Long Homework 2

Bhavika Sewpal - 300089940

4/10/2021

Question 1 - Empirical Study: Cross-Validation Method 1

```
setwd("C:/Users/mahim/Desktop/winter2021/MAT3373/long_hw2")
mnist <- read.csv("mnist.csv")
label <- as.factor(mnist$label)

normalize <- function(x){
  if (max(x) - min(x) == 0){
    return (x)
  }else{
    return ((x-min(x))/(max(x) - min(x)))
  }
}

#normalizing the features
mnist <- as.data.frame(lapply(mnist[,2:ncol(mnist)],normalize))
mnist <- cbind(label,mnist)


set.seed(100)
index1 <- sample(c(1:10000),3333)
fold1 <- mnist[index1,]

mnist2 <- mnist[-index1,]
set.seed(100)
index2 <- sample(c(1:6667),3333)
fold2 <- mnist2[index2,]

fold3 <- mnist2[-index2,]



#Train on fold1 and fold2, test on fold3
train1 <- rbind(fold1,fold2)
train1_features <- train1[,-1]
train1_target <- train1[,1]

test1 <- fold3
test1_features <- test1[,-1]
test1_target <- test1[,1]
```

```r
library(class)
knn1 <- knn(train1_features, test1_features, train1_target, k=3)
table(test1_target,knn1)
```

```
##             knn1
## test1_target   0   1   2   3   4   5   6   7   8   9
##            0 316   1   0   0   0   1   4   0   0   0
##            1   0 372   1   1   0   0   1   0   0   0
##            2   5   3 338   1   1   0   4   5   2   0
##            3   0   1   0 309   0   4   0   8   0   3
##            4   0   2   1   0 305   0   1   0   0  14
##            5   2   3   0  13   0 262   2   0   1   4
##            6   2   1   0   0   0   1 319   0   1   0
##            7   0  13   2   0   0   0   0 325   0   3
##            8   2   2   4   9   1   5   2   2 296   3
##            9   4   3   1   2   9   0   0   7   2 322
```

```r
error_rate1 <- mean(knn1 != test1_target)
error_rate1
```

```
## [1] 0.0509898
```

```r
#Train on fold1 and fold3, test on fold2
train2 <- rbind(fold1,fold3)
train2_features <- train2[,-1]
train2_target <- train2[,1]

test2 <- fold2
test2_features <- test2[,-1]
test2_target <- test2[,1]

knn2 <- knn(train2_features, test2_features, train2_target, k=3)
table(test2_target,knn2)
```

```
##             knn2
## test2_target   0   1   2   3   4   5   6   7   8   9
##            0 323   0   0   0   0   2   1   0   0   0
##            1   0 373   2   0   0   0   0   1   0   0
##            2   6   9 330   3   0   0   1   7   1   0
##            3   0   1   1 320   0   8   0   1   4   2
##            4   0   3   0   0 307   1   1   0   1  11
##            5   0   5   0   8   1 267   2   1   1   4
##            6   4   3   0   0   1   1 308   0   1   0
##            7   0   8   0   1   1   0   0 315   0   6
##            8   1   4   2  10   3   8   2   2 301   4
##            9   1   5   0   1   5   1   0   8   1 316
```

```r
error_rate2 <- mean(knn2 != test2_target)
error_rate2
```

```
## [1] 0.05190519
```

```r
#Train on fold2 and fold3, test on fold1
train3 <- rbind(fold2,fold3)
train3_features <- train3[,-1]
train3_target <- train3[,1]

test3 <- fold1
test3_features <- test3[,-1]
test3_target <- test3[,1]

knn3 <- knn(train3_features, test3_features, train3_target, k=3)
table(test3_target,knn3)
```

```
##             knn3
## test3_target   0   1   2   3   4   5   6   7   8   9
##            0 328   0   0   0   0   0   2   1   1   0
##            1   0 382   1   0   0   0   1   0   0   0
##            2   2   6 291   1   1   0   1  12   1   1
##            3   0   0   2 331   0   6   1   3   5   0
##            4   0   7   0   0 313   0   4   0   0  11
##            5   2   0   0   7   0 301   4   1   1   0
##            6   1   1   0   0   0   0 314   0   0   0
##            7   1  11   2   0   2   0   1 330   0   7
##            8   1   5   4   7   2   4   2   3 280   3
##            9   0   0   2   4   5   1   0   8   2 299
```

```r
error_rate3 <- mean(knn3 != test3_target)
error_rate3
```

```
## [1] 0.04920492
```

```r
average <- (error_rate1 + error_rate2 + error_rate3)/ 3
average
```

```
## [1] 0.05069997
```

The error rate is 0.0507.
For hw1, the error rate was about 0.18 .
The error rate when using cross-validation is much smaller than that without cross-validation.
In general, I would have expected the contrary. Since, we are fitting the model on only 2/3 of the data, I would expect the error rate obtained empirically to be an overestimate of the test error.

Question 2 - Empirical Study: Cross-Validation Method 2

```r
#function to calculate the residual standard error - sigma
rse <- function(true,pred){
  sse <- sum((pred-true)^2)
  mean_sse <- sse / (length(pred))
}

redwine <- read.csv("redwine.csv",sep= ";")
set.seed(100)
```

```r
test_index <- sample(c(1:1599),799)
test <- redwine[test_index,]
train <- redwine[-test_index,]
 #Linear Regression
linear_model <- lm(quality~.,data=train)
summary(linear_model)
```

```
##
## Call:
## lm(formula = quality ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.41873 -0.38160 -0.04562  0.44648  2.10450
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -6.1018296 30.3018271  -0.201 0.840463
## fixed.acidity         0.0055089  0.0368867   0.149 0.881317
## volatile.acidity     -1.1722749  0.1792308  -6.541 1.10e-10 ***
## citric.acid          -0.1221569  0.2261068  -0.540 0.589169
## residual.sugar       -0.0028991  0.0218773  -0.133 0.894610
## chlorides            -2.3202111  0.6351444  -3.653 0.000276 ***
## free.sulfur.dioxide   0.0043875  0.0030721   1.428 0.153644
## total.sulfur.dioxide -0.0033095  0.0009774  -3.386 0.000745 ***
## density              11.0501078 30.9243710   0.357 0.720943
## pH                   -0.6007354  0.2737706  -2.194 0.028505 *
## sulphates             0.9385347  0.1696135   5.533 4.28e-08 ***
## alcohol               0.2882921  0.0380634   7.574 1.01e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6599 on 788 degrees of freedom
## Multiple R-squared:  0.3667, Adjusted R-squared:  0.3578
## F-statistic: 41.48 on 11 and 788 DF,  p-value: < 2.2e-16
```

```r
pred2 <- predict(linear_model,test)
testerr <- rse(test$quality,pred2 )
```

MSE for linear regression = 0.4091203.

```r
#Ridge Regression
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.4
```

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

```
## Loaded glmnet 4.1-1
```

```r
x <- model.matrix (quality~.,redwine )[,-1]
y <- redwine$quality
x_train <- x[-test_index,]
y_train <- y[-test_index]
x_test <- x[test_index,]
y_test <- y[test_index]

#sequence of lambdas to be tested
grid =10^ seq (10,-2, length =100)

#find the optimal lambda using cross validation
ridge_mod =glmnet (x_train,y_train,alpha =0, lambda =grid ,
thresh =1e-12)

set.seed (100)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid)
optimal_lambda <- cv_ridge$lambda.min

#make predictions on the test set
ridge_pred <- predict(ridge_mod, s = optimal_lambda, newx = x_test)
testerr2 <- rse(y_test,ridge_pred)

out=glmnet (x_train,y_train,alpha =0)
predict (out ,type="coefficients",s=optimal_lambda )
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           20.914828645
## fixed.acidity          0.026536059
## volatile.acidity      -1.047451229
## citric.acid            0.031839418
## residual.sugar         0.007182338
## chlorides             -2.173741029
## free.sulfur.dioxide    0.003248019
## total.sulfur.dioxide  -0.002995460
## density              -16.643662685
## pH                    -0.369405178
## sulphates              0.910337638
## alcohol                0.242468832
```

Optimal value of lambda for ridge regression = 0.070548.
The MSE is 0.4083508.

```r
#Lasso Regression

#find the optimal lambda using cross validation
lasso_mod =glmnet (x_train,y_train,alpha =1, lambda =grid)

set.seed (100)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid)
optimal_lambda <- cv_lasso$lambda.min
optimal_lambda
```

```
## [1] 0.01747528
```

```r
#make predictions on the test set
lasso_pred <- predict(lasso_mod, s = optimal_lambda, newx = x_test)
testerr3 <- rse(y_test,lasso_pred)
testerr3
```

```
## [1] 0.4084539
```

```r
out=glmnet (x_train,y_train,alpha =1)
predict (out ,type="coefficients",s=optimal_lambda )
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           4.251626306
## fixed.acidity         0.009283378
## volatile.acidity     -1.122037373
## citric.acid           .
## residual.sugar        .
## chlorides            -1.757534763
## free.sulfur.dioxide   .
## total.sulfur.dioxide -0.002047706
## density               .
## pH                   -0.354407573
## sulphates             0.831489738
## alcohol               0.267056943
```

Optimal value of lambda for lasso regression = 0.0174753.
The MSE is 0.4084539.

```r
#Select best linear model using forward stepwise regression
library (leaps)
```

```
## Warning: package 'leaps' was built under R version 4.0.4
```

```r
set.seed(100)
test_index <- sample(c(1:1599),799)
test <- redwine[test_index,]
train <- redwine[-test_index,]
regfit_fwd=regsubsets(quality~.,data=train,method="forward",nvmax=12)
summary(regfit_fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ ., data = train, method = "forward",
##     nvmax = 12)
## 11 Variables  (and intercept)
##                      Forced in Forced out
## fixed.acidity            FALSE      FALSE
## volatile.acidity         FALSE      FALSE
## citric.acid              FALSE      FALSE
```

```
## residual.sugar          FALSE      FALSE
## chlorides               FALSE      FALSE
## free.sulfur.dioxide     FALSE      FALSE
## total.sulfur.dioxide    FALSE      FALSE
## density                 FALSE      FALSE
## pH                      FALSE      FALSE
## sulphates               FALSE      FALSE
## alcohol                 FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: forward
##           fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1  ( 1 )  " "           " "              " "         " "            " "
## 2  ( 1 )  " "           "*"              " "         " "            " "
## 3  ( 1 )  " "           "*"              " "         " "            " "
## 4  ( 1 )  " "           "*"              " "         " "            " "
## 5  ( 1 )  " "           "*"              " "         " "            "*"
## 6  ( 1 )  " "           "*"              " "         " "            "*"
## 7  ( 1 )  " "           "*"              " "         " "            "*"
## 8  ( 1 )  " "           "*"              " "         " "            "*"
## 9  ( 1 )  " "           "*"              "*"         " "            "*"
## 10  ( 1 ) "*"           "*"              "*"         " "            "*"
## 11  ( 1 ) "*"           "*"              "*"         "*"            "*"
##           free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates
## 1  ( 1 )  " "                 " "                  " "     " " " "
## 2  ( 1 )  " "                 " "                  " "     " " " "
## 3  ( 1 )  " "                 " "                  " "     " " "*"
## 4  ( 1 )  " "                 "*"                  " "     " " "*"
## 5  ( 1 )  " "                 "*"                  " "     " " "*"
## 6  ( 1 )  " "                 "*"                  " "     "*" "*"
## 7  ( 1 )  "*"                 "*"                  " "     "*" "*"
## 8  ( 1 )  "*"                 "*"                  "*"     "*" "*"
## 9  ( 1 )  "*"                 "*"                  "*"     "*" "*"
## 10  ( 1 ) "*"                 "*"                  "*"     "*" "*"
## 11  ( 1 ) "*"                 "*"                  "*"     "*" "*"
##           alcohol
## 1  ( 1 )  "*"
## 2  ( 1 )  "*"
## 3  ( 1 )  "*"
## 4  ( 1 )  "*"
## 5  ( 1 )  "*"
## 6  ( 1 )  "*"
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
## 11  ( 1 ) "*"
```

```
test_mat=model.matrix(quality~.,data=test)
val_errors =rep(NA,11)

for(i in 1:11){
  # Find the coefficients selected in the models of different sizes
  coefi=coef(regfit_fwd ,id=i)
  #predict the test values
```

```r
  pred=test_mat[,names(coefi)]%*%coefi
  #calculate the errors
  val_errors[i]= mean(( test$quality-pred)^2)
}
# number of variables selected
best_model = which.min(val_errors)
best_model
```

```
## [1] 7
```

```r
testerr_fwd = val_errors[best_model]
# 7 variables selected over the test set
coef(regfit_fwd,best_model)
```

```
##          (Intercept)     volatile.acidity            chlorides
##          5.031454579         -1.114244843         -2.400007239
##  free.sulfur.dioxide total.sulfur.dioxide                   pH
##          0.004544875         -0.003472825         -0.605115158
##            sulphates              alcohol
##          0.956816066          0.275876634
```

The MSE is 0.4085524.

```r
#Select best linear model using backward stepwise regression
set.seed(100)
test_index <- sample(c(1:1599),799)
test <- redwine[test_index,]
train <- redwine[-test_index,]
regfit_bwd=regsubsets(quality~.,data=train,method="backward",nvmax=12)
summary(regfit_bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ ., data = train, method = "backward",
##     nvmax = 12)
## 11 Variables  (and intercept)
##                      Forced in Forced out
## fixed.acidity            FALSE      FALSE
## volatile.acidity         FALSE      FALSE
## citric.acid              FALSE      FALSE
## residual.sugar           FALSE      FALSE
## chlorides                FALSE      FALSE
## free.sulfur.dioxide      FALSE      FALSE
## total.sulfur.dioxide     FALSE      FALSE
## density                  FALSE      FALSE
## pH                       FALSE      FALSE
## sulphates                FALSE      FALSE
## alcohol                  FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: backward
##           fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
```

```
## 1  ( 1 )  " "           " "              " "         " "            " "
## 2  ( 1 )  " "           "*"              " "         " "            " "
## 3  ( 1 )  " "           "*"              " "         " "            " "
## 4  ( 1 )  " "           "*"              " "         " "            " "
## 5  ( 1 )  " "           "*"              " "         " "            "*"
## 6  ( 1 )  " "           "*"              " "         " "            "*"
## 7  ( 1 )  " "           "*"              " "         " "            "*"
## 8  ( 1 )  " "           "*"              " "         " "            "*"
## 9  ( 1 )  " "           "*"              "*"         " "            "*"
## 10  ( 1 ) "*"           "*"              "*"         " "            "*"
## 11  ( 1 ) "*"           "*"              "*"         "*"            "*"
##           free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates
## 1  ( 1 )  " "                 " "                  " "     " " " "
## 2  ( 1 )  " "                 " "                  " "     " " " "
## 3  ( 1 )  " "                 " "                  " "     " " "*"
## 4  ( 1 )  " "                 "*"                  " "     " " "*"
## 5  ( 1 )  " "                 "*"                  " "     " " "*"
## 6  ( 1 )  " "                 "*"                  " "     "*" "*"
## 7  ( 1 )  "*"                 "*"                  " "     "*" "*"
## 8  ( 1 )  "*"                 "*"                  "*"     "*" "*"
## 9  ( 1 )  "*"                 "*"                  "*"     "*" "*"
## 10  ( 1 ) "*"                 "*"                  "*"     "*" "*"
## 11  ( 1 ) "*"                 "*"                  "*"     "*" "*"
##           alcohol
## 1  ( 1 )  "*"
## 2  ( 1 )  "*"
## 3  ( 1 )  "*"
## 4  ( 1 )  "*"
## 5  ( 1 )  "*"
## 6  ( 1 )  "*"
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
## 11  ( 1 ) "*"
```

```r
test_mat=model.matrix(quality~.,data=test)
val_errors =rep(NA,11)

for(i in 1:11){
  # Find the coefficients selected in the models of different sizes
  coefi=coef(regfit_bwd ,id=i)
  #predict the test values
  pred=test_mat[,names(coefi)]%*%coefi
  #calculate the errors
  val_errors[i]= mean((test$quality-pred)^2)
}

best_model = which.min(val_errors)
best_model
```

```
## [1] 7
```

```r
testerr_bwd = val_errors[best_model]
# 7 variables selected over the test set

coef(regfit_bwd,best_model)
```

```
##          (Intercept)     volatile.acidity            chlorides
##          5.031454579         -1.114244843         -2.400007239
##  free.sulfur.dioxide total.sulfur.dioxide                   pH
##          0.004544875         -0.003472825         -0.605115158
##            sulphates              alcohol
##          0.956816066          0.275876634
```

The MSE is 0.4085524.

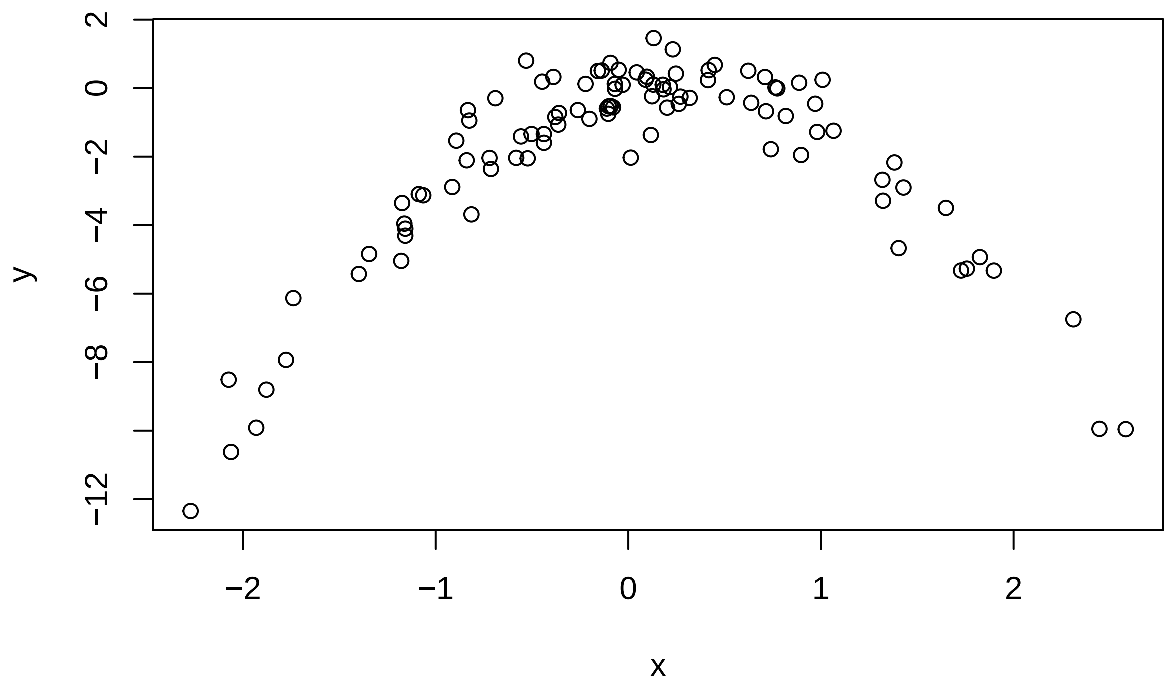
The test error for all the models are about 0.41.
It appears that the probability that we will correctly predict wine quality is about 0.6.
The same 7 predictors are chosen for lasso,forward and backward regression: volatile acidity, chlorides, free sulfur dioxide, total sulfur dioxide, pH, suplphates and alcohol.
With linear regression, we find 6 variables to be significant : the same as above except free sulfur dioxide.
With ridge regression, all variables are used.
All things considered, we can say that all models (except the ridge model) are fairly similar.

Question 3 - Simulation Study: Cross Validation Method

```r
set.seed(100)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
```

n = 100.
$y = (x-2)x^2 +   y = x^3 -2x^2 + $ where $\epsilon \sim N(0,1)$ p =2

```r
plot(x,y)
```



There appears to be a non linear (quadratic relationship) between x and y.

```r
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.0.4
```

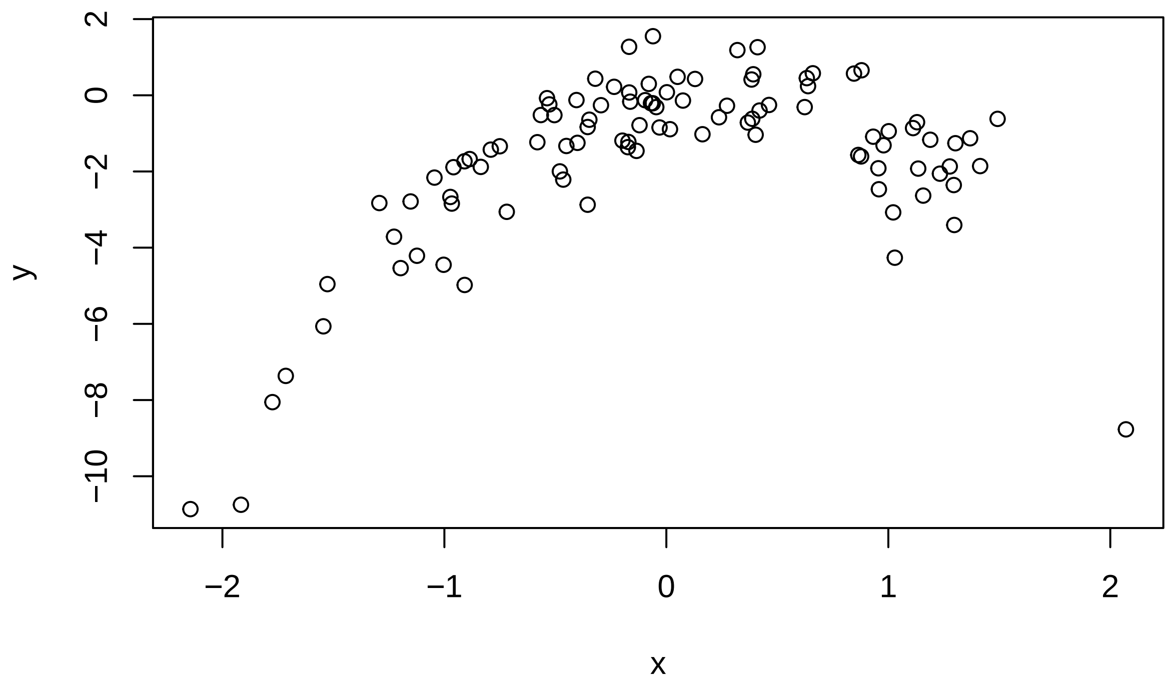
```r
set.seed(100)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
simulated <- data.frame(x,y)

cv_error=rep (0,5)
for (i in 1:5){
  glm_fit=glm(y~poly(x ,i),data=simulated)
  cv_error[i]=cv.glm(simulated,glm_fit)$delta [1]
}
cv_error
```

```
## [1] 9.0606362 0.6511909 0.6665339 0.6671261 0.6744096
```

```r
index <- which.min(cv_error)
min_error1 <- cv_error[index]
```

The quadratic model has the smallest LOOCV error : 0.6511909.

```r
set.seed(205)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
plot(x,y)
```




```r
simulated <- data.frame(x,y)
cv_error=rep (0,5)
for (i in 1:5){
  glm_fit=glm(y~poly(x ,i),data=simulated)
  cv_error[i]=cv.glm(simulated,glm_fit)$delta [1]
}
cv_error
```

```
## [1] 4.755583 1.037305 1.175235 1.427486 3.159778
```

```r
index <- which.min(cv_error)
min_error2 <- cv_error[index]
```

Again, the quadratic model has the smallest LOOCV error : 1.0373049.
This is because the scatterplot in both cases is best approximated by a quadratic curve. (even if the true function is cubic).
However even if the quadratic model has the smallest error in both cases, the LOOCV error for the second experiment is larger.
I expected the models to have different errors because each time, the values of x and y will change depending

on the seed we set.

```
set.seed(100)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
simulated <- data.frame(x,y)

fit1=glm(y~poly(x ,1),data=simulated)
summary(fit1)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 1), data = simulated)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -9.313  -1.212   1.125   1.968   3.439
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0488     0.2908  -7.045 2.59e-10 ***
## poly(x, 1)    5.5351     2.9080   1.903   0.0599 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.456659)
##
##     Null deviance: 859.39  on 99  degrees of freedom
## Residual deviance: 828.75  on 98  degrees of freedom
## AIC: 501.26
##
## Number of Fisher Scoring iterations: 2
```

```
fit2=glm(y~poly(x ,2),data=simulated)
summary(fit2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2), data = simulated)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0511  -0.4242  -0.1232   0.5291   1.8763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.04883    0.07897 -25.945  < 2e-16 ***
## poly(x, 2)1   5.53505    0.78969   7.009  3.2e-10 ***
## poly(x, 2)2 -27.71753    0.78969 -35.099  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for gaussian family taken to be 0.6236173)
##
##     Null deviance: 859.389  on 99  degrees of freedom
## Residual deviance:  60.491  on 97  degrees of freedom
## AIC: 241.52
##
## Number of Fisher Scoring iterations: 2
```

```r
fit3=glm(y~poly(x ,3),data=simulated)
summary(fit3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3), data = simulated)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0027  -0.4533  -0.1187   0.5101   1.8385
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.04883    0.07924 -25.856  < 2e-16 ***
## poly(x, 3)1   5.53505    0.79238   6.985 3.72e-10 ***
## poly(x, 3)2 -27.71753    0.79238 -34.980  < 2e-16 ***
## poly(x, 3)3   0.46381    0.79238   0.585     0.56
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.6278725)
##
##     Null deviance: 859.389  on 99  degrees of freedom
## Residual deviance:  60.276  on 96  degrees of freedom
## AIC: 243.16
##
## Number of Fisher Scoring iterations: 2
```

```r
fit4=glm(y~poly(x ,4),data=simulated)
summary(fit4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), data = simulated)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0603  -0.4947  -0.1133   0.5593   1.8436
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.04883    0.07883 -25.991  < 2e-16 ***
## poly(x, 4)1   5.53505    0.78829   7.022 3.26e-10 ***
## poly(x, 4)2 -27.71753    0.78829 -35.162  < 2e-16 ***
## poly(x, 4)3   0.46381    0.78829   0.588    0.558
```

```
## poly(x, 4)4   1.11467    0.78829   1.414    0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.6214028)
##
##     Null deviance: 859.389  on 99  degrees of freedom
## Residual deviance:  59.033  on 95  degrees of freedom
## AIC: 243.08
##
## Number of Fisher Scoring iterations: 2
```

As illustrated by the summaries, the AIC value for the quadratic model is the smallest (as suggested by the cross-validation method).
For the linear model, the coefficien for β_1 is not significant.
For the quadratic model, both β_1 and β_2 are significant.
For the cubic model, both β_1 and β_2 are significant but β_3 is not.
For the quartic model, β_1 and β_2 are significant but β_3 and β_4 are not.
These observations agree with the results of the LOOCV, which selected the quadratic model.

Question 4 - Simulation Study: Screening, Stepwise Selection and ROC Curves

```r
gwas <- read.csv("GWAS.CSV")
gwas <- gwas[,-1]
#Convert all the columns to factors except V1
set.seed(100)
train_index <- sample( c(1:3000), 1500)
train <- gwas[train_index,]
test <- gwas[-train_index,]
logistic <- glm(V1~.,data=train,family="binomial")
#Significant attributes
significant <- summary(logistic)$coef[,4][summary(logistic)$coef[,4] < 0.05]
prob <- predict(logistic,test,type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
pred <- ifelse(prob >= 0.5, 1, 0)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Loading required package: ggplot2
```

```r
confusionMatrix(as.factor(pred),as.factor(test$V1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 646 336
##          1 324 194
##
##                Accuracy : 0.56
##                  95% CI : (0.5344, 0.5853)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0322
##
##  Mcnemar's Test P-Value : 0.6685
##
##             Sensitivity : 0.6660
##             Specificity : 0.3660
##          Pos Pred Value : 0.6578
##          Neg Pred Value : 0.3745
##              Prevalence : 0.6467
##          Detection Rate : 0.4307
##    Detection Prevalence : 0.6547
##       Balanced Accuracy : 0.5160
##
##        'Positive' Class : 0
##
```

/ On test data set, the model predicts correctly 56% of the time./ It is not a good model./

```r
#Ridge Regression
library(glmnet)
x <- model.matrix (V1~.,train )[,-1]
y <- train$V1
x_test<- model.matrix (V1~.,test )[,-1]
y_test <- test$V1


#sequence of lambdas to be tested
grid =10^ seq (10,-2, length =100)

#find the optimal lambda using cross validation


set.seed (100)
cv_ridge <- cv.glmnet(x, y, alpha = 0, lambda = grid,family="binomial")
optimal_lambda <- cv_ridge$lambda.min

#make predictions on the test set
ridge_prob<- predict(cv_ridge, s = optimal_lambda, newx = x_test,type="response")
```

```r
ridge_pred <- ifelse(ridge_prob >= 0.5, 1, 0)
confusionMatrix(as.factor(y), as.factor(ridge_pred))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 964   0
##          1 534   2
##
##                Accuracy : 0.644
##                  95% CI : (0.6192, 0.6683)
##     No Information Rate : 0.9987
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0048
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.643525
##             Specificity : 1.000000
##          Pos Pred Value : 1.000000
##          Neg Pred Value : 0.003731
##              Prevalence : 0.998667
##          Detection Rate : 0.642667
##    Detection Prevalence : 0.642667
##       Balanced Accuracy : 0.821762
##
##        'Positive' Class : 0
##
```

The ridge logistic regression model is 64.4% accurate.

```r
#Lasso Regression

#find the optimal lambda using cross validation
set.seed (100)
cv_lasso <- cv.glmnet(x, y, alpha = 1, lambda = grid,family="binomial")
optimal_lambda <- cv_lasso$lambda.min

#make predictions on the test set
lasso_prob<- predict(cv_lasso, s = optimal_lambda, newx = x_test,type="response")
lasso_pred <- ifelse(lasso_prob >= 0.5, 1, 0)
table(lasso_pred,y)
```

```
##           y
## lasso_pred   0   1
##          0 964 536
```

```r
mean(lasso_pred==y)
```

```
## [1] 0.6426667
```

The lasso logistic regression model is 64.27 % accurate but it classifies all the test data as 0.
Summary:
Model Accuracy
Logistic 56%
Ridge Logistic 64.4%
Lasso Logistic 64.3%

It appears that the penalised logistic regression models (Lasso and Ridge) work better than the traditional logistic regression model.

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.4
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

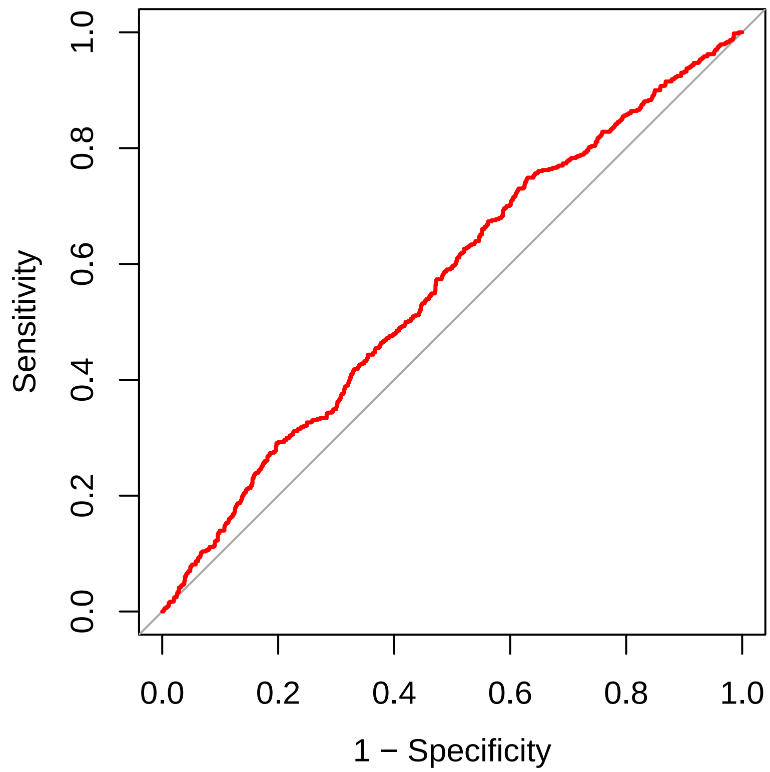
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
par(pty ="s")
#plot ROC curve for ridge logistic regression
roc_info_ridge <-  roc(response=test$V1, predictor=ridge_prob , plot= TRUE, legacy.axes=TRUE,col="red")
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response = test$V1, predictor = ridge_prob, plot
## = TRUE, : Deprecated use a matrix as predictor. Unexpected results may be
## produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```

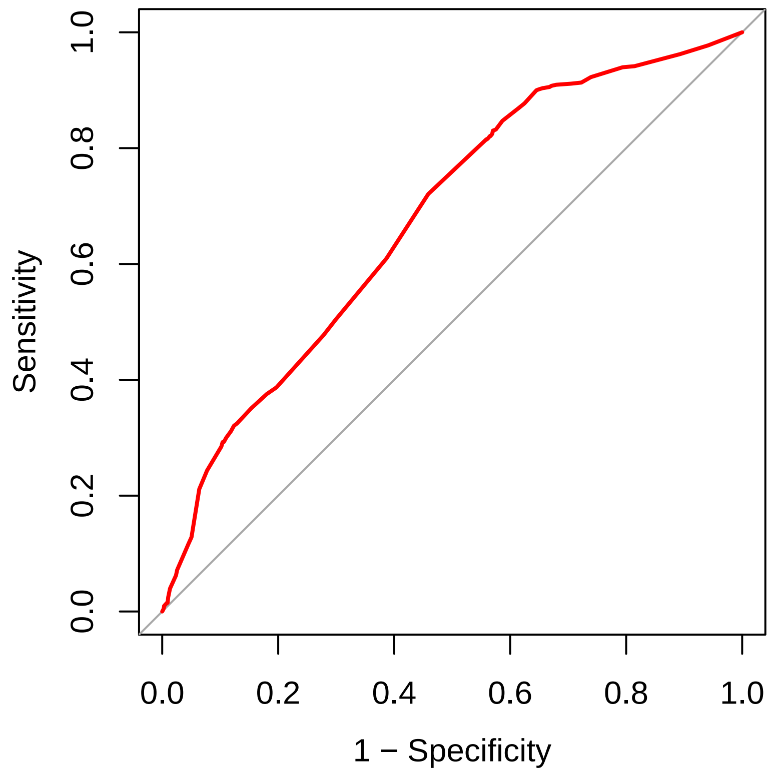



```
#plot ROC curve for lasso logistic regression
roc_info_lasso <-  roc(response=test$V1, predictor=lasso_prob , plot= TRUE, legacy.axes=TRUE,col="red")
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response = test$V1, predictor = lasso_prob, plot
## = TRUE, : Deprecated use a matrix as predictor. Unexpected results may be
## produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```




```
roc.df <- data.frame(tpp=roc_info_lasso$sensitivities*100,
fpp=(1-roc_info_lasso$specificities)*100,thresholds=roc_info_lasso$thresholds)


threshold <- seq(0,1,by=0.001)
accuracy <- rep(0,length(threshold))
for(i in 1:length(threshold)){
  lasso_new_pred <- ifelse(lasso_prob >= threshold[i], 1, 0)
  accuracy[i] <- mean(lasso_new_pred==y)

}
index <- which.max(accuracy)
accuracy[index]
```

```
## [1] 0.6426667
```

```
threshold[index]
```

```
## [1] 0.498
```

It seems that for a threshold above or equal to 0.498, the highest accuracy = 0.6426667 remains constant. This is because the lasso model at a threshold of 0.498 classfies all observations as 0. Thus, increasing the threshold above 0.50 does not change the accuracy but decreasing the threshold leads to a decrease in accuracy.

```r
library(coefplot)
```

```
## Warning: package 'coefplot' was built under R version 4.0.4
```

```r
extract.coef(cv.glmnet(x, y, alpha = 1, lambda = grid,family="binomial"))
```

```
##                   Value SE Coefficient
## (Intercept) -0.45289612 NA (Intercept)
## V65          0.29290046 NA         V65
## V80          0.10746436 NA         V80
## V300        -0.66302796 NA        V300
## V316         0.04070697 NA        V316
## V337         0.02334652 NA        V337
## V408        -0.03933016 NA        V408
```

Decision procedure:
Collect data about alleles V65, V80, V300, V316, V337 and V408 for the population.
We choose these alleles because these are the only variables that have non zero coefficients in the lasso model.
We predict the chances of having the disease using the lasso model.
We want the number of false negatives to be as small as possible (as we don't want a patient having the disease to be classified as not having the disease)
One course of action would be to establish an accepted level of false negatives and try to reduce the number of false positives (i.e aim for a high sensibility), by choosing an appropriate threshold based on the ROC graph.
We then send these people for screening

Question 7 - Empirical Study: LASSO vs. Regression Workflow

Bhavika Sewpal - 300089940

4/10/2021

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


#Cleaning up for train dataset

#Removing columns neighbourhood_group_cleansed and bathroom (contain only NAs)
train <- train[,c(-13,-19)]

#Converting categorical variables (true/false) to 0 and 1.
train$host_is_superhost <- factor(train$host_is_superhost, levels = c("t","f"), labels=c(1,0))

train$host_has_profile_pic <- factor(train$host_has_profile_pic, levels = c("t","f"), labels=c(1,0))
train$host_identity_verified <- factor(train$host_identity_verified, levels = c("t","f"), labels=c(1,0))
train$has_availability<- factor(train$has_availability, levels = c("t","f"), labels=c(1,0))
train$instant_bookable <- factor(train$instant_bookable, levels = c("t","f"), labels=c(1,0))

#Remove 72 rows
index <- is.na(train$review_scores_checkin)
train <- train[(!index),]

#Remove dollar sign from price column
train$price = as.numeric(gsub("\\$", "", train$price))
```

```
## Warning: NAs introduced by coercion
```

```
#find out the NAs in price column and replace them with the average value of price
index1 <- which(is.na(train$price))
median_price <- 89
train[index1,21] = median_price


#Remove percentage sign in host_response_rate column
train$host_response_rate = as.numeric(gsub("\\%", "", train$host_response_rate))
```

```
## Warning: NAs introduced by coercion
```

```
#find out the NAs in the response rate column and replace them with the average value of response rate
index2 <- which(is.na(train$host_response_rate))
avg_response <- 92.5
train[index2,4] = avg_response
```

```r
#Remove percentage sign in host_acceptance_rate column
train$host_acceptance_rate = as.numeric(gsub("\\%", "", train$host_acceptance_rate))
```

```
## Warning: NAs introduced by coercion
```

```r
#find out the NAs in the acceptance rate column and replace them with the average value of acceptance r
index3 <- which(is.na(train$host_acceptance_rate))
avg_acceptance <- 76.87
train[index3,5] = avg_acceptance


#Replace missing values in host_response_time with more frequent category:within an hour
index4 <- which((train$host_response_time=="N/A"))
train[index4,3] <- "within an hour"

index5 <- which(is.na(train$bedrooms))
avg_bedrooms <- 1.7
train[index5,18] <- 1.7

#Converting to categorical variables to factors
train$room_type <- factor(train$room_type)
train$host_response_time <- factor(train$host_response_time)
train$host_neighbourhood<- factor(train$host_neighbourhood)
train$neighbourhood_cleansed<- factor(train$neighbourhood_cleansed)
train$property_type <- factor(train$property_type)

summary(train)
```

```
##  description        neighborhood_overview          host_response_time
##  Length:928         Length:928            a few days or more: 23
##  Class :character   Class :character      within a day      : 79
##  Mode  :character   Mode  :character      within a few hours:132
##                                           within an hour    :694
##
##
##
##  host_response_rate host_acceptance_rate host_is_superhost
##  Min.   :  0.00     Min.   :  0.00       1:361
##  1st Qu.: 92.50     1st Qu.: 76.40       0:567
##  Median : 99.00     Median : 80.00
##  Mean   : 92.78     Mean   : 77.62
##  3rd Qu.:100.00     3rd Qu.: 97.25
##  Max.   :100.00     Max.   :100.00
##
##                         host_neighbourhood host_listings_count
##                                  :827      Min.   :  0.000
##  Sandy Hill                      : 24      1st Qu.:  1.000
##  Byward Market - Parliament Hill: 11      Median :  2.000
##  Centretown                      : 11      Mean   :  5.413
##  Downtown                        : 10      3rd Qu.:  3.000
##  Lower Town                      :  6      Max.   :272.000
##  (Other)                         : 39
```

```
##  host_total_listings_count host_has_profile_pic host_identity_verified
##  Min.   :  0.000          1:927                1:781
##  1st Qu.:  1.000          0:  1                0:147
##  Median :  2.000
##  Mean   :  5.413
##  3rd Qu.:  3.000
##  Max.   :272.000
##
##        neighbourhood_cleansed    latitude       longitude
##  Rideau-Vanier    :176        Min.   :45.13   Min.   :-76.22
##  Somerset         :132        1st Qu.:45.37   1st Qu.:-75.73
##  Kitchissippi     : 95        Median :45.41   Median :-75.69
##  Capital          : 78        Mean   :45.39   Mean   :-75.70
##  River            : 47        3rd Qu.:45.43   3rd Qu.:-75.67
##  Rideau-Rockcliffe: 41        Max.   :45.51   Max.   :-75.35
##  (Other)          :359
##                property_type           room_type    accommodates
##  Entire apartment     :240   Entire home/apt:583   Min.   : 1.000
##  Private room in house:201   Hotel room     :  1   1st Qu.: 2.000
##  Entire house         :152   Private room   :338   Median : 3.000
##  Entire guest suite   : 49   Shared room    :  6   Mean   : 3.642
##  Entire condominium   : 46                         3rd Qu.: 5.000
##  Entire townhouse     : 45                         Max.   :16.000
##  (Other)              :195
##     bedrooms        beds         amenities             price
##  Min.   :1.0   Min.   : 0.000   Length:928         Min.   :   0.0
##  1st Qu.:1.0   1st Qu.: 1.000   Class :character   1st Qu.:  60.0
##  Median :1.0   Median : 1.000   Mode  :character   Median :  89.0
##  Mean   :1.7   Mean   : 1.928                      Mean   : 133.9
##  3rd Qu.:2.0   3rd Qu.: 3.000                      3rd Qu.: 125.0
##  Max.   :6.0   Max.   :12.000                      Max.   :9800.0
##
##  minimum_nights    maximum_nights      has_availability number_of_reviews
##  Min.   :   1.00   Min.   :1.000e+00   1:913            Min.   :  1.00
##  1st Qu.:   1.00   1st Qu.:5.000e+01   0: 15            1st Qu.: 12.00
##  Median :   2.00   Median :1.125e+03                    Median : 39.00
##  Mean   :  11.93   Mean   :2.315e+06                    Mean   : 73.13
##  3rd Qu.:   5.00   3rd Qu.:1.125e+03                    3rd Qu.: 98.25
##  Max.   :1000.00   Max.   :2.147e+09                    Max.   :553.00
##
##  number_of_reviews_ltm number_of_reviews_l30d first_review
##  Min.   :  0.000       Min.   :0.0000         Length:928
##  1st Qu.:  0.000       1st Qu.:0.0000         Class :character
##  Median :  1.000       Median :0.0000         Mode  :character
##  Mean   :  6.534       Mean   :0.2263
##  3rd Qu.:  7.000       3rd Qu.:0.0000
##  Max.   :112.000       Max.   :6.0000
##
##  last_review        review_scores_rating review_scores_accuracy
##  Length:928         Min.   : 20.00       Min.   : 2.000
##  Class :character   1st Qu.: 94.00       1st Qu.:10.000
##  Mode  :character   Median : 97.00       Median :10.000
##                     Mean   : 95.38       Mean   : 9.749
##                     3rd Qu.: 99.00       3rd Qu.:10.000
```

```
##                    Max.   :100.00       Max.   :10.000
##
##  review_scores_cleanliness review_scores_checkin review_scores_communication
##  Min.   : 2.00             Min.   : 2.000        Min.   : 2.000
##  1st Qu.: 9.00             1st Qu.:10.000        1st Qu.:10.000
##  Median :10.00             Median :10.000        Median :10.000
##  Mean   : 9.58             Mean   : 9.894        Mean   : 9.874
##  3rd Qu.:10.00             3rd Qu.:10.000        3rd Qu.:10.000
##  Max.   :10.00             Max.   :10.000        Max.   :10.000
##
##  review_scores_location review_scores_value instant_bookable
##  Min.   : 2.000         Min.   : 2.000      1:191
##  1st Qu.:10.000         1st Qu.: 9.000      0:737
##  Median :10.000         Median :10.000
##  Mean   : 9.706         Mean   : 9.564
##  3rd Qu.:10.000         3rd Qu.:10.000
##  Max.   :10.000         Max.   :10.000
##
##  calculated_host_listings_count calculated_host_listings_count_entire_homes
##  Min.   : 1.000                 Min.   : 0.000
##  1st Qu.: 1.000                 1st Qu.: 0.000
##  Median : 1.000                 Median : 1.000
##  Mean   : 3.454                 Mean   : 2.505
##  3rd Qu.: 3.000                 3rd Qu.: 2.000
##  Max.   :69.000                 Max.   :69.000
##
##  calculated_host_listings_count_private_rooms reviews_per_month
##  Min.   : 0.0000                              Min.   :0.010
##  1st Qu.: 0.0000                              1st Qu.:0.270
##  Median : 0.0000                              Median :0.800
##  Mean   : 0.9332                              Mean   :1.509
##  3rd Qu.: 1.0000                              3rd Qu.:2.060
##  Max.   :12.0000                              Max.   :9.810
##
```

```r
#Cleaning up optimization dataset
#Removing columns neighbourhood_group_cleansed and bathroom (contain only NAs)
opt <- read.csv("ListingsOptimization.csv")
opt <- opt[,c(-13,-19)]

#Converting categorical variables (true/false) to 0 and 1.
opt$host_is_superhost <- factor(opt$host_is_superhost, levels = c("t","f"), labels=c(1,0))

opt$host_has_profile_pic <- factor(opt$host_has_profile_pic, levels = c("t","f"), labels=c(1,0))
opt$host_identity_verified <- factor(opt$host_identity_verified, levels = c("t","f"), labels=c(1,0))
opt$has_availability<- factor(opt$has_availability, levels = c("t","f"), labels=c(1,0))
opt$instant_bookable <- factor(opt$instant_bookable, levels = c("t","f"), labels=c(1,0))

#Remove 72 rows
index <- is.na(opt$review_scores_checkin)
opt <- opt[(!index),]

#Remove dollar sign from price column
opt$price = as.numeric(gsub("\\$", "", opt$price))
```

```
## Warning: NAs introduced by coercion
```

```
#find out the NAs in price column and replace them with the average value of price
index1 <- which(is.na(opt$price))
median_price <- 80
opt[index1,21] <- median_price


#Remove percentage sign in host_response_rate column
opt$host_response_rate = as.numeric(gsub("\\%", "", opt$host_response_rate))
```

```
## Warning: NAs introduced by coercion
```

```
#find out the NAs in the response rate column and replace them with the average value of response rate
index2 <- which(is.na(opt$host_response_rate))
avg_response <- 92.76
opt[index2,4] <- avg_response

#Remove percentage sign in host_acceptance_rate column
opt$host_acceptance_rate = as.numeric(gsub("\\%", "", opt$host_acceptance_rate))
```

```
## Warning: NAs introduced by coercion
```

```
#find out the NAs in the acceptance rate column and replace them with the average value of acceptance r
index3 <- which(is.na(opt$host_acceptance_rate))
avg_acceptance <- 82.98
opt[index3,5] <- avg_acceptance


#Replace missing values in host_response_time with more frequent category:within an hour
index4 <- which((opt$host_response_time=="N/A"))
opt[index4,3] <-"within an hour"

#Converting to categorical variables to factors
opt$room_type <- factor(opt$room_type)
opt$host_response_time <- factor(opt$host_response_time)
opt$host_neighbourhood<- factor(opt$host_neighbourhood)
opt$neighbourhood_cleansed<- factor(opt$neighbourhood_cleansed)
opt$property_type <- factor(opt$property_type)

#dealing with NAs
index5 <- which(is.na(opt$host_is_superhost))
opt[index5,6] = 0

index6 <- which(is.na(opt$host_listings_count))
opt[index6,8] = 2

index7 <- which(is.na(opt$host_total_listings_count))
opt[index7,9] = 2

index8 <- which(is.na(opt$host_has_profile_pic))
opt[index8,10] = 1
```

```r
index9<- which(is.na(opt$host_identity_verified))
opt[index9,11] = 1

index10 <- which(is.na(opt$bedrooms))
opt[index10,18] = 1

summary(opt)
```

```
##  description        neighborhood_overview          host_response_time
##  Length:848         Length:848                                  :  1
##  Class :character   Class :character      a few days or more: 16
##  Mode  :character   Mode  :character      within a day      : 50
##                                           within a few hours: 84
##                                           within an hour    :697
##
##
##  host_response_rate host_acceptance_rate host_is_superhost
##  Min.   :  0.00     Min.   :  0.00       1:355
##  1st Qu.: 92.76     1st Qu.: 82.98       0:493
##  Median :100.00     Median : 91.00
##  Mean   : 94.12     Mean   : 84.91
##  3rd Qu.:100.00     3rd Qu.: 99.00
##  Max.   :100.00     Max.   :100.00
##
##                       host_neighbourhood host_listings_count
##                                :710      Min.   :  0.00
##  Byward Market - Parliament Hill: 22      1st Qu.:  1.00
##  Sandy Hill                     : 18      Median :  2.00
##  Centretown                     : 13      Mean   :  7.14
##  Downtown                       : 13      3rd Qu.:  4.00
##  Centretown West                :  8      Max.   :272.00
##  (Other)                        : 64
##  host_total_listings_count host_has_profile_pic host_identity_verified
##  Min.   :  0.00            1:841                1:667
##  1st Qu.:  1.00            0:  7                0:181
##  Median :  2.00
##  Mean   :  7.14
##  3rd Qu.:  4.00
##  Max.   :272.00
##
##    neighbourhood_cleansed    latitude       longitude
##  Rideau-Vanier:165        Min.   :44.99   Min.   :-76.11
##  Somerset     :111        1st Qu.:45.35   1st Qu.:-75.74
##  Kanata North : 53        Median :45.40   Median :-75.69
##  River        : 49        Mean   :45.38   Mean   :-75.71
##  Capital      : 48        3rd Qu.:45.43   3rd Qu.:-75.67
##  Kitchissippi : 42        Max.   :45.52   Max.   :-75.39
##  (Other)      :380
##                   property_type            room_type    accommodates
##  Entire apartment          :196   Entire home/apt:500   Min.   : 1.000
##  Private room in house     :163   Hotel room     :  5   1st Qu.: 2.000
##  Entire house              :123   Private room   :337   Median : 3.000
```

```
##  Private room in townhouse: 78   Shared room    :  6   Mean   : 3.586
##  Entire guest suite       : 55                         3rd Qu.: 5.000
##  Entire condominium       : 41                         Max.   :16.000
##  (Other)                  :192
##     bedrooms          beds          amenities             price
##  Min.   :1.000   Min.   : 0.000   Length:848         Min.   : 20.00
##  1st Qu.:1.000   1st Qu.: 1.000   Class :character   1st Qu.: 53.00
##  Median :1.000   Median : 1.000   Mode  :character   Median : 80.00
##  Mean   :1.605   Mean   : 1.909                      Mean   : 96.29
##  3rd Qu.:2.000   3rd Qu.: 2.000                      3rd Qu.:115.00
##  Max.   :8.000   Max.   :16.000                      Max.   :868.00
##
##  minimum_nights     maximum_nights   has_availability number_of_reviews
##  Min.   :   1.000   Min.   :   1.0   1:842            Min.   :  1.00
##  1st Qu.:   1.000   1st Qu.:  31.0   0:  6            1st Qu.:  5.00
##  Median :   2.000   Median :1125.0                    Median : 17.00
##  Mean   :   8.888   Mean   : 643.1                    Mean   : 32.23
##  3rd Qu.:   4.000   3rd Qu.:1125.0                    3rd Qu.: 44.25
##  Max.   :1000.000   Max.   :1125.0                    Max.   :279.00
##
##  number_of_reviews_ltm number_of_reviews_l30d first_review
##  Min.   :  0.0         Min.   :0.0000         Length:848
##  1st Qu.:  0.0         1st Qu.:0.0000         Class :character
##  Median :  3.0         Median :0.0000         Mode  :character
##  Mean   :  9.3         Mean   :0.3396
##  3rd Qu.: 11.0         3rd Qu.:0.0000
##  Max.   :118.0         Max.   :9.0000
##
##  last_review        review_scores_rating review_scores_accuracy
##  Length:848         Min.   : 20.00       Min.   : 2.000
##  Class :character   1st Qu.: 93.00       1st Qu.:10.000
##  Mode  :character   Median : 97.00       Median :10.000
##                     Mean   : 94.35       Mean   : 9.611
##                     3rd Qu.:100.00       3rd Qu.:10.000
##                     Max.   :100.00       Max.   :10.000
##
##  review_scores_cleanliness review_scores_checkin review_scores_communication
##  Min.   : 2.000            Min.   : 2.000        Min.   : 2.000
##  1st Qu.: 9.000            1st Qu.:10.000        1st Qu.:10.000
##  Median :10.000            Median :10.000        Median :10.000
##  Mean   : 9.495            Mean   : 9.789        Mean   : 9.739
##  3rd Qu.:10.000            3rd Qu.:10.000        3rd Qu.:10.000
##  Max.   :10.000            Max.   :10.000        Max.   :10.000
##
##  review_scores_location review_scores_value instant_bookable
##  Min.   : 2.000         Min.   : 2.000      1:303
##  1st Qu.:10.000         1st Qu.: 9.000      0:545
##  Median :10.000         Median :10.000
##  Mean   : 9.678         Mean   : 9.441
##  3rd Qu.:10.000         3rd Qu.:10.000
##  Max.   :10.000         Max.   :10.000
##
##  calculated_host_listings_count calculated_host_listings_count_entire_homes
##  Min.   : 1.000                 Min.   : 0.000
```

```
##  1st Qu.: 1.000               1st Qu.: 0.000
##  Median : 2.000               Median : 1.000
##  Mean   : 4.421               Mean   : 3.179
##  3rd Qu.: 4.000               3rd Qu.: 2.000
##  Max.   :69.000               Max.   :69.000
##
##  calculated_host_listings_count_private_rooms reviews_per_month
##  Min.   : 0.000                               Min.   :0.0300
##  1st Qu.: 0.000                               1st Qu.:0.2575
##  Median : 0.000                               Median :0.8000
##  Mean   : 1.202                               Mean   :1.4326
##  3rd Qu.: 2.000                               3rd Qu.:2.0400
##  Max.   :12.000                               Max.   :9.8800
##
```

```r
#Cleaning up for test dataset

#Removing columns neighbourhood_group_cleansed and bathroom (contain only NAs)
test <- read.csv("ListingsTest.csv")
test <- test[,c(-13,-19)]

#Converting categorical variables (true/false) to 0 and 1.
test$host_is_superhost <- factor(test$host_is_superhost, levels = c("t","f"), labels=c(1,0))

test$host_has_profile_pic <- factor(test$host_has_profile_pic, levels = c("t","f"), labels=c(1,0))
test$host_identity_verified <- factor(test$host_identity_verified, levels = c("t","f"), labels=c(1,0))
test$has_availability<- factor(test$has_availability, levels = c("t","f"), labels=c(1,0))
test$instant_bookable <- factor(test$instant_bookable, levels = c("t","f"), labels=c(1,0))

#Remove 400 rows
index <- is.na(test$review_scores_checkin)
test <- test[(!index),]

#Remove dollar sign from price column
test$price = as.numeric(gsub("\\$", "", test$price))
#find out the NAs in price column and replace them with the average value of price
index1 <- which(is.na(test$price))
median_price <- 80
test[index1,21] <- median_price


#Remove percentage sign in host_response_rate column
test$host_response_rate = as.numeric(gsub("\\%", "", test$host_response_rate))
```

```
## Warning: NAs introduced by coercion
```

```r
#find out the NAs in the response rate column and replace them with the average value of response rate
index2 <- which(is.na(test$host_response_rate))
avg_response <- 94.34
test[index2,4] <- avg_response

#Remove percentage sign in host_acceptance_rate column
test$host_acceptance_rate = as.numeric(gsub("\\%", "", test$host_acceptance_rate))
```

```
## Warning: NAs introduced by coercion
```

```r
#find out the NAs in the acceptance rate column and replace them with the average value of acceptance r
index3 <- which(is.na(test$host_acceptance_rate))
avg_acceptance <- 88.18
test[index3,5] <- avg_acceptance


#Replace missing values in host_response_time with more frequent category:within an hour
index4 <- which((test$host_response_time=="N/A"))
test[index4,3] <- "within an hour"

#Converting to categorical variables to factors
test$room_type <- factor(test$room_type)
test$host_response_time <- factor(test$host_response_time)
test$host_neighbourhood<- factor(test$host_neighbourhood)
test$neighbourhood_cleansed<- factor(test$neighbourhood_cleansed)
test$property_type <- factor(test$property_type)

index5 <- which(is.na(test$bedrooms))
test[index5,18] = 1

index6 <- which(is.na(test$beds))
test[index6,19] = 1
summary(test)
```

```
##  description        neighborhood_overview          host_response_time
##  Length:404         Length:404            a few days or more:  5
##  Class :character   Class :character      within a day      : 22
##  Mode  :character   Mode  :character      within a few hours: 48
##                                           within an hour    :329
##
##
##
##  host_response_rate host_acceptance_rate host_is_superhost
##  Min.   :  0.00     Min.   :  0.00       1:113
##  1st Qu.: 99.00     1st Qu.: 89.00       0:291
##  Median :100.00     Median : 95.00
##  Mean   : 96.41     Mean   : 90.57
##  3rd Qu.:100.00     3rd Qu.:100.00
##  Max.   :100.00     Max.   :100.00
##
##                        host_neighbourhood host_listings_count
##                                 :153      Min.   :  0.000
##  Downtown                       : 41      1st Qu.:  0.000
##  Byward Market - Parliament Hill: 28      Median :  1.000
##  Centretown West                : 27      Mean   :  9.077
##  Centretown                     : 20      3rd Qu.:  4.000
##  Sandy Hill                     : 17      Max.   :272.000
##  (Other)                        :118
##  host_total_listings_count host_has_profile_pic host_identity_verified
##  Min.   :  0.000           1:404                1:334
##  1st Qu.:  0.000           0:  0                0: 70
##  Median :  1.000
```

```
## Mean   :  9.077
## 3rd Qu.:  4.000
## Max.   :272.000
##
##       neighbourhood_cleansed    latitude       longitude
## Rideau-Vanier    :111        Min.   :45.05   Min.   :-76.00
## Somerset         : 58        1st Qu.:45.36   1st Qu.:-75.73
## Kitchissippi     : 26        Median :45.41   Median :-75.69
## Rideau-Rockcliffe: 24        Mean   :45.39   Mean   :-75.70
## Capital          : 22        3rd Qu.:45.43   3rd Qu.:-75.67
## College          : 16        Max.   :45.49   Max.   :-75.46
## (Other)          :147
##                     property_type           room_type    accommodates
## Entire apartment         :159   Entire home/apt:270   Min.   : 1.000
## Entire house             : 44   Private room   :132   1st Qu.: 2.000
## Private room in house    : 41   Shared room    :  2   Median : 2.000
## Private room in townhouse: 38                         Mean   : 3.391
## Entire condominium       : 25                         3rd Qu.: 4.000
## Entire guest suite       : 20                         Max.   :16.000
## (Other)                  : 77
##    bedrooms          beds       amenities            price
## Min.   :1.000   Min.   :0.00   Length:404         Min.   : 18.00
## 1st Qu.:1.000   1st Qu.:1.00   Class :character   1st Qu.: 52.75
## Median :1.000   Median :1.00   Mode  :character   Median : 78.50
## Mean   :1.562   Mean   :1.79                      Mean   : 89.74
## 3rd Qu.:2.000   3rd Qu.:2.00                      3rd Qu.:106.75
## Max.   :9.000   Max.   :9.00                      Max.   :700.00
##
## minimum_nights    maximum_nights   has_availability number_of_reviews
## Min.   :   1.00   Min.   :   1.0   1:403            Min.   : 1.000
## 1st Qu.:   1.00   1st Qu.:  60.0   0:  1            1st Qu.: 2.000
## Median :   2.00   Median : 682.0                    Median : 4.000
## Mean   :   6.53   Mean   : 643.9                    Mean   : 8.342
## 3rd Qu.:   4.00   3rd Qu.:1125.0                    3rd Qu.:11.000
## Max.   :1000.00   Max.   :1125.0                    Max.   :81.000
##
## number_of_reviews_ltm number_of_reviews_l30d first_review
## Min.   : 1.000        Min.   :0.0000        Length:404
## 1st Qu.: 2.000        1st Qu.:0.0000        Class :character
## Median : 4.000        Median :0.0000        Mode  :character
## Mean   : 8.104        Mean   :0.8639
## 3rd Qu.:10.250        3rd Qu.:1.0000
## Max.   :69.000        Max.   :8.0000
##
## last_review          review_scores_rating review_scores_accuracy
## Length:404          Min.   : 20.00       Min.   : 2.0
## Class :character    1st Qu.: 92.00       1st Qu.: 9.0
## Mode  :character    Median : 98.00       Median :10.0
##                     Mean   : 93.07       Mean   : 9.5
##                     3rd Qu.:100.00       3rd Qu.:10.0
##                     Max.   :100.00       Max.   :10.0
##
## review_scores_cleanliness review_scores_checkin review_scores_communication
## Min.   : 2.000            Min.   : 2.000        Min.   : 2.000
```

```
##  1st Qu.: 9.000            1st Qu.:10.000       1st Qu.:10.000
##  Median :10.000            Median :10.000       Median :10.000
##  Mean   : 9.413            Mean   : 9.723       Mean   : 9.678
##  3rd Qu.:10.000            3rd Qu.:10.000       3rd Qu.:10.000
##  Max.   :10.000            Max.   :10.000       Max.   :10.000
##
##  review_scores_location review_scores_value instant_bookable
##  Min.   : 2.000         Min.   : 2.000      1:162
##  1st Qu.:10.000         1st Qu.: 9.000      0:242
##  Median :10.000         Median :10.000
##  Mean   : 9.696         Mean   : 9.403
##  3rd Qu.:10.000         3rd Qu.:10.000
##  Max.   :10.000         Max.   :10.000
##
##  calculated_host_listings_count calculated_host_listings_count_entire_homes
##  Min.   : 1.00                  Min.   : 0.000
##  1st Qu.: 1.00                  1st Qu.: 0.000
##  Median : 2.00                  Median : 1.000
##  Mean   : 9.53                  Mean   : 7.891
##  3rd Qu.: 9.00                  3rd Qu.: 7.000
##  Max.   :69.00                  Max.   :69.000
##
##  calculated_host_listings_count_private_rooms reviews_per_month
##  Min.   : 0.000                               Min.   :0.080
##  1st Qu.: 0.000                               1st Qu.:0.530
##  Median : 0.000                               Median :1.000
##  Mean   : 1.624                               Mean   :1.648
##  3rd Qu.: 2.000                               3rd Qu.:2.260
##  Max.   :11.000                               Max.   :8.700
##
```

The variables neighbourhood_group_cleansed and bathrooms are useless. They contain NAs only.

```r
# Regression with suspicious points
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.4
```

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

```
## Loaded glmnet 4.1-1
```

```r
#select the best lambda with cross validation
grid =10^ seq (10,-2, length =100)

mse <- function(true,pred){
  sse <- mean((pred-true)^2)
}

selected_opt <- opt[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_opt<- model.matrix(price~.,selected_opt)[,-1]
y_opt <- selected_opt$price
```

```r
selected_train <- train[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_train<- model.matrix(price~.,selected_train)[,-1]
y_train <- selected_train$price

selected_test <- test[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_test<- model.matrix(price~.,selected_test)[,-1]
y_test <- selected_test$price




set.seed (100)
cv_lasso <- cv.glmnet(x_opt, y_opt, alpha = 1, lambda = grid)
optimal_lambda <- cv_lasso$lambda.min
optimal_lambda
```

```
## [1] 0.6579332
```

```r
 #train the model
lasso_mod =glmnet(x_train,y_train,alpha =1, lambda =optimal_lambda)
summary(lasso_mod)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      55     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

```r
#make predictions on the test set
lasso_pred <- predict(lasso_mod, s = optimal_lambda, newx = x_test)
lasso_coef <- predict(lasso_mod,s=optimal_lambda,type="coefficients")
lasso_coef
```

```
## 56 x 1 sparse Matrix of class "dgCMatrix"
##                                                         1
## (Intercept)                                  8.792078e+03
## host_response_timewithin a day              -8.692954e+01
## host_response_timewithin a few hours         9.054449e+00
## host_response_timewithin an hour            -6.837487e+01
## host_response_rate                           7.209731e-01
## host_acceptance_rate                         2.474053e-01
## host_is_superhost0                          -6.440685e+01
## host_listings_count                         -2.795725e-01
```

```
## host_total_listings_count                          -6.297941e-13
## host_has_profile_pic0                              -1.089051e+02
## host_identity_verified0                            -1.569526e+01
## neighbourhood_cleansedBarrhaven                    -1.754701e+01
## neighbourhood_cleansedBay                           .
## neighbourhood_cleansedBeacon Hill-Cyrville         -4.706946e+01
## neighbourhood_cleansedCapital                      -2.647551e+00
## neighbourhood_cleansedCollege                      -2.695608e+01
## neighbourhood_cleansedCumberland                   -1.717832e+01
## neighbourhood_cleansedGloucester-South Nepean -3.819953e+01
## neighbourhood_cleansedGloucester-Southgate         -2.588383e+01
## neighbourhood_cleansedInnes                        -2.825754e+01
## neighbourhood_cleansedKanata North                  .
## neighbourhood_cleansedKanata South                 -1.056111e+01
## neighbourhood_cleansedKitchissippi                  1.524314e+02
## neighbourhood_cleansedKnoxdale-Merivale            -4.074176e+01
## neighbourhood_cleansedOrleans                      -5.384571e+01
## neighbourhood_cleansedOsgoode                      -2.110030e+01
## neighbourhood_cleansedRideau-Goulbourn              2.409357e+01
## neighbourhood_cleansedRideau-Rockcliffe            -1.565087e+01
## neighbourhood_cleansedRideau-Vanier                 .
## neighbourhood_cleansedRiver                        -1.582774e+01
## neighbourhood_cleansedSomerset                      9.435319e+01
## neighbourhood_cleansedStittsville-Kanata West  .
## neighbourhood_cleansedWest Carleton-March           2.952699e+01
## latitude                                            .
## longitude                                           1.166092e+02
## accommodates                                        5.861266e+00
## bedrooms                                            3.588107e+01
## beds                                               -9.975633e+00
## minimum_nights                                      1.986073e-01
## maximum_nights                                     -7.260631e-08
## has_availability0                                   5.249541e+01
## number_of_reviews                                   .
## number_of_reviews_ltm                               4.004092e+00
## number_of_reviews_l30d                              1.780602e+00
## review_scores_rating                                .
## review_scores_accuracy                              .
## review_scores_cleanliness                           1.965311e+01
## review_scores_checkin                               1.909110e+00
## review_scores_communication                         2.567770e+01
## review_scores_location                             -1.041705e+01
## review_scores_value                                -2.476359e+01
## instant_bookable0                                  -1.215086e+01
## calculated_host_listings_count                      .
## calculated_host_listings_count_entire_homes         .
## calculated_host_listings_count_private_rooms   7.304412e-01
## reviews_per_month                                  -2.842106e+01
```

```
testerr <- mse(y_test,lasso_pred)
testerr
```

```
## [1] 7629.632
```

```r
#normal regression on dataset
mod <- lm(price~.,data=selected_train)
summary(mod)
```

```
##
## Call:
## lm(formula = price ~ ., data = selected_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -374.7  -84.5  -20.9   34.9 9487.0
##
## Coefficients: (1 not defined because of singularities)
##                                                 Estimate Std. Error t value
## (Intercept)                                    2.267e+04  5.856e+04   0.387
## host_response_timewithin a day                -1.694e+02  2.257e+02  -0.750
## host_response_timewithin a few hours          -7.231e+01  2.397e+02  -0.302
## host_response_timewithin an hour              -1.525e+02  2.326e+02  -0.655
## host_response_rate                             1.485e+00  2.258e+00   0.658
## host_acceptance_rate                           2.950e-01  7.010e-01   0.421
## host_is_superhost0                            -6.698e+01  3.775e+01  -1.775
## host_listings_count                           -3.702e-01  1.394e+00  -0.266
## host_total_listings_count                             NA         NA      NA
## host_has_profile_pic0                         -1.377e+02  4.828e+02  -0.285
## host_identity_verified0                       -1.633e+01  4.621e+01  -0.353
## neighbourhood_cleansedBarrhaven               -3.454e+01  1.991e+02  -0.173
## neighbourhood_cleansedBay                     -4.819e+00  1.593e+02  -0.030
## neighbourhood_cleansedBeacon Hill-Cyrville    -9.707e+01  1.378e+02  -0.704
## neighbourhood_cleansedCapital                 -3.252e+01  1.047e+02  -0.311
## neighbourhood_cleansedCollege                 -4.181e+01  1.562e+02  -0.268
## neighbourhood_cleansedCumberland              -9.952e+01  1.888e+02  -0.527
## neighbourhood_cleansedGloucester-South Nepean -5.978e+01  1.481e+02  -0.404
## neighbourhood_cleansedGloucester-Southgate    -6.779e+01  1.361e+02  -0.498
## neighbourhood_cleansedInnes                   -9.613e+01  1.588e+02  -0.605
## neighbourhood_cleansedKanata North             1.541e+01  2.400e+02   0.064
## neighbourhood_cleansedKanata South            -5.951e+00  2.216e+02  -0.027
## neighbourhood_cleansedKitchissippi             1.362e+02  1.137e+02   1.198
## neighbourhood_cleansedKnoxdale-Merivale       -6.796e+01  1.471e+02  -0.462
## neighbourhood_cleansedOrleans                 -1.219e+02  1.776e+02  -0.686
## neighbourhood_cleansedOsgoode                 -6.827e+01  2.489e+02  -0.274
## neighbourhood_cleansedRideau-Goulbourn         2.166e+01  2.270e+02   0.095
## neighbourhood_cleansedRideau-Rockcliffe       -5.670e+01  1.206e+02  -0.470
## neighbourhood_cleansedRideau-Vanier           -2.723e+01  1.009e+02  -0.270
## neighbourhood_cleansedRiver                   -3.960e+01  1.212e+02  -0.327
## neighbourhood_cleansedSomerset                 7.493e+01  1.040e+02   0.720
## neighbourhood_cleansedStittsville-Kanata West  3.041e+01  2.516e+02   0.121
## neighbourhood_cleansedWest Carleton-March      8.231e+01  3.025e+02   0.272
## latitude                                       4.324e+01  8.258e+02   0.052
## longitude                                      3.254e+02  6.211e+02   0.524
## accommodates                                   7.076e+00  1.473e+01   0.480
## bedrooms                                       3.540e+01  3.160e+01   1.120
## beds                                          -1.144e+01  2.208e+01  -0.518
## minimum_nights                                 2.013e-01  3.579e-01   0.562
```

```
## maximum_nights                                -8.600e-08  2.278e-07  -0.377
## has_availability0                              7.863e+01  1.360e+02   0.578
## number_of_reviews                              4.012e-02  5.829e-01   0.069
## number_of_reviews_ltm                          4.311e+00  2.292e+00   1.881
## number_of_reviews_l30d                         1.718e+00  2.902e+01   0.059
## review_scores_rating                          -4.180e-01  5.755e+00  -0.073
## review_scores_accuracy                        -1.496e-02  3.852e+01   0.000
## review_scores_cleanliness                      2.376e+01  3.550e+01   0.669
## review_scores_checkin                          1.542e+00  5.243e+01   0.029
## review_scores_communication                    2.811e+01  5.707e+01   0.493
## review_scores_location                        -1.091e+01  3.091e+01  -0.353
## review_scores_value                           -2.596e+01  3.504e+01  -0.741
## instant_bookable0                             -1.869e+01  4.196e+01  -0.445
## calculated_host_listings_count                -8.669e+01  1.204e+02  -0.720
## calculated_host_listings_count_entire_homes    8.674e+01  1.203e+02   0.721
## calculated_host_listings_count_private_rooms   8.994e+01  1.215e+02   0.740
## reviews_per_month                             -3.359e+01  3.401e+01  -0.988
##                                               Pr(>|t|)
## (Intercept)                                     0.6987
## host_response_timewithin a day                  0.4532
## host_response_timewithin a few hours            0.7630
## host_response_timewithin an hour                0.5123
## host_response_rate                              0.5109
## host_acceptance_rate                            0.6740
## host_is_superhost0                              0.0763 .
## host_listings_count                             0.7906
## host_total_listings_count                           NA
## host_has_profile_pic0                           0.7756
## host_identity_verified0                         0.7239
## neighbourhood_cleansedBarrhaven                 0.8623
## neighbourhood_cleansedBay                       0.9759
## neighbourhood_cleansedBeacon Hill-Cyrville      0.4814
## neighbourhood_cleansedCapital                   0.7562
## neighbourhood_cleansedCollege                   0.7890
## neighbourhood_cleansedCumberland                0.5983
## neighbourhood_cleansedGloucester-South Nepean   0.6867
## neighbourhood_cleansedGloucester-Southgate      0.6186
## neighbourhood_cleansedInnes                     0.5451
## neighbourhood_cleansedKanata North              0.9488
## neighbourhood_cleansedKanata South              0.9786
## neighbourhood_cleansedKitchissippi              0.2312
## neighbourhood_cleansedKnoxdale-Merivale         0.6443
## neighbourhood_cleansedOrleans                   0.4926
## neighbourhood_cleansedOsgoode                   0.7839
## neighbourhood_cleansedRideau-Goulbourn          0.9240
## neighbourhood_cleansedRideau-Rockcliffe         0.6383
## neighbourhood_cleansedRideau-Vanier             0.7874
## neighbourhood_cleansedRiver                     0.7438
## neighbourhood_cleansedSomerset                  0.4716
## neighbourhood_cleansedStittsville-Kanata West   0.9038
## neighbourhood_cleansedWest Carleton-March       0.7856
## latitude                                        0.9583
## longitude                                       0.6004
## accommodates                                    0.6310
```

```
## bedrooms                                     0.2630
## beds                                         0.6045
## minimum_nights                               0.5739
## maximum_nights                               0.7059
## has_availability0                            0.5634
## number_of_reviews                            0.9451
## number_of_reviews_ltm                        0.0604 .
## number_of_reviews_l30d                       0.9528
## review_scores_rating                         0.9421
## review_scores_accuracy                       0.9997
## review_scores_cleanliness                    0.5034
## review_scores_checkin                        0.9765
## review_scores_communication                  0.6224
## review_scores_location                       0.7243
## review_scores_value                          0.4590
## instant_bookable0                            0.6562
## calculated_host_listings_count               0.4716
## calculated_host_listings_count_entire_homes  0.4711
## calculated_host_listings_count_private_rooms 0.4595
## reviews_per_month                            0.3236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 477.6 on 873 degrees of freedom
## Multiple R-squared:  0.04063,    Adjusted R-squared:  -0.01871
## F-statistic: 0.6847 on 54 and 873 DF,  p-value: 0.9597
```

```r
reg_pred <- predict(mod,selected_test)
```

```
## Warning in predict.lm(mod, selected_test): prediction from a rank-deficient fit
## may be misleading
```

```r
testerr2 <- mse(selected_test$price,reg_pred)
testerr2
```

```
## [1] 8782.083
```

```r
#Regression without suspicious points
#Removing suspicious values
train_new <- train

i1 <- which(train_new$host_listings_count  > 200)
train_new <- train_new[-i1,]

i2 <- which(train_new$price > 9000)
train_new <- train_new[-i2,]

i3 <- which(train_new$beds > 8)
train_new <- train_new[-i3,]

i4 <- which(train_new$number_of_reviews > 500)
train_new <- train_new[-i4,]
```

```r
i5 <- which(train_new$maximum_nights> 3000)
train_new <- train_new[-i5,]

i6 <- which(train_new$number_of_reviews_ltm > 100)
train_new <- train_new[-i6,]

i7 <- which(train_new$number_of_reviews_l30d > 4)
train_new <- train_new[-i7,]

i8 <- which(train_new$room_type=="Hotel room")
train_new <- train_new[-i8,]
train_new$room_type <- factor(train_new$room_type)

#normal regression

selected_train2 <- train_new[,c(-1,-2,-7,-15,-20,-28,-29)]
selected_test2 <- test[,c(-1,-2,-7,-15,-20,-28,-29)]
mod2 <- lm(price~.,data=selected_train2)
reg_pred2 <- predict(mod2,selected_test2)
```

```
## Warning in predict.lm(mod2, selected_test2): prediction from a rank-deficient
## fit may be misleading
```

```r
testerr3 <- mse(selected_test2$price,reg_pred2)
testerr3
```

```
## [1] 9412.505
```

```r
#lasso regression
x_train2<- model.matrix(price~.,selected_train2)[,-1]
y_train2 <- selected_train2$price

x_test2<- model.matrix(price~.,selected_test2)[,-1]
y_test2 <- selected_test2$price

#make predictions on the test set
lasso_mod2 =glmnet(x_train2,y_train2,alpha =1, lambda =optimal_lambda)
lasso_pred2 <- predict(lasso_mod2, s = optimal_lambda, newx = x_test2)
lasso_coef2 <- predict(lasso_mod2,s=optimal_lambda,type="coefficients")
lasso_coef2
```

```
## 58 x 1 sparse Matrix of class "dgCMatrix"
##                                                 1
## (Intercept)                          362.59935968
## host_response_timewithin a day       -24.28236218
## host_response_timewithin a few hours  69.69136419
## host_response_timewithin an hour     -22.36362625
## host_response_rate                     0.39479992
## host_acceptance_rate                   0.16473544
## host_is_superhost0                   -46.10799535
## host_listings_count                    .
```

```
## host_total_listings_count                            .
## host_has_profile_pic0                                .
## host_identity_verified0                            -6.78262309
## neighbourhood_cleansedBarrhaven                    -5.81670075
## neighbourhood_cleansedBay                          -8.88591034
## neighbourhood_cleansedBeacon Hill-Cyrville        -42.06722246
## neighbourhood_cleansedCapital                        .
## neighbourhood_cleansedCollege                     -34.53889260
## neighbourhood_cleansedCumberland                  -10.36678978
## neighbourhood_cleansedGloucester-South Nepean      -3.18289886
## neighbourhood_cleansedGloucester-Southgate         -8.73129313
## neighbourhood_cleansedInnes                       -32.52841798
## neighbourhood_cleansedKanata North                -14.23714023
## neighbourhood_cleansedKanata South                 -6.25143930
## neighbourhood_cleansedKitchissippi                127.52626268
## neighbourhood_cleansedKnoxdale-Merivale           -63.82631203
## neighbourhood_cleansedOrleans                     -15.71073765
## neighbourhood_cleansedOsgoode                        .
## neighbourhood_cleansedRideau-Goulbourn             13.05757041
## neighbourhood_cleansedRideau-Rockcliffe              .
## neighbourhood_cleansedRideau-Vanier               -15.60361123
## neighbourhood_cleansedRiver                       -13.62803857
## neighbourhood_cleansedSomerset                      8.16855131
## neighbourhood_cleansedStittsville-Kanata West        .
## neighbourhood_cleansedWest Carleton-March          29.69534078
## latitude                                           62.31218362
## longitude                                          42.05976715
## room_typePrivate room                             -92.98318120
## room_typeShared room                             -130.96938140
## accommodates                                        4.25517822
## bedrooms                                           26.80813998
## beds                                               -4.42445324
## minimum_nights                                      0.11177620
## maximum_nights                                      0.02352460
## has_availability0                                  47.62759260
## number_of_reviews                                  -0.06133796
## number_of_reviews_ltm                              -0.05625324
## number_of_reviews_l30d                               .
## review_scores_rating                               -0.75816715
## review_scores_accuracy                              7.67340012
## review_scores_cleanliness                          17.03538570
## review_scores_checkin                               6.89244771
## review_scores_communication                        19.66233872
## review_scores_location                             -6.09738229
## review_scores_value                               -27.19507325
## instant_bookable0                                 -33.99588358
## calculated_host_listings_count                       .
## calculated_host_listings_count_entire_homes        -0.70819760
## calculated_host_listings_count_private_rooms       15.86610128
## reviews_per_month                                 -17.43142192
```

```
testerr4 <- mse(y_test2,lasso_pred2)
testerr4
```

```
## [1] 8216.946
```

```
testerr;testerr2;testerr3;testerr4
```

```
## [1] 7629.632

## [1] 8782.083

## [1] 9412.505

## [1] 8216.946
```

Without removing susicious points:
The test error for standard regression is 8782.0830349.
The test error for Lasso regression is 7629.6322779.

Removed suspicious points:
The test error for standad regression is 9412.504995.
The test error for Lasso regression is 8216.9460501.

In both cases, the test error for Lasso Regression is smaller than the test error for standard regression. But the test errors for the models containing the suspicious points are smaller than the corresponding test errors for the models in which the suspicious points have been removed.

It appears that lasso regression for the model containing the suspicious points is the best model.

It appears that removing the data points caused the sample size to decrease and thus might have inadvertently led to an increase in the test error rate.

```
text <- rep("",928)
for(i in 1:928){
  text[i] <- paste(train$description[i],train$neighborhood_overview[i],train$amenities[i],sep="")
}
n = length(text)
n
```

```
## [1] 928
```

```
w1 <- grepl("bedroom", text, ignore.case = TRUE)
table(w1)/n
```

```
## w1
##     FALSE      TRUE
## 0.3415948 0.6584052
```

```
w2 <- grepl("Wifi", text, ignore.case = TRUE)
table(w2)/n
```

```
## w2
##      FALSE       TRUE
## 0.01831897 0.98168103
```

```r
w3<- grepl("heating", text, ignore.case = TRUE)
table(w3)/n
```

```
## w3
##     FALSE      TRUE
## 0.0237069 0.9762931
```

```r
w4 <- grepl("walk", text, ignore.case = TRUE)
table(w4)/n
```

```
## w4
##     FALSE      TRUE
## 0.3340517 0.6659483
```

```r
w5 <- grepl("university", text, ignore.case = TRUE)
table(w5)/n
```

```
## w5
##     FALSE      TRUE
## 0.8706897 0.1293103
```

```r
w6 <- grepl("Refrigerator", text, ignore.case = TRUE)
table(w6)/n
```

```
## w6
##     FALSE      TRUE
## 0.3038793 0.6961207
```

```r
w7 <- grepl("downtown", text, ignore.case = TRUE)
table(w7)/n
```

```
## w7
##     FALSE      TRUE
## 0.4353448 0.5646552
```

```r
w8 <- grepl("parking", text, ignore.case = TRUE)
table(w8)/n
```

```
## w8
##  FALSE   TRUE
## 0.0625 0.9375
```

```r
w9 <- grepl("water", text, ignore.case = TRUE)
table(w9)/n
```

```
## w9
##     FALSE      TRUE
## 0.1831897 0.8168103
```

```r
w10 <- grepl("alarm", text, ignore.case = TRUE)
table(w10)/n
```

```
## w10
##     FALSE      TRUE
## 0.0237069 0.9762931
```

```r
w11 <- grepl("bus",text,ignore.case = TRUE)
table(w11)/n
```

```
## w11
##     FALSE      TRUE
## 0.5344828 0.4655172
```

```r
w12 <- grepl("quiet",text,ignore.case = TRUE)
table(w11)/n
```

```
## w11
##     FALSE      TRUE
## 0.5344828 0.4655172
```

```r
bedroom <- ifelse(w1,1,0)
wifi <- ifelse(w2,1,0)
heating <- ifelse(w3,1,0)
walk <- ifelse(w4,1,0)
university <- ifelse(w5,1,0)
refrigerator <- ifelse(w6,1,0)
downtown <- ifelse(w7,1,0)
parking <- ifelse(w8,1,0)
water <- ifelse(w9,1,0)
alarm <- ifelse(w10,1,0)
bus <- ifelse(w11,1,0)
quiet <- ifelse(w12,1,0)

train$bedroom <- bedroom
train$wifi <- wifi
train$heating <- heating
train$walk <- walk
train$university <- university
train$refrigerator <- refrigerator
train$downtown <- downtown
train$parking <- parking
train$water <- water
train$alarm <- alarm
train$bus <- bus
train$quiet <- quiet
```

We choose the above 12 words.
Each p(w) is obtained in the tables (in the column TRUE).

```r
#modify the test dataset
texttest <- rep("",404)
for(i in 1:404){
  texttest[i] <- paste(test$description[i],test$neighborhood_overview[i],test$amenities[i],sep="")
}

w11 <- grepl("bedroom", texttest, ignore.case = TRUE)
w22 <- grepl("Wifi", texttest, ignore.case = TRUE)
w33<- grepl("heating", texttest, ignore.case = TRUE)
w44 <- grepl("walk", texttest, ignore.case = TRUE)
w55 <- grepl("university", texttest, ignore.case = TRUE)
w66 <- grepl("Refrigerator", texttest, ignore.case = TRUE)
w77 <- grepl("downtown", texttest, ignore.case = TRUE)
w88 <- grepl("parking", texttest, ignore.case = TRUE)
w99 <- grepl("water", texttest, ignore.case = TRUE)
w100 <- grepl("alarm", texttest, ignore.case = TRUE)
w111 <- grepl("bus",texttest,ignore.case = TRUE)
w122 <- grepl("quiet",texttest,ignore.case = TRUE)

bedroom <- ifelse(w11,1,0)
wifi <- ifelse(w22,1,0)
heating <- ifelse(w33,1,0)
walk <- ifelse(w44,1,0)
university <- ifelse(w55,1,0)
refrigerator <- ifelse(w66,1,0)
downtown <- ifelse(w77,1,0)
parking <- ifelse(w88,1,0)
water <- ifelse(w99,1,0)
alarm <- ifelse(w100,1,0)
bus <- ifelse(w111,1,0)
quiet <- ifelse(w122,1,0)

test$bedroom <- bedroom
test$wifi <- wifi
test$heating <- heating
test$walk <- walk
test$university <- university
test$refrigerator <- refrigerator
test$downtown <- downtown
test$parking <- parking
test$water <- water
test$alarm <- alarm
test$bus <- bus
test$quiet <- quiet
#modify the optimization dataset

textopt <- rep("",848)
for(i in 1:848){
  textopt[i] <- paste(opt$description[i],opt$neighborhood_overview[i],opt$amenities[i],sep="")
}

o1 <- grepl("bedroom", textopt, ignore.case = TRUE)
o2 <- grepl("Wifi", textopt, ignore.case = TRUE)
```

```
o3<- grepl("heating", textopt, ignore.case = TRUE)
o4 <- grepl("walk", textopt, ignore.case = TRUE)
o5 <- grepl("university", textopt, ignore.case = TRUE)
o6 <- grepl("Refrigerator", textopt, ignore.case = TRUE)
o7 <- grepl("downtown", textopt, ignore.case = TRUE)
o8 <- grepl("parking", textopt, ignore.case = TRUE)
o9 <- grepl("water", textopt, ignore.case = TRUE)
o10 <- grepl("alarm", textopt, ignore.case = TRUE)
o11 <- grepl("bus",textopt,ignore.case = TRUE)
o12 <- grepl("quiet",textopt,ignore.case = TRUE)

bedroom <- ifelse(o1,1,0)
wifi <- ifelse(o2,1,0)
heating <- ifelse(o3,1,0)
walk <- ifelse(o4,1,0)
university <- ifelse(o5,1,0)
refrigerator <- ifelse(o6,1,0)
downtown <- ifelse(o7,1,0)
parking <- ifelse(o8,1,0)
water <- ifelse(o9,1,0)
alarm <- ifelse(o10,1,0)
bus <- ifelse(o11,1,0)
quiet <- ifelse(o12,1,0)

opt$bedroom <- bedroom
opt$wifi <- wifi
opt$heating <- heating
opt$walk <- walk
opt$university <- university
opt$refrigerator <- refrigerator
opt$downtown <- downtown
opt$parking <- parking
opt$water <- water
opt$alarm <- alarm
opt$bus <- bus
opt$quiet <- quiet

#standard regression model
selected_train <- train[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_train<- model.matrix(price~.,selected_train)[,-1]
y_train <- selected_train$price

selected_test <- test[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_test<- model.matrix(price~.,selected_test)[,-1]
y_test <- selected_test$price

selected_opt <- opt[, c(-1,-2,-7,-15,-16,-20,-28,-29)]
x_opt<- model.matrix(price~.,selected_opt)[,-1]
y_opt <- selected_opt$price


#Find the new lambda
set.seed (100)
```

```r
las <- cv.glmnet(x_opt, y_opt, alpha = 1, lambda = grid)
optimal <- las$lambda.min
optimal
```

```
## [1] 0.869749
```

```r
 #train the model
lassomod =glmnet(x_train,y_train,alpha =1, lambda =optimal)
summary(lassomod)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      67     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

```r
#make predictions on the test set
lasso_pred <- predict(lassomod, s = optimal, newx = x_test)
lasso_coef <- predict(lassomod,s=optimal,type="coefficients")
lasso_coef
```

```
## 68 x 1 sparse Matrix of class "dgCMatrix"
##                                                        1
## (Intercept)                                 8.122593e+03
## host_response_timewithin a day             -5.312847e+01
## host_response_timewithin a few hours        4.266451e+01
## host_response_timewithin an hour           -3.336827e+01
## host_response_rate                          4.430360e-01
## host_acceptance_rate                        2.700334e-01
## host_is_superhost0                         -6.671329e+01
## host_listings_count                        -2.892500e-01
## host_total_listings_count                  -4.284730e-13
## host_has_profile_pic0                      -1.110841e+02
## host_identity_verified0                    -1.380409e+01
## neighbourhood_cleansedBarrhaven            -7.329558e+00
## neighbourhood_cleansedBay                    .
## neighbourhood_cleansedBeacon Hill-Cyrville -4.234317e+01
## neighbourhood_cleansedCapital              -3.307401e-01
## neighbourhood_cleansedCollege              -3.077579e+01
## neighbourhood_cleansedCumberland           -9.701915e+00
## neighbourhood_cleansedGloucester-South Nepean -3.457931e+01
## neighbourhood_cleansedGloucester-Southgate -2.363459e+01
## neighbourhood_cleansedInnes                -2.588075e+01
## neighbourhood_cleansedKanata North           .
```

```
## neighbourhood_cleansedKanata South            -1.547461e+01
## neighbourhood_cleansedKitchissippi             1.483358e+02
## neighbourhood_cleansedKnoxdale-Merivale       -3.285513e+01
## neighbourhood_cleansedOrleans                 -5.783579e+01
## neighbourhood_cleansedOsgoode                 -2.186176e+01
## neighbourhood_cleansedRideau-Goulbourn         1.970330e+01
## neighbourhood_cleansedRideau-Rockcliffe       -8.638366e+00
## neighbourhood_cleansedRideau-Vanier            2.747876e+00
## neighbourhood_cleansedRiver                   -1.926635e+00
## neighbourhood_cleansedSomerset                 9.211956e+01
## neighbourhood_cleansedStittsville-Kanata West  .
## neighbourhood_cleansedWest Carleton-March      2.513699e+01
## latitude                                       .
## longitude                                      1.078330e+02
## accommodates                                   6.188338e+00
## bedrooms                                       3.434256e+01
## beds                                          -9.768057e+00
## minimum_nights                                 2.250069e-01
## maximum_nights                                -7.094830e-08
## has_availability0                              5.550456e+01
## number_of_reviews                             -1.726663e-02
## number_of_reviews_ltm                          3.847302e+00
## number_of_reviews_l30d                         1.856676e-01
## review_scores_rating                           .
## review_scores_accuracy                         .
## review_scores_cleanliness                      2.209868e+01
## review_scores_checkin                          3.751949e+00
## review_scores_communication                    2.300307e+01
## review_scores_location                        -1.071529e+01
## review_scores_value                           -2.710528e+01
## instant_bookable0                             -1.261257e+01
## calculated_host_listings_count                 .
## calculated_host_listings_count_entire_homes    .
## calculated_host_listings_count_private_rooms   1.408283e+00
## reviews_per_month                             -2.662334e+01
## bedroom                                       -1.350833e+01
## wifi                                           5.425367e+00
## heating                                        2.790146e+01
## walk                                           2.447594e+01
## university                                    -3.616422e+01
## refrigerator                                  -1.464859e+00
## downtown                                      -1.229204e+01
## parking                                        1.899674e+01
## water                                         -3.803156e+00
## alarm                                         -2.651582e+01
## bus                                           -2.315646e+01
## quiet                                          .
```

```r
test_error2 <- mse(y_test,lasso_pred)
test_error2
```

```
## [1] 8195.195
```

```r
#Standard regression
mod3 <- lm(price~.,data=selected_train)
summary(mod3)
```

```
##
## Call:
## lm(formula = price ~ ., data = selected_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -399.8  -83.8  -20.2   35.5 9488.5
##
## Coefficients: (1 not defined because of singularities)
##                                                  Estimate Std. Error t value
## (Intercept)                                     2.796e+04  6.009e+04   0.465
## host_response_timewithin a day                 -1.568e+02  2.302e+02  -0.681
## host_response_timewithin a few hours           -6.108e+01  2.446e+02  -0.250
## host_response_timewithin an hour               -1.402e+02  2.375e+02  -0.590
## host_response_rate                              1.433e+00  2.306e+00   0.621
## host_acceptance_rate                            3.512e-01  7.267e-01   0.483
## host_is_superhost0                             -7.055e+01  3.852e+01  -1.831
## host_listings_count                            -3.351e-01  1.413e+00  -0.237
## host_total_listings_count                              NA         NA      NA
## host_has_profile_pic0                          -1.530e+02  4.872e+02  -0.314
## host_identity_verified0                        -1.510e+01  4.697e+01  -0.322
## neighbourhood_cleansedBarrhaven                -3.477e+01  2.018e+02  -0.172
## neighbourhood_cleansedBay                      -2.349e+00  1.625e+02  -0.014
## neighbourhood_cleansedBeacon Hill-Cyrville     -8.897e+01  1.405e+02  -0.633
## neighbourhood_cleansedCapital                  -2.772e+01  1.072e+02  -0.259
## neighbourhood_cleansedCollege                  -4.789e+01  1.592e+02  -0.301
## neighbourhood_cleansedCumberland               -9.072e+01  1.915e+02  -0.474
## neighbourhood_cleansedGloucester-South Nepean -6.794e+01  1.509e+02  -0.450
## neighbourhood_cleansedGloucester-Southgate     -6.937e+01  1.378e+02  -0.504
## neighbourhood_cleansedInnes                    -9.417e+01  1.640e+02  -0.574
## neighbourhood_cleansedKanata North              1.275e+01  2.446e+02   0.052
## neighbourhood_cleansedKanata South             -1.824e+01  2.258e+02  -0.081
## neighbourhood_cleansedKitchissippi              1.364e+02  1.163e+02   1.172
## neighbourhood_cleansedKnoxdale-Merivale        -6.512e+01  1.493e+02  -0.436
## neighbourhood_cleansedOrleans                  -1.262e+02  1.812e+02  -0.696
## neighbourhood_cleansedOsgoode                  -9.141e+01  2.526e+02  -0.362
## neighbourhood_cleansedRideau-Goulbourn          3.600e+00  2.313e+02   0.016
## neighbourhood_cleansedRideau-Rockcliffe        -4.644e+01  1.230e+02  -0.377
## neighbourhood_cleansedRideau-Vanier            -1.559e+01  1.037e+02  -0.150
## neighbourhood_cleansedRiver                    -2.454e+01  1.238e+02  -0.198
## neighbourhood_cleansedSomerset                  7.827e+01  1.068e+02   0.733
## neighbourhood_cleansedStittsville-Kanata West   1.435e+01  2.558e+02   0.056
## neighbourhood_cleansedWest Carleton-March       9.118e+01  3.083e+02   0.296
## latitude                                       -5.158e+01  8.418e+02  -0.061
## longitude                                       3.387e+02  6.349e+02   0.533
## accommodates                                    8.442e+00  1.508e+01   0.560
## bedrooms                                        3.408e+01  3.221e+01   1.058
## beds                                           -1.247e+01  2.239e+01  -0.557
## minimum_nights                                  2.359e-01  3.621e-01   0.651
```

```
## maximum_nights                                -8.387e-08  2.384e-07  -0.352
## has_availability0                              8.529e+01  1.426e+02   0.598
## number_of_reviews                             -9.327e-03  5.921e-01  -0.016
## number_of_reviews_ltm                          4.190e+00  2.320e+00   1.806
## number_of_reviews_l30d                         2.199e-01  2.943e+01   0.007
## review_scores_rating                          -6.519e-01  5.860e+00  -0.111
## review_scores_accuracy                        -1.215e+00  3.925e+01  -0.031
## review_scores_cleanliness                      2.828e+01  3.608e+01   0.784
## review_scores_checkin                          4.591e+00  5.360e+01   0.086
## review_scores_communication                    2.661e+01  5.783e+01   0.460
## review_scores_location                        -1.131e+01  3.174e+01  -0.356
## review_scores_value                           -2.841e+01  3.588e+01  -0.792
## instant_bookable0                             -1.940e+01  4.263e+01  -0.455
## calculated_host_listings_count                -7.293e+01  1.234e+02  -0.591
## calculated_host_listings_count_entire_homes    7.272e+01  1.233e+02   0.590
## calculated_host_listings_count_private_rooms   7.804e+01  1.245e+02   0.627
## reviews_per_month                             -3.045e+01  3.470e+01  -0.877
## bedroom                                       -1.747e+01  3.544e+01  -0.493
## wifi                                           1.954e+01  1.452e+02   0.135
## heating                                        2.566e+01  1.227e+02   0.209
## walk                                           2.664e+01  3.827e+01   0.696
## university                                    -4.113e+01  5.252e+01  -0.783
## refrigerator                                  -6.324e+00  4.509e+01  -0.140
## downtown                                      -1.496e+01  3.407e+01  -0.439
## parking                                        2.102e+01  7.337e+01   0.286
## water                                         -6.987e+00  5.384e+01  -0.130
## alarm                                         -3.066e+01  1.167e+02  -0.263
## bus                                           -2.199e+01  3.381e+01  -0.650
## quiet                                         -3.873e-01  3.464e+01  -0.011
##                                               Pr(>|t|)
## (Intercept)                                     0.6418
## host_response_timewithin a day                  0.4959
## host_response_timewithin a few hours            0.8029
## host_response_timewithin an hour                0.5551
## host_response_rate                              0.5346
## host_acceptance_rate                            0.6290
## host_is_superhost0                              0.0674 .
## host_listings_count                             0.8126
## host_total_listings_count                           NA
## host_has_profile_pic0                           0.7536
## host_identity_verified0                         0.7479
## neighbourhood_cleansedBarrhaven                 0.8632
## neighbourhood_cleansedBay                       0.9885
## neighbourhood_cleansedBeacon Hill-Cyrville      0.5267
## neighbourhood_cleansedCapital                   0.7960
## neighbourhood_cleansedCollege                   0.7637
## neighbourhood_cleansedCumberland                0.6359
## neighbourhood_cleansedGloucester-South Nepean   0.6526
## neighbourhood_cleansedGloucester-Southgate      0.6147
## neighbourhood_cleansedInnes                     0.5660
## neighbourhood_cleansedKanata North              0.9584
## neighbourhood_cleansedKanata South              0.9356
## neighbourhood_cleansedKitchissippi              0.2415
## neighbourhood_cleansedKnoxdale-Merivale         0.6628
```

```
## neighbourhood_cleansedOrleans                    0.4864
## neighbourhood_cleansedOsgoode                    0.7176
## neighbourhood_cleansedRideau-Goulbourn           0.9876
## neighbourhood_cleansedRideau-Rockcliffe          0.7060
## neighbourhood_cleansedRideau-Vanier              0.8805
## neighbourhood_cleansedRiver                      0.8430
## neighbourhood_cleansedSomerset                   0.4638
## neighbourhood_cleansedStittsville-Kanata West    0.9553
## neighbourhood_cleansedWest Carleton-March        0.7675
## latitude                                         0.9512
## longitude                                        0.5938
## accommodates                                     0.5758
## bedrooms                                         0.2904
## beds                                             0.5777
## minimum_nights                                   0.5150
## maximum_nights                                   0.7250
## has_availability0                                0.5500
## number_of_reviews                                0.9874
## number_of_reviews_ltm                            0.0713 .
## number_of_reviews_l30d                           0.9940
## review_scores_rating                             0.9114
## review_scores_accuracy                           0.9753
## review_scores_cleanliness                        0.4334
## review_scores_checkin                            0.9318
## review_scores_communication                      0.6455
## review_scores_location                           0.7216
## review_scores_value                              0.4286
## instant_bookable0                                0.6492
## calculated_host_listings_count                   0.5547
## calculated_host_listings_count_entire_homes      0.5556
## calculated_host_listings_count_private_rooms     0.5311
## reviews_per_month                                0.3805
## bedroom                                          0.6223
## wifi                                             0.8930
## heating                                          0.8343
## walk                                             0.4865
## university                                       0.4338
## refrigerator                                     0.8885
## downtown                                         0.6607
## parking                                          0.7746
## water                                            0.8968
## alarm                                            0.7928
## bus                                              0.5156
## quiet                                            0.9911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 480.3 on 861 degrees of freedom
## Multiple R-squared:  0.04302,    Adjusted R-squared:  -0.03034
## F-statistic: 0.5864 on 66 and 861 DF,  p-value: 0.9964
```

```
reg_pred3 <- predict(mod3,selected_test)
```

```
## Warning in predict.lm(mod3, selected_test): prediction from a rank-deficient fit
```

```
## may be misleading
```

```r
test_error <- mse(selected_test$price,reg_pred3)
test_error
```

```
## [1] 9814.827
```

The test error for lasso regression is 8195.1953312.It is higher than the test error for lasso regression with suspicious points.
As expected, the test error for the standard regression (9814.8270392) is higher than that for lasso but it is also higher than the test error for standard regression with suspicious points
The new variables added to the model didn't improve the model.