0.03018120
## 3  0.06953308      2 0.5010048 0.5348085 0.02771944
## 4  0.03088007      3 0.4314717 0.4670606 0.02467142
## 5  0.02990381      4 0.4005916 0.4368430 0.02365415
## 6  0.02002912      5 0.3706878 0.4050782 0.02158609
## 7  0.01524533      6 0.3506587 0.3997837 0.02168006
## 8  0.01495805      7 0.3354134 0.3896463 0.02177584
## 9  0.01287667      8 0.3204553 0.3817897 0.02164414
## 10 0.01169808      9 0.3075786 0.3625696 0.02097233
## 11 0.01000000     10 0.2958806 0.3577442 0.02157218
##
## Variable importance
##    GrLivArea   GarageArea TotRmsAbvGrd newBathrooms   Fireplaces
##           27           17           17           12           10
##  TotalBsmtSF      LotArea
##            9            8
##
## Node number 1: 1160 observations,    complexity param=0.396005
##   mean=12.01482, MSE=0.1529746
##   left son=2 (554 obs) right son=3 (606 obs)
##   Primary splits:
##       GrLivArea    < 7.0875   to the left,  improve=0.3960050, (0 missing)
##       newBathrooms < 2.5      to the left,  improve=0.3486078, (0 missing)
##       GarageArea   < 3.87     to the left,  improve=0.3017753, (0 missing)
##       TotalBsmtSF  < 5.5175   to the left,  improve=0.3009318, (0 missing)
##       Fireplaces   < 0.5      to the left,  improve=0.2597044, (0 missing)
##   Surrogate splits:
##       TotRmsAbvGrd < 6.5      to the left,  agree=0.844, adj=0.673, (0 split)
##       newBathrooms < 2.5      to the left,  agree=0.750, adj=0.477, (0 split)
##       Fireplaces   < 0.5      to the left,  agree=0.718, adj=0.410, (0 split)
##       GarageArea   < 3.59     to the left,  agree=0.672, adj=0.314, (0 split)
##       LotArea      < 9.117237 to the left,  agree=0.667, adj=0.303, (0 split)
##
## Node number 2: 554 observations,    complexity param=0.06953308
##   mean=11.7574, MSE=0.07841395
##   left son=4 (358 obs) right son=5 (196 obs)
##   Primary splits:
##       TotalBsmtSF  < 5.1975   to the left,  improve=0.28403090, (0 missing)
##       GarageArea   < 2.91     to the left,  improve=0.24027770, (0 missing)
```

```
##      newBathrooms < 1.5      to the left,  improve=0.23513110, (0 missing)
##      GrLivArea    < 5.6575   to the left,  improve=0.19279770, (0 missing)
##      Fireplaces   < 0.5      to the left,  improve=0.09244042, (0 missing)
##   Surrogate splits:
##      GrLivArea    < 5.1975   to the left,  agree=0.724, adj=0.219, (0 split)
##      newBathrooms < 2.5      to the left,  agree=0.702, adj=0.158, (0 split)
##      Fireplaces   < 0.5      to the left,  agree=0.666, adj=0.056, (0 split)
##      LotArea      < 9.42641  to the left,  agree=0.659, adj=0.036, (0 split)
##      GarageArea   < 4.325    to the left,  agree=0.653, adj=0.020, (0 split)
##
## Node number 3: 606 observations,    complexity param=0.1029902
##   mean=12.25015, MSE=0.105178
##   left son=6 (406 obs) right son=7 (200 obs)
##   Primary splits:
##      GarageArea   < 6.055    to the left,  improve=0.2867315, (0 missing)
##      TotalBsmtSF  < 5.2625   to the left,  improve=0.2841444, (0 missing)
##      GrLivArea    < 9.65     to the left,  improve=0.1714049, (0 missing)
##      newBathrooms < 2.5      to the left,  improve=0.1629443, (0 missing)
##      Fireplaces   < 0.5      to the left,  improve=0.1347779, (0 missing)
##   Surrogate splits:
##      TotalBsmtSF  < 7.685    to the left,  agree=0.738, adj=0.205, (0 split)
##      GrLivArea    < 11.3425  to the left,  agree=0.719, adj=0.150, (0 split)
##      TotRmsAbvGrd < 8.5      to the left,  agree=0.686, adj=0.050, (0 split)
##      newBathrooms < 4.5      to the left,  agree=0.675, adj=0.015, (0 split)
##
## Node number 4: 358 observations,    complexity param=0.02990381
##   mean=11.64697, MSE=0.06482154
##   left son=8 (68 obs) right son=9 (290 obs)
##   Primary splits:
##      GarageArea   < 2.27     to the left,  improve=0.22866590, (0 missing)
##      GrLivArea    < 3.8275   to the left,  improve=0.15661190, (0 missing)
##      newBathrooms < 1.5      to the left,  improve=0.14621680, (0 missing)
##      TotalBsmtSF  < 2.7325   to the left,  improve=0.14441400, (0 missing)
##      LotArea      < 8.829345 to the left,  improve=0.05114143, (0 missing)
##   Surrogate splits:
##      GrLivArea < 3.255    to the left,  agree=0.816, adj=0.029, (0 split)
##
## Node number 5: 196 observations
##   mean=11.95909, MSE=0.04028845
##
## Node number 6: 406 observations,    complexity param=0.03088007
##   mean=12.12826, MSE=0.06637386
##   left son=12 (124 obs) right son=13 (282 obs)
##   Primary splits:
##      GarageArea   < 4.08     to the left,  improve=0.20334450, (0 missing)
##      TotalBsmtSF  < 5.45     to the left,  improve=0.19265880, (0 missing)
##      newBathrooms < 2.5      to the left,  improve=0.18453800, (0 missing)
##      Fireplaces   < 0.5      to the left,  improve=0.11270150, (0 missing)
##      GrLivArea    < 9.6175   to the left,  improve=0.08325704, (0 missing)
##   Surrogate splits:
##      newBathrooms < 2.5      to the left,  agree=0.719, adj=0.081, (0 split)
##      TotalBsmtSF  < 1.98     to the left,  agree=0.709, adj=0.048, (0 split)
##      GrLivArea    < 14.895   to the right, agree=0.697, adj=0.008, (0 split)
##
```

```
## Node number 7: 200 observations,    complexity param=0.02002912
##   mean=12.49758, MSE=0.09257222
##   left son=14 (84 obs) right son=15 (116 obs)
##   Primary splits:
##       GrLivArea    < 9.09     to the left,  improve=0.1919679, (0 missing)
##       TotalBsmtSF  < 8.2025   to the left,  improve=0.1868778, (0 missing)
##       Fireplaces   < 0.5      to the left,  improve=0.1625503, (0 missing)
##       newBathrooms < 2.5      to the left,  improve=0.1338120, (0 missing)
##       TotRmsAbvGrd < 9.5      to the left,  improve=0.1307306, (0 missing)
##   Surrogate splits:
##       TotRmsAbvGrd < 7.5      to the left,  agree=0.825, adj=0.583, (0 split)
##       TotalBsmtSF  < 7.4175   to the right, agree=0.685, adj=0.250, (0 split)
##       newBathrooms < 2.5      to the left,  agree=0.685, adj=0.250, (0 split)
##       Fireplaces   < 0.5      to the left,  agree=0.670, adj=0.214, (0 split)
##       LotArea      < 9.108744 to the left,  agree=0.660, adj=0.190, (0 split)
##
## Node number 8: 68 observations
##   mean=11.39555, MSE=0.07940848
##
## Node number 9: 290 observations,    complexity param=0.01495805
##   mean=11.70593, MSE=0.04310306
##   left son=18 (39 obs) right son=19 (251 obs)
##   Primary splits:
##       TotalBsmtSF  < 2.7325   to the left,  improve=0.21234700, (0 missing)
##       newBathrooms < 1.5      to the left,  improve=0.15809320, (0 missing)
##       GrLivArea    < 3.8275   to the left,  improve=0.11472150, (0 missing)
##       GarageArea   < 3.99     to the left,  improve=0.09629875, (0 missing)
##       LotArea      < 7.640114 to the left,  improve=0.08051376, (0 missing)
##   Surrogate splits:
##       LotArea   < 7.640114 to the left,  agree=0.876, adj=0.077, (0 split)
##       GrLivArea < 2.875    to the left,  agree=0.872, adj=0.051, (0 split)
##
## Node number 12: 124 observations
##   mean=11.95306, MSE=0.05540806
##
## Node number 13: 282 observations,    complexity param=0.01524533
##   mean=12.2053, MSE=0.0517642
##   left son=26 (145 obs) right son=27 (137 obs)
##   Primary splits:
##       TotalBsmtSF  < 5.2975   to the left,  improve=0.18532570, (0 missing)
##       newBathrooms < 3.5      to the left,  improve=0.07882496, (0 missing)
##       Fireplaces   < 0.5      to the left,  improve=0.07775615, (0 missing)
##       GarageArea   < 4.985    to the left,  improve=0.06979215, (0 missing)
##       GrLivArea    < 9.6175   to the left,  improve=0.06229510, (0 missing)
##   Surrogate splits:
##       GarageArea   < 4.775    to the left,  agree=0.642, adj=0.263, (0 split)
##       LotArea      < 9.369604 to the left,  agree=0.631, adj=0.241, (0 split)
##       TotRmsAbvGrd < 6.5      to the right, agree=0.613, adj=0.204, (0 split)
##       GrLivArea    < 8.1275   to the right, agree=0.599, adj=0.175, (0 split)
##       newBathrooms < 2.5      to the right, agree=0.589, adj=0.153, (0 split)
##
## Node number 14: 84 observations,    complexity param=0.01169808
##   mean=12.34092, MSE=0.07658596
##   left son=28 (24 obs) right son=29 (60 obs)
```

```
##   Primary splits:
##       TotalBsmtSF  < 7.335    to the left,  improve=0.32267360, (0 missing)
##       GarageArea   < 7.335    to the left,  improve=0.12493230, (0 missing)
##       newBathrooms < 2.5      to the left,  improve=0.10105710, (0 missing)
##       Fireplaces   < 0.5      to the left,  improve=0.09018083, (0 missing)
##       LotArea      < 9.348144 to the left,  improve=0.08324983, (0 missing)
##   Surrogate splits:
##       GarageArea   < 6.88     to the left,  agree=0.798, adj=0.292, (0 split)
##       GrLivArea    < 7.2725   to the left,  agree=0.750, adj=0.125, (0 split)
##       LotArea      < 8.250908 to the left,  agree=0.738, adj=0.083, (0 split)
##       newBathrooms < 3.5      to the right, agree=0.726, adj=0.042, (0 split)
##
## Node number 15: 116 observations,    complexity param=0.01287667
##   mean=12.61102, MSE=0.07350901
##   left son=30 (67 obs) right son=31 (49 obs)
##   Primary splits:
##       TotalBsmtSF  < 6.96     to the left,  improve=0.2679676, (0 missing)
##       Fireplaces   < 0.5      to the left,  improve=0.1234853, (0 missing)
##       GarageArea   < 7.265    to the left,  improve=0.1106920, (0 missing)
##       TotRmsAbvGrd < 9.5      to the left,  improve=0.1071147, (0 missing)
##       GrLivArea    < 13.355   to the left,  improve=0.1001320, (0 missing)
##   Surrogate splits:
##       LotArea      < 9.56251  to the left,  agree=0.655, adj=0.184, (0 split)
##       GrLivArea    < 13.9225  to the left,  agree=0.629, adj=0.122, (0 split)
##       TotRmsAbvGrd < 6.5      to the right, agree=0.612, adj=0.082, (0 split)
##       GarageArea   < 6.125    to the right, agree=0.603, adj=0.061, (0 split)
##       newBathrooms < 4.5      to the left,  agree=0.595, adj=0.041, (0 split)
##
## Node number 18: 39 observations
##   mean=11.46322, MSE=0.05004452
##
## Node number 19: 251 observations
##   mean=11.74364, MSE=0.03144955
##
## Node number 26: 145 observations
##   mean=12.11009, MSE=0.0297574
##
## Node number 27: 137 observations
##   mean=12.30606, MSE=0.05530941
##
## Node number 28: 24 observations
##   mean=12.09236, MSE=0.05838933
##
## Node number 29: 60 observations
##   mean=12.44034, MSE=0.04926744
##
## Node number 30: 67 observations
##   mean=12.49099, MSE=0.04307921
##
## Node number 31: 49 observations
##   mean=12.77513, MSE=0.06848503
```

```r
predictRT <- predict(modelRT, test)
```

```r
summary(test$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.47   11.81   12.01   12.05   12.29   13.52
```

```r
summary(predictRT)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.40   11.74   11.96   12.07   12.31   12.78
```

```r
RMSE(test$SalePrice, predictRT)
```

```
## [1] 0.2548018
```

Random Forests

The Random Forest algorithm is a development of those used to build regression trees that are able to correct for overfitting to the training set by constructing a large number of decision trees during training and producing an average of individual trees.

Parameters for the for this random forest model were copied from similar descriptions of kernels found at Kaggle that used the same library[5].

```r
modelRF <- randomForest(SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
  TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train, method = "anova",
  ntree = 300, replace = FALSE, nodesize = 1, importance = TRUE)
summary(modelRF)
```

```
##                 Length Class  Mode
## call               8   -none- call
## type               1   -none- character
## predicted       1160   -none- numeric
## mse              300   -none- numeric
## rsq              300   -none- numeric
## oob.times       1160   -none- numeric
## importance        14   -none- numeric
## importanceSD       7   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               1160   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```r
predictRF <- predict(modelRF, test)

summary(test$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.47   11.81   12.01   12.05   12.29   13.52
```

[5] e.g. see https://www.kaggle.com/myonin/prediction-of-house-prices-3-methods

```r
summary(predictRF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.17   11.81   12.04   12.06   12.28   13.19
```

```r
RMSE(test$SalePrice, predictRF)
```

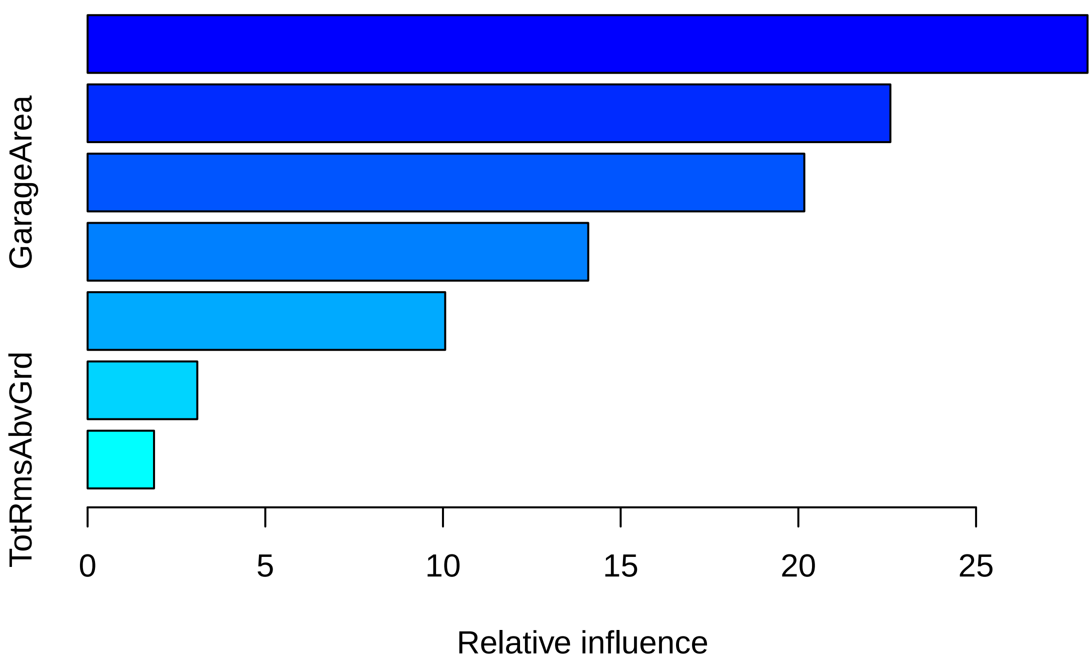
```
## [1] 0.1895513
```

Gradiant Boosting Regression

Gradiant boosting machines are yet more examples of algorithms based on an ensemble of individually-weaker decision trees, generalising their output by the optimisation of a differentiable loss function.

Parameters for the for this GBM model were copied from similar descriptions of kernels found at Kaggle that used the same library[6].

```r
modelGBM <- gbm(SalePrice ~ LotArea + TotalBsmtSF + GrLivArea +
  TotRmsAbvGrd + Fireplaces + GarageArea + newBathrooms, data = train,
  distribution = "laplace", shrinkage = 0.05, interaction.depth = 5,
  bag.fraction = 0.66, n.minobsinnode = 1, cv.folds = 100, keep.data = FALSE,
  verbose = FALSE, n.trees = 300)
summary(modelGBM)
```




```
##                       var   rel.inf
## GrLivArea       GrLivArea 28.137761
## TotalBsmtSF   TotalBsmtSF 22.589239
## GarageArea     GarageArea 20.167544
## newBathrooms newBathrooms 14.087041
## LotArea           LotArea 10.062128
## Fireplaces     Fireplaces  3.086723
## TotRmsAbvGrd TotRmsAbvGrd  1.869563
```

[6] e.g. see https://www.kaggle.com/myonin/prediction-of-house-prices-3-methods

```r
predictGBM <- predict(modelGBM, test, n.trees = 300)

summary(test$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.47   11.81   12.01   12.05   12.29   13.52
```

```r
summary(predictGBM)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.11   11.82   12.05   12.08   12.29   13.27
```

```r
RMSE(test$SalePrice, predictGBM)
```

```
## [1] 0.2002724
```

Accuracy Comparison

The source dataset used in this assignment was split arbitrarily into a training set (`train`) and a validation set (`test`) of first-seen data. Each of the models used were trained using the former and cross-validated using the latter, in effect testing the model against data that was not used in estimating it. The accuracy of the model was quantified as the root mean squared error between target variables in the validation set and the corresponding predicted values generated by each of the models.

Regression Model	RMSE
Random Forests	0.1678204
Gradiant Boosting Machine	0.1692927
Multivariate Regression	0.1789892
Support Vector Regression	0.1980624
Regression Trees	0.2230123

No attempt was made to tune most of the models, with the notable exception of the two that gave the lowest RMSE which were tuned using parameters used by a Kaggle competitor working on the same dataset (albeit a dataset that was likely transformed differently, and probably not reduced with such viciousness). It seems entirely plausible that other models would see lower RMSE than that observed if some effort was made to tune them.