HedonicModel:SND

LuisM

3/22/2021

VARIABLES SELECTION

Loading the data

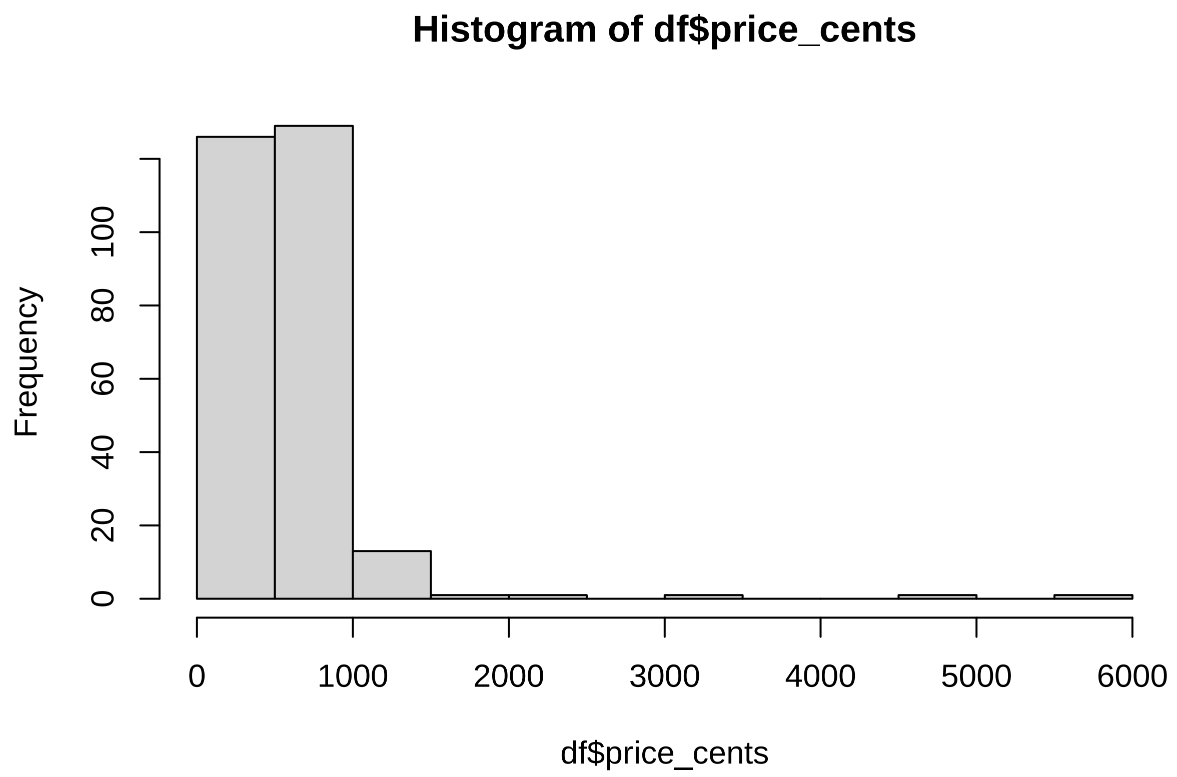
```
library(readxl)
df <- read_excel("/Users/Unimooc/Dropbox/2021/Directorio R/SpaceNextDoor/NextDoor/Optimal pricing/NextDo
                  sheet = "Data1")
```

```
head(df)
```

Visualization

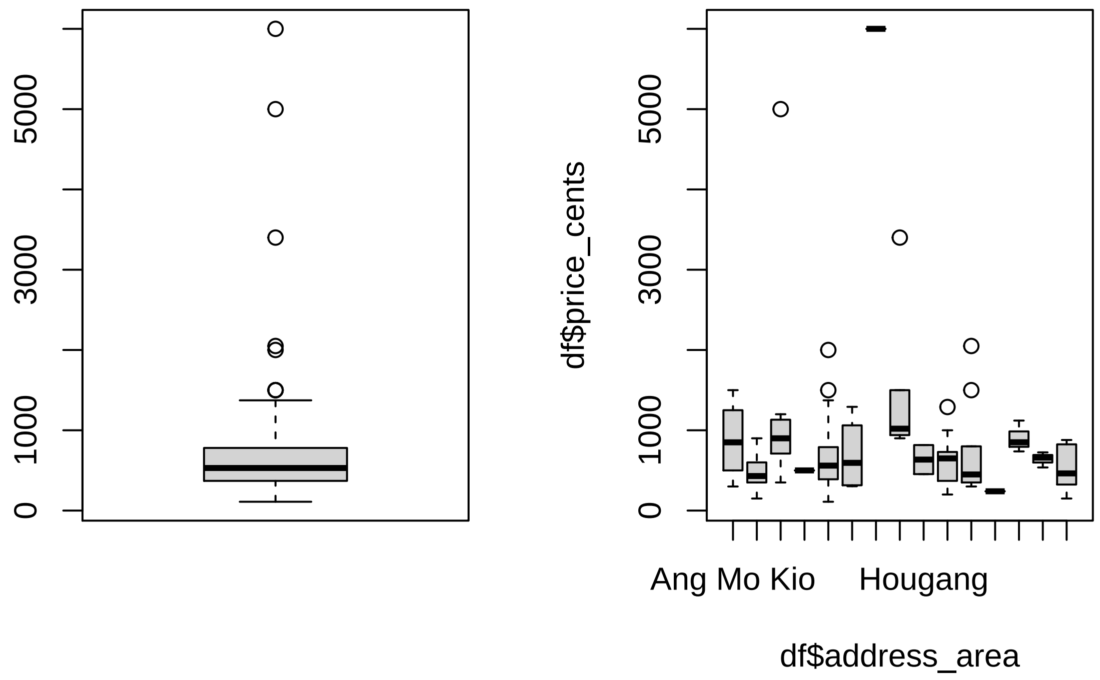
```
library(ggplot2)
library(corrplot)
library(tidyverse)
library(MASS)
```

```
hist(df$price_cents)
```



```
#The histogram shows a positive skweness (positioned to the left). There are many outliers that should
```

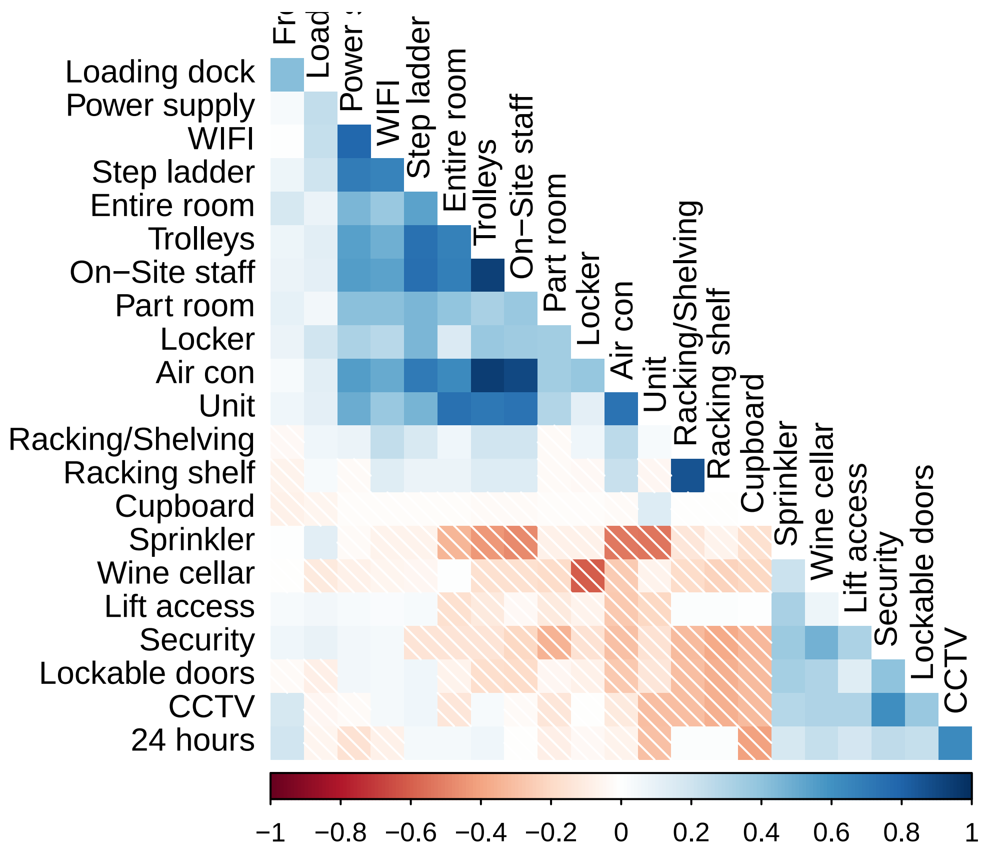
```
par(mfrow=c(1,2))

  boxplot(df$price_cents)
  boxplot(df$price_cents ~ df$address_area)
```



```
#The Boxplot confirms the presence of outliers caused by some isolated data in a few districts. As we g
```

```
df.cor <- cor(df[,c(21:42)], method = "kendall")
round(df.cor, digits = 1)
```

```
corrplot(df.cor, method ="shade",
         tl.col ="black",
         order = "AOE", type = "lower", diag = F)
```



```
#The correlation matrix indicates the presence of strong autocorrelation between some variables. We sho
```

Training and test sample division

```
library(dplyr)
library(caret)
```

```
  df.sel <- df[,-c(1,3,4,6:20)]

  set.seed(2021)
  dfPartition <- createDataPartition(y = df.sel$price_cents,
                                     p = 0.7, list = F)

  Training <- df.sel[dfPartition,]
```

```
## Warning: The ‘i‘ argument of ‘‘[‘()‘ can’t be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```
  Test <- df.sel[-dfPartition,]
```

Variables selection

AIC forward selection

```
Modelzero <- lm(price_cents~1,data=Training)
  summary(Modelzero)

  FitAll = lm(price_cents ~ ., data=Training)
  formula(FitAll)

  model.forward <- step(Modelzero,direction="forward",scope=formula(FitAll))
```

```
summary(model.forward)
```

```
##
## Call:
## lm(formula = price_cents ~ `area squared foot` + `Lift access` +
##     `Step ladder` + Locker + CCTV + `24 hours` + `Loading dock` +
##     `Free Parking` + WIFI + `On-Site staff` + `Entire room` +
##     `Lockable doors` + Unit + Sprinkler + Security + `Wine cellar` +
##     `Part room`, data = Training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -473.97  -72.02   -7.73   70.93  515.74
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           985.189    146.850   6.709 2.62e-10 ***
## `area squared foot`    55.976      2.058  27.201  < 2e-16 ***
## `Lift access`        -799.348    100.674  -7.940 2.35e-13 ***
## `Step ladder`         140.546     96.978   1.449 0.149055
## Locker                -59.443     82.170  -0.723 0.470388
## CCTV                 -325.207    180.135  -1.805 0.072739 .
## `24 hours`            493.892    163.015   3.030 0.002819 **
## `Loading dock`        131.575     25.772   5.105 8.56e-07 ***
## `Free Parking`        -91.800     25.972  -3.535 0.000523 ***
## WIFI                  -17.075     83.769  -0.204 0.838718
## `On-Site staff`       152.321     99.474   1.531 0.127510
## `Entire room`        -236.118     80.769  -2.923 0.003920 **
## `Lockable doors`      152.104     66.511   2.287 0.023400 *
## Unit                  274.955     86.167   3.191 0.001681 **
## Sprinkler             130.131     49.903   2.608 0.009903 **
## Security             -595.850    167.229  -3.563 0.000472 ***
## `Wine cellar`         219.157     91.442   2.397 0.017598 *
## `Part room`          -219.952     96.771  -2.273 0.024248 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 137.4 on 175 degrees of freedom
## Multiple R-squared:  0.9214, Adjusted R-squared:  0.9138
## F-statistic: 120.7 on 17 and 175 DF,  p-value: < 2.2e-16
```

Forward model:

price_cents ~ ‘area squared foot‘ + ‘Lift access‘ + ‘Step ladder‘ + Locker + CCTV + ‘24 hours‘ + ‘Loadi

```
predict.for.tr <- predict(model.forward, newdata = Training)

training.for.mse <- mean((predict.for.tr - Training$price_cents)^2)
paste("Training MSE error:", training.for.mse)
```

```
## [1] "Training MSE error: 17119.3585103714"
```

```
predict.for.tst <- predict(model.forward, newdata = Test)

test.for.mse <- mean((predict.for.tst - Test$price_cents)^2)
paste("Test MSE error:", test.for.mse)
```

```
## [1] "Test MSE error: 415545.240857092"
```

Summary

price_cents ~ ‘area squared foot‘ + ‘Lift access‘ + ‘Step ladder‘ + Locker + CCTV + ‘24 hours‘ + ‘Loadi

AIC backward selection

```
model.backward <- stepAIC(FitAll, trace=TRUE, direction="backward")
```

```
summary(model.backward)
```

```
##
## Call:
## lm(formula = price_cents ~ ‘area squared foot‘ + ‘Part room‘ +
##     ‘Entire room‘ + Unit + ‘Wine cellar‘ + ‘24 hours‘ + ‘Lift access‘ +
##     Security + ‘Loading dock‘ + CCTV + ‘Lockable doors‘ + ‘Free Parking‘ +
##     Trolleys + ‘Step ladder‘ + Sprinkler, data = Training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -457.06  -71.35   -7.86   69.29  527.52
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           919.675    133.102   6.910 8.45e-11 ***
## ‘area squared foot‘    55.888      2.014  27.751  < 2e-16 ***
## ‘Part room‘          -231.221     86.147  -2.684 0.007964 **
## ‘Entire room‘        -235.911     78.712  -2.997 0.003117 **
## Unit                  285.802     79.996   3.573 0.000455 ***
## ‘Wine cellar‘         292.047     53.050   5.505 1.28e-07 ***
## ‘24 hours‘            491.699    153.464   3.204 0.001608 **
## ‘Lift access‘        -735.360     93.231  -7.888 3.06e-13 ***
## Security             -657.253    136.807  -4.804 3.30e-06 ***
## `Loading dock`        131.412     25.154   5.224 4.88e-07 ***
## CCTV                 -322.249    165.985  -1.941 0.053793 .
## `Lockable doors`      140.312     64.461   2.177 0.030827 *
## `Free Parking`        -90.558     25.458  -3.557 0.000481 ***
## Trolleys              136.907     88.730   1.543 0.124626
## `Step ladder`         128.258     83.634   1.534 0.126923
## Sprinkler             130.705     48.089   2.718 0.007222 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 136.8 on 177 degrees of freedom
## Multiple R-squared:  0.9212, Adjusted R-squared:  0.9145
## F-statistic: 137.9 on 15 and 177 DF,  p-value: < 2.2e-16
```

Summary

```
price_cents ~ `area squared foot` + `Part room` + `Entire room` + Unit + `Wine cellar` + `24 hours` + `
```

```
predict.bck.tr <- predict(model.backward, newdata = Training)

training.bck.mse <- mean((predict.bck.tr - Training$price_cents)^2)
paste("Training MSE error:", training.bck.mse)
```

```
## [1] "Training MSE error: 17167.6882295733"
```

```
predict.bck.tst <- predict(model.backward, newdata = Test)

test.bck.mse <- mean((predict.bck.tst - Test$price_cents)^2)
paste("Test MSE error:", test.bck.mse)
```

```
## [1] "Test MSE error: 440903.002491439"
```

Ridge and Lasso regularizations

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

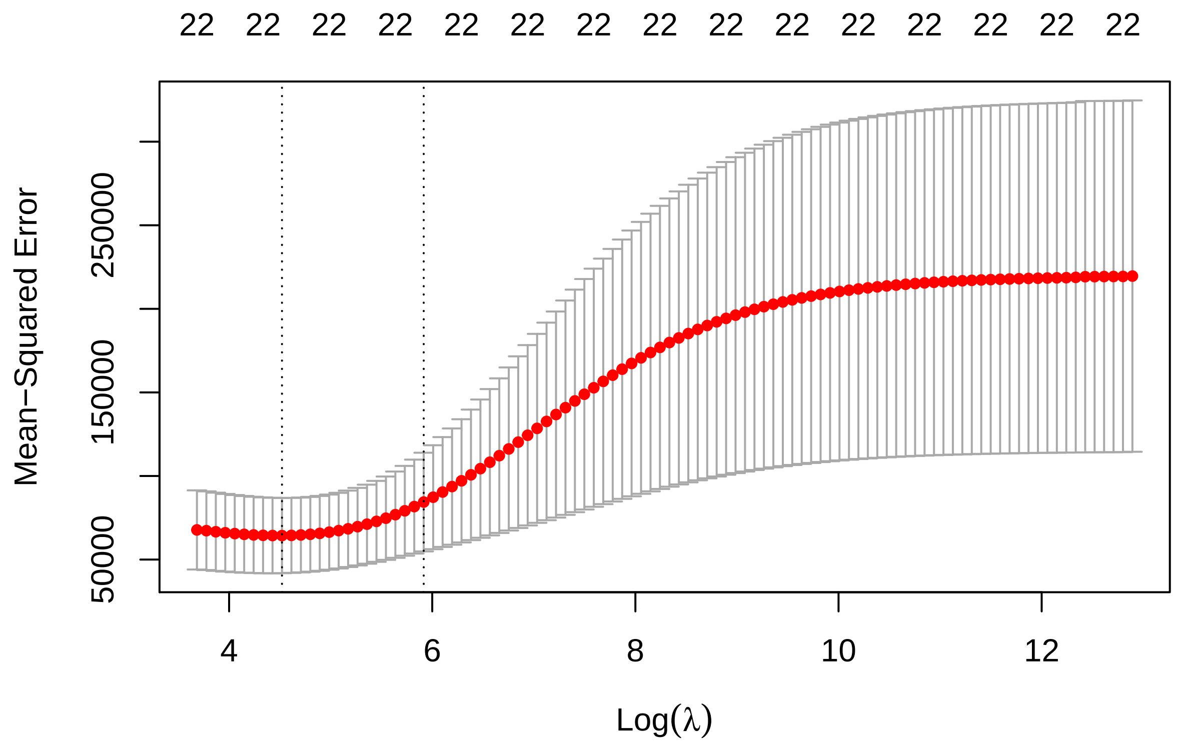
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```r
# Convert into a matrix train and test data
train.mat <- model.matrix(price_cents ~ ., data = Training)
test.mat <- model.matrix(price_cents ~ ., data = Test)
```

Ridge

```r
# Cross validation to obtain the best value of lambda. Error evolution.
cv.ridge <- cv.glmnet(x = train.mat, y = Training$price_cents, alpha = 0,
                      lambda = NULL, type.measure="mse")

plot(cv.ridge)
```



```r
paste("Best lambda:", cv.ridge$lambda.min)
```

```
## [1] "Best lambda: 91.8822861493194"
```

```r
paste("Best lambda + y sd:", cv.ridge$lambda.1se)
```

```
## [1] "Best lambda + y sd: 370.930374462024"
```

```r
# Training the model
mod.ridge.train <- glmnet(x = train.mat, y = Training$price_cents, alpha = 0,
                          lambda = cv.ridge$lambda.1se)

dim(coef(mod.ridge.train))
```

```
## [1] 25  1
```

```r
coef(mod.ridge.train, s = "lambda.1se")
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)         1050.104172
## (Intercept)            .
## `area squared foot`   29.275100
## Locker              -103.832446
## Cupboard               .
## `Racking shelf`      -30.279797
## `Part room`          -84.405881
## `Entire room`         73.527024
## Unit                 112.990315
## `Wine cellar`        131.036923
## `Air con`             39.245186
## `24 hours`           174.035224
## `Lift access`       -352.066609
## Security            -239.700388
## `Loading dock`        70.563736
## CCTV                -434.236379
## `Lockable doors`     110.067463
## `On-Site staff`        5.434037
## `Free Parking`       -24.292767
## Trolleys              10.110869
## `Step ladder`         52.541545
## `Racking/Shelving`   -30.444043
## Sprinkler            -38.295701
## `Power supply`       -18.012737
## WIFI                  14.963850
```

```r
# Training predictions
pred.ridge <- predict(mod.ridge.train, newx = train.mat)

# Training error (MSE)
tr.ridge.mse <- mean((pred.ridge - Training$price_cents)^2)
paste("Training MSE error:", tr.ridge.mse)
```

```
## [1] "Training MSE error: 50145.6072531696"
```

```r
#Test predictions: using training model
pred.test.ridge <- predict(mod.ridge.train,newx = test.mat)

test.ridge.mse <- mean((pred.test.ridge - Test$price_cents)^2)
paste("Test MSE error:",test.ridge.mse)
```

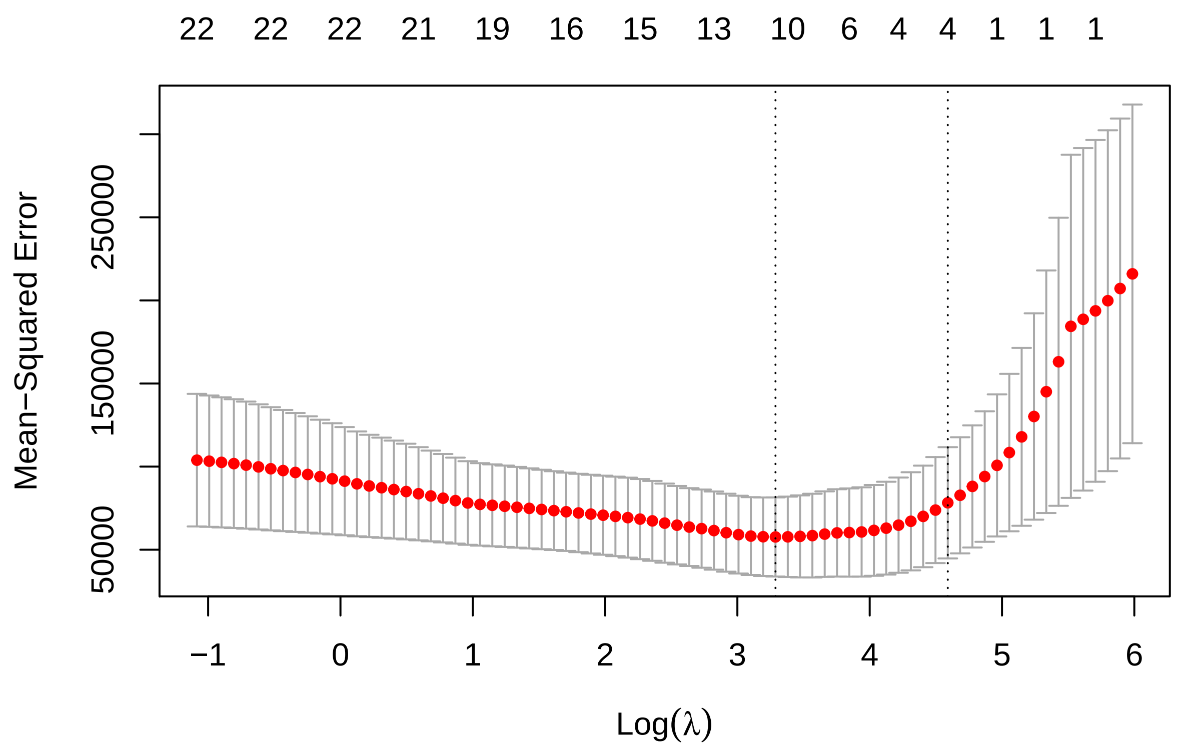
```
## [1] "Test MSE error: 413568.619418348"
```

Ridge model:

```
price_cents ~ ‘area squared foot‘ + Locker + ‘Racking shelf‘ + ‘Part room‘
+ ‘Entire room‘ + Unit + ‘Wine cellar‘ + ‘Air con‘ + ‘24 hours‘+ ‘Lift access‘
+ Security + ‘Loading dock‘ + CCTV + ‘Lockable doors‘ + ‘On-Site staff‘ + ‘Free Parking‘
+ Trolleys + ‘Step ladder‘ + ‘Racking/Shelving‘ + Sprinkler + ‘Power supply‘ +  WIFI
```

Lasso

```r
cv.lasso <- cv.glmnet(x = train.mat, y = Training$price_cents, alpha = 1,
                      lambda = NULL, type.measure="mse")

plot(cv.lasso)
```



```r
paste("Best lambda:", cv.lasso$lambda.min)
```

```
## [1] "Best lambda: 26.7841738243326"
```

```r
paste("Best lambda + y sd:", cv.lasso$lambda.1se)
```

```
## [1] "Best lambda + y sd: 98.5223630810571"
```

```r
# Training the model
mod.lasso.train <- glmnet(x = train.mat, y = Training$price_cents, alpha = 1,
                          lambda = cv.lasso$lambda.1se)

dim(coef(mod.lasso.train))
```

```
## [1] 25  1
```

```r
coef(mod.lasso.train, s = "lambda.1se")
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)           645.44705
## (Intercept)             .
## `area squared foot`    43.68657
## Locker                  .
## Cupboard                .
## `Racking shelf`         .
## `Part room`             .
## `Entire room`           .
## Unit                   45.59185
## `Wine cellar`           .
## `Air con`               .
## `24 hours`              .
## `Lift access`        -194.63918
## Security                .
## `Loading dock`          .
## CCTV                  -89.24854
## `Lockable doors`        .
## `On-Site staff`         .
## `Free Parking`          .
## Trolleys                .
## `Step ladder`           .
## `Racking/Shelving`      .
## Sprinkler               .
## `Power supply`          .
## WIFI                    .
```

```r
# Training predictions
pred.lasso <- predict(mod.lasso.train, newx = train.mat)

# Training error (MSE)
tr.lasso.mse <- mean((pred.lasso - Training$price_cents)^2)
paste("Training MSE error:", tr.lasso.mse)
```

```
## [1] "Training MSE error: 57760.6240008345"
```

```r
#Test predictions: using training model
pred.test.lasso <- predict(mod.lasso.train, newx = test.mat)

test.lasso.mse <- mean((pred.test.lasso - Test$price_cents)^2)
paste("Test MSE error:",test.lasso.mse)
```

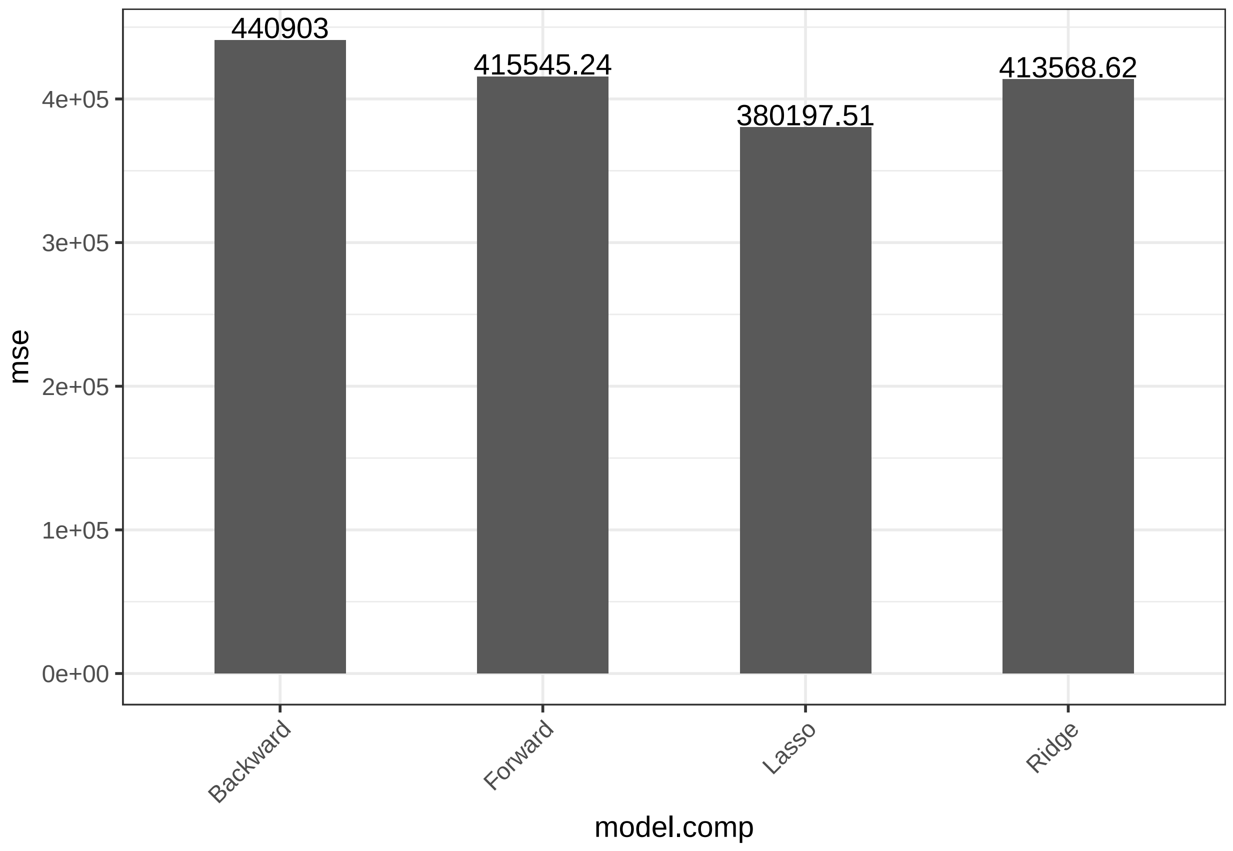
```
## [1] "Test MSE error: 380197.505939534"
```

Lasso model:

price_cents ~ ‘area squared foot‘ + Unit + ‘Lift access‘ + CCTV

Comparing results

```r
df_compar <- data.frame(model.comp = c("Forward", "Backward", "Ridge", "Lasso"),

mse = c(test.for.mse, test.bck.mse, test.ridge.mse, test.lasso.mse))

ggplot(data = df_compar, aes(x = model.comp, y = mse)) + geom_col(width = 0.5) +
    geom_text(aes(label = round(mse, 2)), vjust = -0.1) + theme_bw() + theme(axis.text.x = element_text
    hjust = 1))
```



Linear Model

```
lmodel.Train <- lm (price_cents ~ `area squared foot` + Unit + `Lift access` + CCTV, data = Training)

summary(lmodel.Train)
```

```
##
## Call:
## lm(formula = price_cents ~ `area squared foot` + Unit + `Lift access` +
##     CCTV, data = Training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -581.08  -74.27  -20.80   58.26 1337.59
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1163.617    116.902   9.954  < 2e-16 ***
## `area squared foot`   55.419      2.214  25.029  < 2e-16 ***
## Unit                 233.773     44.983   5.197 5.25e-07 ***
## `Lift access`       -670.286    106.657  -6.285 2.24e-09 ***
## CCTV                -235.437     93.576  -2.516   0.0127 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 189.3 on 188 degrees of freedom
## Multiple R-squared:  0.8398, Adjusted R-squared:  0.8364
## F-statistic: 246.3 on 4 and 188 DF,  p-value: < 2.2e-16
```

```
lmodel.Test <- lm (price_cents ~ `area squared foot` + Unit + `Lift access` + CCTV, data = Test)

summary(lmodel.Test)
```

```
##
## Call:
## lm(formula = price_cents ~ `area squared foot` + Unit + `Lift access` +
##     CCTV, data = Test)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -747.0 -149.4   30.5   69.4 4429.6
##
## Coefficients: (1 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -160.11     319.12  -0.502 0.617306
## `area squared foot`    82.02      16.04   5.112 2.31e-06 ***
## Unit                  643.57     159.82   4.027 0.000133 ***
## `Lift access`             NA         NA      NA       NA
## CCTV                  266.82     299.15   0.892 0.375248
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 554.9 on 76 degrees of freedom
## Multiple R-squared:  0.3849, Adjusted R-squared:  0.3606
## F-statistic: 15.85 on 3 and 76 DF,  p-value: 4.24e-08
```