Modèles de Régression Régularisée

CHHEANG Vinha & MECH Bealy

10/21/2021

II. Real estate data

```r
tab=read.table("housedata.csv",header=TRUE,sep=',')
head(tab)
```

```
##           id            date   price bedrooms bathrooms sqft_living sqft_lot
## 1 7129300520 20141013T000000  221900        3      1.00        1180     5650
## 2 6414100192 20141209T000000  538000        3      2.25        2570     7242
## 3 5631500400 20150225T000000  180000        2      1.00         770    10000
## 4 2487200875 20141209T000000  604000        4      3.00        1960     5000
## 5 1954400510 20150218T000000  510000        3      2.00        1680     8080
## 6 7237550310 20140512T000000 1225000        4      4.50        5420   101930
##   floors waterfront view condition grade sqft_above sqft_basement yr_built
## 1      1          0    0         3     7       1180             0     1955
## 2      2          0    0         3     7       2170           400     1951
## 3      1          0    0         3     6        770             0     1933
## 4      1          0    0         5     7       1050           910     1965
## 5      1          0    0         3     8       1680             0     1987
## 6      1          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

```r
names(tab)
```

```
##  [1] "id"            "date"          "price"         "bedrooms"
##  [5] "bathrooms"     "sqft_living"   "sqft_lot"      "floors"
##  [9] "waterfront"    "view"          "condition"     "grade"
## [13] "sqft_above"    "sqft_basement" "yr_built"      "yr_renovated"
## [17] "zipcode"       "lat"           "long"          "sqft_living15"
## [21] "sqft_lot15"
```

```r
dim(tab)
```

```
## [1] 21613    21
```

```r
medianHousePrice=median(tab$price)
medHousePriceBin=as.numeric(tab$price > medianHousePrice)
head(medHousePriceBin)
```

```
## [1] 0 1 0 1 1 1
```

Right now, to be able apply Logistic Regression we will build a new data with 'medHousePriceBin'.

```r
tab1 <- tab[, 4:21]
X <- cbind(tab1, medHousePriceBin)
head(X)
```

```
##   bedrooms bathrooms sqft_living sqft_lot floors waterfront view condition
## 1        3      1.00        1180     5650      1          0    0         3
## 2        3      2.25        2570     7242      2          0    0         3
## 3        2      1.00         770    10000      1          0    0         3
## 4        4      3.00        1960     5000      1          0    0         5
## 5        3      2.00        1680     8080      1          0    0         3
## 6        4      4.50        5420   101930      1          0    0         3
##   grade sqft_above sqft_basement yr_built yr_renovated zipcode     lat     long
## 1     7       1180             0     1955            0   98178 47.5112 -122.257
## 2     7       2170           400     1951         1991   98125 47.7210 -122.319
## 3     6        770             0     1933            0   98028 47.7379 -122.233
## 4     7       1050           910     1965            0   98136 47.5208 -122.393
## 5     8       1680             0     1987            0   98074 47.6168 -122.045
## 6    11       3890          1530     2001            0   98053 47.6561 -122.005
##   sqft_living15 sqft_lot15 medHousePriceBin
## 1          1340       5650                0
## 2          1690       7639                1
## 3          2720       8062                0
## 4          1360       5000                1
## 5          1800       7503                1
## 6          4760     101930                1
```

```r
str(X)
```

```
## 'data.frame':    21613 obs. of  19 variables:
##  $ bedrooms        : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms       : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living     : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot        : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors          : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition       : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade           : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above      : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement   : int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built        : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated    : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode         : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat             : num  47.5 47.7 47.7 47.5 47.6 ...
```

```
##  $ long            : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15   : int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15      : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
##  $ medHousePriceBin: num  0 1 0 1 1 1 0 0 0 0 ...
```

```r
dim(X)
```

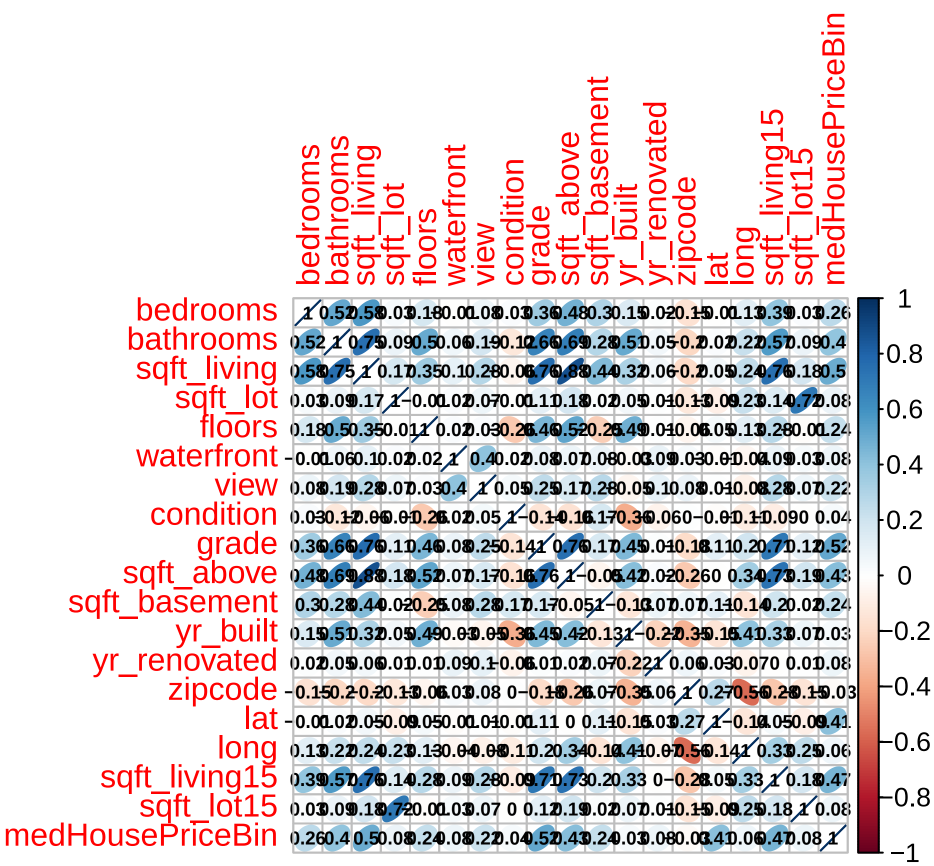
```
## [1] 21613    19
```

There are $n = 21613$ observation and $p = 18$ variables.

```r
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```r
corrplot(cor(X), is.corr = T, method = "ellipse", number.cex= .6, addCoef.col= "black")
```



1. Split the dataset into training and testing sets:

If we don't want to over-evaluate the model, we cannot test it on the data it has already used during the training process. So we decide to split the dataset into 80% for training and 20% for testing.

```r
set.seed(1234)
training_indexes <- sample(1:21613, size=round(21613*0.8))
# the training dataset
X_train <- X[training_indexes,]
# the testing dataset
X_test <- X[-training_indexes,]
testing_indexes <- as.numeric(rownames(X_test))
# medHousePriceBin, the target values to test (20%)
Y_test  <- medHousePriceBin[testing_indexes]
Y_train <- medHousePriceBin[training_indexes]
```

2. Simple Logistic Regression:

```r
res = glm(medHousePriceBin~. , family = binomial, data =X_train)
summary(res)
```

```
##
## Call:
## glm(formula = medHousePriceBin ~ ., family = binomial, data = X_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4712  -0.4754  -0.0423   0.4733   4.0103
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.044e+02  5.273e+01  -1.980  0.04771 *
## bedrooms      -2.452e-01  3.442e-02  -7.124 1.05e-12 ***
## bathrooms      4.672e-01  5.729e-02   8.154 3.51e-16 ***
## sqft_living    1.312e-03  8.397e-05  15.622  < 2e-16 ***
## sqft_lot       5.241e-06  1.180e-06   4.440 9.00e-06 ***
## floors         7.207e-01  5.935e-02  12.143  < 2e-16 ***
## waterfront     2.380e+00  5.237e-01   4.544 5.51e-06 ***
## view           4.414e-01  4.680e-02   9.432  < 2e-16 ***
## condition      3.137e-01  4.025e-02   7.795 6.43e-15 ***
## grade          1.264e+00  4.357e-02  29.007  < 2e-16 ***
## sqft_above    -2.945e-04  8.066e-05  -3.652  0.00026 ***
## sqft_basement         NA         NA      NA       NA
## yr_built      -3.403e-02  1.312e-03 -25.947  < 2e-16 ***
## yr_renovated   3.431e-05  6.624e-05   0.518  0.60453
## zipcode       -3.301e-03  6.110e-04  -5.403 6.54e-08 ***
## lat            1.025e+01  2.187e-01  46.890  < 2e-16 ***
## long           6.743e-02  2.321e-01   0.291  0.77142
## sqft_living15  9.014e-04  6.774e-05  13.306  < 2e-16 ***
## sqft_lot15    -4.328e-08  1.654e-06  -0.026  0.97913
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 23968  on 17289  degrees of freedom
```

```
## Residual deviance: 11780  on 17272  degrees of freedom
## AIC: 11816
##
## Number of Fisher Scoring iterations: 6
```

- For instant, we can conclude that there are 3 variables (" sqft_lot15 " , “long” " yr_renovated ") are not influence on the model logistic with variable target: ‘medHousePriceBin’.
- For the variable “sqft_basement”, we don’t know information yet, so we can not conclude that it’s useful for model or not. However, we will check it by another method.

Let’s predict some testing values and compare with the target values:

```
Y_pred <- round(predict(res, X_test[,1:18], type = "response"))
```

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

```
Y_pred[1:10]
```

```
##  2  4  5  8  9 11 17 19 20 21
##  1  0  0  0  0  1  1  0  0  1
```

```
Y_test[1:10]
```

```
##  [1] 1 1 1 0 0 1 0 0 0 0
```

To be able to clearly the result, we will check the Confusion Matrix:

```
# Confusion Matrix:
library(caret)
```

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

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

```
confusionMatrix(data = as.factor(Y_pred),
                reference = as.factor(Y_test),positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1843  355
##          1  320 1805
##
##                Accuracy : 0.8439
##                  95% CI : (0.8327, 0.8546)
##     No Information Rate : 0.5003
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.6877
##
##  Mcnemar's Test P-Value : 0.1906
##
##             Sensitivity : 0.8356
##             Specificity : 0.8521
##          Pos Pred Value : 0.8494
##          Neg Pred Value : 0.8385
##              Prevalence : 0.4997
##          Detection Rate : 0.4175
##    Detection Prevalence : 0.4916
##       Balanced Accuracy : 0.8439
##
##        'Positive' Class : 1
##
```

- The total accuracy is 0.8439 on data test which is pretty good.
- The value of Sensitivity (true positive rate) is: 0.8356
- The value of specificity (true negative rate) is: 0.8521

We obtained the value of specificity is greater than the value of the sensitivity (recall) which means that the model is well performance to predict the "0" than the "1" in the binary value.

3. Logistic Regression with K-Folds Cross-Validation:

We use in each fold, 80% of the dataset for training and the rest of 20% for testing in k-folds, k=5.

```
set.seed(1234)
library("caret")
cross5 = train(
form = as.factor(medHousePriceBin) ~ .,
data = X,
trControl = trainControl(method = "cv", number = 5, savePredictions = TRUE, p = 0.8),
method = "glm",
family = "binomial")
```

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

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

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

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

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

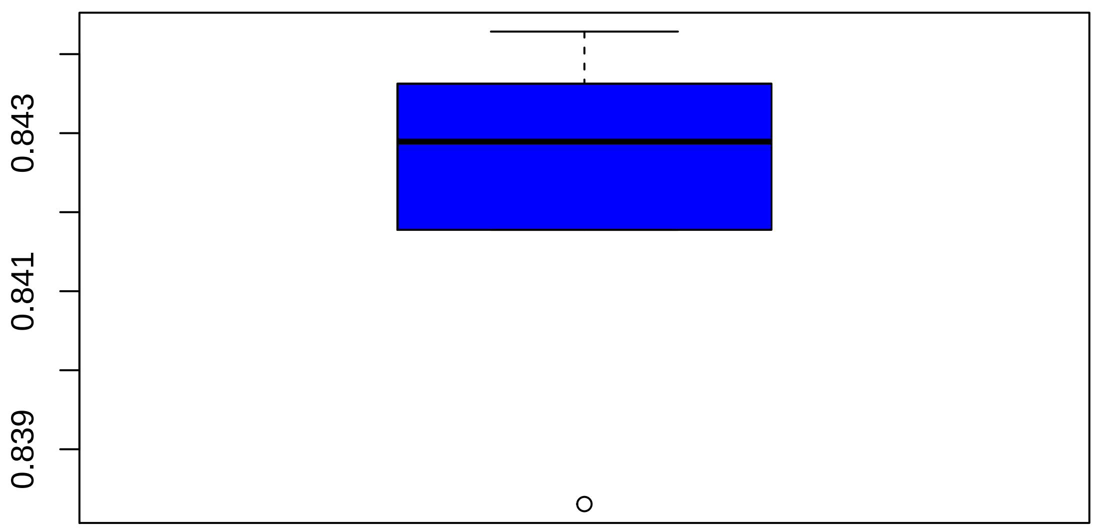
The accuracy in each fold:

```
cross5$resample$Accuracy
```

```
## [1] 0.8436271 0.8428968 0.8442851 0.8383067 0.8417765
```

The boxplot of the accuracy of the performance for the model in k-fold:

```
boxplot(cross5$resample$Accuracy, col = "blue")
```



Average accuracy:

```
cross5$results$Accuracy
```

```
## [1] 0.8421784
```

In average, using 80% of the X to train the model leads to a performance (accuracy) of 84% with k_fold, $k = 5$.

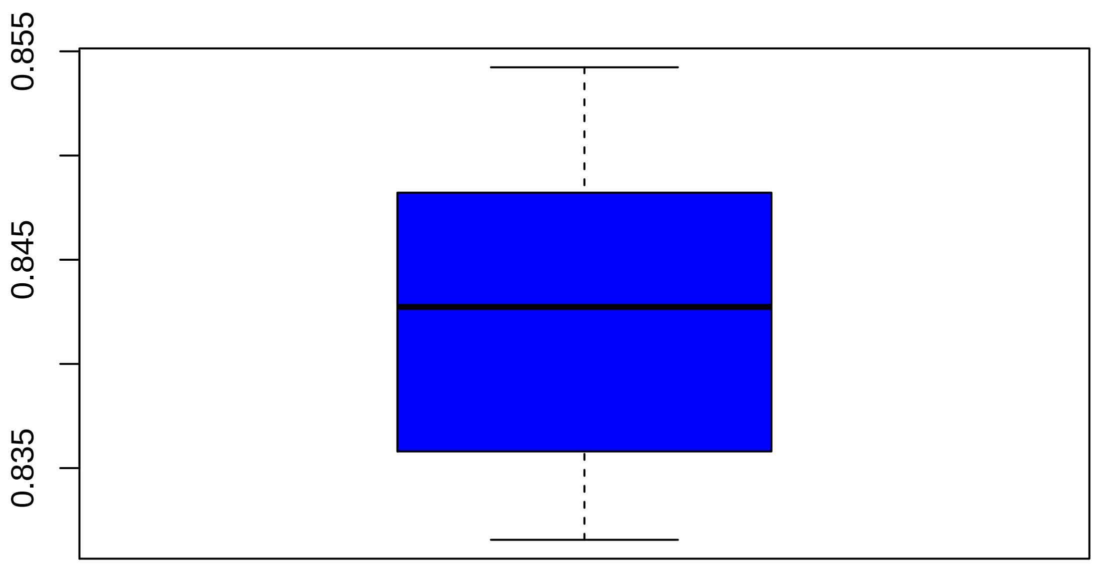
Now we will verify with k = 10, with datatrain 90%

```
set.seed(1234)
library("caret")
cross10 = train(
form = as.factor(medHousePriceBin) ~ ., data = X,
trControl = trainControl(method = "cv", number = 10, savePredictions = TRUE, p = 0.9),
method = "glm",
family = "binomial")
```

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

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

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

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

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

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

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

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

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

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

```
boxplot(cross10$resample$Accuracy, col = "blue")
```



Average accuracy:

```
cross5$results$Accuracy
```

```
## [1] 0.8421784
```

In average, using 90% of the X to train the model leads to a performance (accuracy) of 84% with k_fold, $k = 10$.

Therefore, the accuracy of $k - fold$ is 84%. However, if we train more data we will get the same result. So we use 80% of datatrain for create model.

4. Regularization + Variable selection

Variable selection with Regression Logistic forward We all data for selection in $Xdata$.

```
# Regression logistique Forward
resall <- glm(medHousePriceBin~., data = X, family = binomial)
res0   <- glm(medHousePriceBin~1, data = X, family = binomial)
resforward <- step(res0, list(upper=resall), direction = "forward")
```

```
## Start:  AIC=29963.37
## medHousePriceBin ~ 1
##
##                 Df Deviance   AIC
## + grade          1    22697 22701
## + sqft_living    1    23053 23057
## + sqft_living15  1    24380 24384
## + sqft_above     1    25270 25274
## + lat            1    25994 25998
## + bathrooms      1    26201 26205
## + bedrooms       1    28349 28353
## + sqft_basement  1    28698 28702
## + floors         1    28712 28716
## + view           1    28788 28792
## + sqft_lot       1    29763 29767
## + sqft_lot15     1    29810 29814
## + waterfront     1    29814 29818
## + yr_renovated   1    29830 29834
## + long           1    29882 29886
## + condition      1    29933 29937
## + yr_built       1    29938 29942
## + zipcode        1    29943 29947
## <none>                29961 29963
##
## Step:  AIC=22700.91
## medHousePriceBin ~ grade
##
##                 Df Deviance   AIC
## + lat            1    18705 18711
## + yr_built       1    20706 20712
## + sqft_living    1    21569 21575
## + sqft_basement  1    21802 21808
## + sqft_living15  1    22023 22029
## + condition      1    22248 22254
## + view           1    22277 22283
## + bedrooms       1    22505 22511
## + zipcode        1    22530 22536
## + sqft_above     1    22546 22552
## + yr_renovated   1    22549 22555
## + bathrooms      1    22589 22595
## + waterfront     1    22604 22610
## + long           1    22611 22617
## + sqft_lot       1    22656 22662
## + floors         1    22674 22680
## + sqft_lot15     1    22677 22683
```

```
## <none>                22697 22701
##
## Step:  AIC=18711.19
## medHousePriceBin ~ grade + lat
##
##                 Df Deviance   AIC
## + sqft_living    1    16877 16885
## + sqft_living15  1    17526 17534
## + yr_built       1    17534 17542
## + sqft_basement  1    18034 18042
## + sqft_above     1    18062 18070
## + view           1    18116 18124
## + condition      1    18216 18224
## + bedrooms       1    18329 18337
## + bathrooms      1    18424 18432
## + sqft_lot       1    18483 18491
## + sqft_lot15     1    18531 18539
## + waterfront     1    18545 18553
## + yr_renovated   1    18576 18584
## + floors         1    18678 18686
## + zipcode        1    18694 18702
## <none>                18705 18711
## + long           1    18705 18713
##
## Step:  AIC=16884.98
## medHousePriceBin ~ grade + lat + sqft_living
##
##                 Df Deviance   AIC
## + yr_built       1    15727 15737
## + view           1    16443 16453
## + condition      1    16535 16545
## + sqft_living15  1    16647 16657
## + waterfront     1    16732 16742
## + sqft_lot       1    16774 16784
## + yr_renovated   1    16796 16806
## + sqft_lot15     1    16808 16818
## + long           1    16829 16839
## + bedrooms       1    16842 16852
## + sqft_basement  1    16845 16855
## + sqft_above     1    16845 16855
## + bathrooms      1    16849 16859
## + floors         1    16854 16864
## <none>                16877 16885
## + zipcode        1    16876 16886
##
## Step:  AIC=15737.43
## medHousePriceBin ~ grade + lat + sqft_living + yr_built
##
##                 Df Deviance   AIC
## + sqft_living15  1    15467 15479
## + view           1    15469 15481
## + bathrooms      1    15612 15624
## + waterfront     1    15617 15629
## + floors         1    15628 15640
```

```
## + sqft_lot       1    15633 15645
## + sqft_lot15     1    15659 15671
## + condition      1    15665 15677
## + zipcode        1    15673 15685
## + bedrooms       1    15680 15692
## + long           1    15700 15712
## + sqft_basement  1    15724 15736
## + sqft_above     1    15724 15736
## + yr_renovated   1    15725 15737
## <none>                15727 15737
##
## Step:  AIC=15478.74
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15
##
##                 Df Deviance   AIC
## + view           1    15256 15270
## + floors         1    15309 15323
## + bathrooms      1    15314 15328
## + waterfront     1    15363 15377
## + sqft_lot       1    15393 15407
## + condition      1    15401 15415
## + sqft_lot15     1    15422 15436
## + bedrooms       1    15429 15443
## + zipcode        1    15454 15468
## + yr_renovated   1    15461 15475
## <none>                15467 15479
## + long           1    15466 15480
## + sqft_above     1    15466 15480
## + sqft_basement  1    15466 15480
##
## Step:  AIC=15270.28
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view
##
##                 Df Deviance   AIC
## + floors         1    15098 15114
## + bathrooms      1    15111 15127
## + condition      1    15188 15204
## + sqft_lot       1    15194 15210
## + sqft_lot15     1    15220 15236
## + zipcode        1    15221 15237
## + waterfront     1    15227 15243
## + bedrooms       1    15230 15246
## + long           1    15248 15264
## + yr_renovated   1    15252 15268
## <none>                15256 15270
## + sqft_above     1    15255 15271
## + sqft_basement  1    15255 15271
##
## Step:  AIC=15113.46
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors
##
##                 Df Deviance   AIC
```

```
## + condition      1    14998 15016
## + bathrooms      1    15004 15022
## + sqft_lot       1    15019 15037
## + zipcode        1    15031 15049
## + sqft_lot15     1    15047 15065
## + waterfront     1    15071 15089
## + bedrooms       1    15073 15091
## + long           1    15076 15094
## + sqft_basement  1    15076 15094
## + sqft_above     1    15076 15094
## <none>                15098 15114
## + yr_renovated   1    15096 15114
##
## Step:  AIC=15016.14
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition
##
##                 Df Deviance   AIC
## + sqft_lot       1    14917 14937
## + bathrooms      1    14922 14942
## + zipcode        1    14944 14964
## + sqft_lot15     1    14948 14968
## + bedrooms       1    14969 14989
## + waterfront     1    14973 14993
## + long           1    14977 14997
## + sqft_above     1    14983 15003
## + sqft_basement  1    14983 15003
## + yr_renovated   1    14992 15012
## <none>                14998 15016
##
## Step:  AIC=14937.04
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot
##
##                 Df Deviance   AIC
## + bathrooms      1    14831 14853
## + zipcode        1    14875 14897
## + waterfront     1    14891 14913
## + bedrooms       1    14895 14917
## + sqft_basement  1    14895 14917
## + sqft_above     1    14895 14917
## + long           1    14908 14930
## + yr_renovated   1    14912 14934
## <none>                14917 14937
## + sqft_lot15     1    14917 14939
##
## Step:  AIC=14853.44
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot + bathrooms
##
##                 Df Deviance   AIC
## + bedrooms       1    14788 14812
## + zipcode        1    14790 14814
## + waterfront     1    14804 14828
```

```
## + long           1    14820 14844
## + sqft_basement  1    14822 14846
## + sqft_above     1    14822 14846
## <none>                14831 14853
## + yr_renovated   1    14830 14854
## + sqft_lot15     1    14831 14855
##
## Step:  AIC=14811.5
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot + bathrooms + bedrooms
##
##                 Df Deviance   AIC
## + zipcode        1    14741 14767
## + waterfront     1    14762 14788
## + long           1    14777 14803
## + sqft_above     1    14778 14804
## + sqft_basement  1    14778 14804
## <none>                14788 14812
## + yr_renovated   1    14786 14812
## + sqft_lot15     1    14787 14813
##
## Step:  AIC=14767.02
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot + bathrooms + bedrooms +
##     zipcode
##
##                 Df Deviance   AIC
## + waterfront     1    14717 14745
## + sqft_basement  1    14724 14752
## + sqft_above     1    14724 14752
## <none>                14741 14767
## + yr_renovated   1    14740 14768
## + sqft_lot15     1    14741 14769
## + long           1    14741 14769
##
## Step:  AIC=14744.62
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot + bathrooms + bedrooms +
##     zipcode + waterfront
##
##                 Df Deviance   AIC
## + sqft_basement  1    14698 14728
## + sqft_above     1    14698 14728
## <none>                14717 14745
## + yr_renovated   1    14716 14746
## + sqft_lot15     1    14717 14747
## + long           1    14717 14747
##
## Step:  AIC=14728.29
## medHousePriceBin ~ grade + lat + sqft_living + yr_built + sqft_living15 +
##     view + floors + condition + sqft_lot + bathrooms + bedrooms +
##     zipcode + waterfront + sqft_basement
##
##                Df Deviance   AIC
```

```
## <none>                14698 14728
## + long           1    14698 14730
## + yr_renovated   1    14698 14730
## + sqft_lot15     1    14698 14730
```

The final computed model:

```r
summary(resforward)
```

```
##
## Call:
## glm(formula = medHousePriceBin ~ grade + lat + sqft_living +
##     yr_built + sqft_living15 + view + floors + condition + sqft_lot +
##     bathrooms + bedrooms + zipcode + waterfront + sqft_basement,
##     family = binomial, data = X)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7051  -0.4765  -0.0407   0.4713   4.0197
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.801e+01  4.706e+01  -2.083   0.0373 *
## grade           1.289e+00  3.859e-02  33.413  < 2e-16 ***
## lat             1.032e+01  1.959e-01  52.685  < 2e-16 ***
## sqft_living     9.741e-04  6.454e-05  15.093  < 2e-16 ***
## yr_built       -3.431e-02  1.111e-03 -30.872  < 2e-16 ***
## sqft_living15   9.060e-04  6.032e-05  15.021  < 2e-16 ***
## view            4.643e-01  4.291e-02  10.819  < 2e-16 ***
## floors          6.845e-01  5.279e-02  12.966  < 2e-16 ***
## condition       2.954e-01  3.561e-02   8.295  < 2e-16 ***
## sqft_lot        5.427e-06  6.967e-07   7.790 6.72e-15 ***
## bathrooms       4.850e-01  5.122e-02   9.468  < 2e-16 ***
## bedrooms       -2.064e-01  3.089e-02  -6.683 2.34e-11 ***
## zipcode        -3.479e-03  4.773e-04  -7.288 3.15e-13 ***
## waterfront      2.283e+00  4.815e-01   4.741 2.13e-06 ***
## sqft_basement   3.050e-04  7.129e-05   4.278 1.88e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29961  on 21612  degrees of freedom
## Residual deviance: 14698  on 21598  degrees of freedom
## AIC: 14728
##
## Number of Fisher Scoring iterations: 6
```

```r
Y_forward <- round(predict(resforward, X_test[,1:18], type="response"))

confusionMatrix(data = as.factor(Y_forward), reference = as.factor(Y_test), positive='1')
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    0    1
##          0 1843  357
##          1  320 1803
##
##                Accuracy : 0.8434
##                  95% CI : (0.8322, 0.8541)
##     No Information Rate : 0.5003
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6868
##
##  Mcnemar's Test P-Value : 0.1665
##
##             Sensitivity : 0.8347
##             Specificity : 0.8521
##          Pos Pred Value : 0.8493
##          Neg Pred Value : 0.8377
##              Prevalence : 0.4997
##          Detection Rate : 0.4171
##    Detection Prevalence : 0.4911
##       Balanced Accuracy : 0.8434
##
##        'Positive' Class : 1
##
```

Regression logistique Forward:

- Accuracy = 0.8434
- Sensitivity (true positive rate) = 0.8347
- Specificity (true negative rate) = 0.8521

5. Logistic regression with $l1$ or $l2$ penalizations

a) Logistic regression with $l2$ penalization (Ridge)

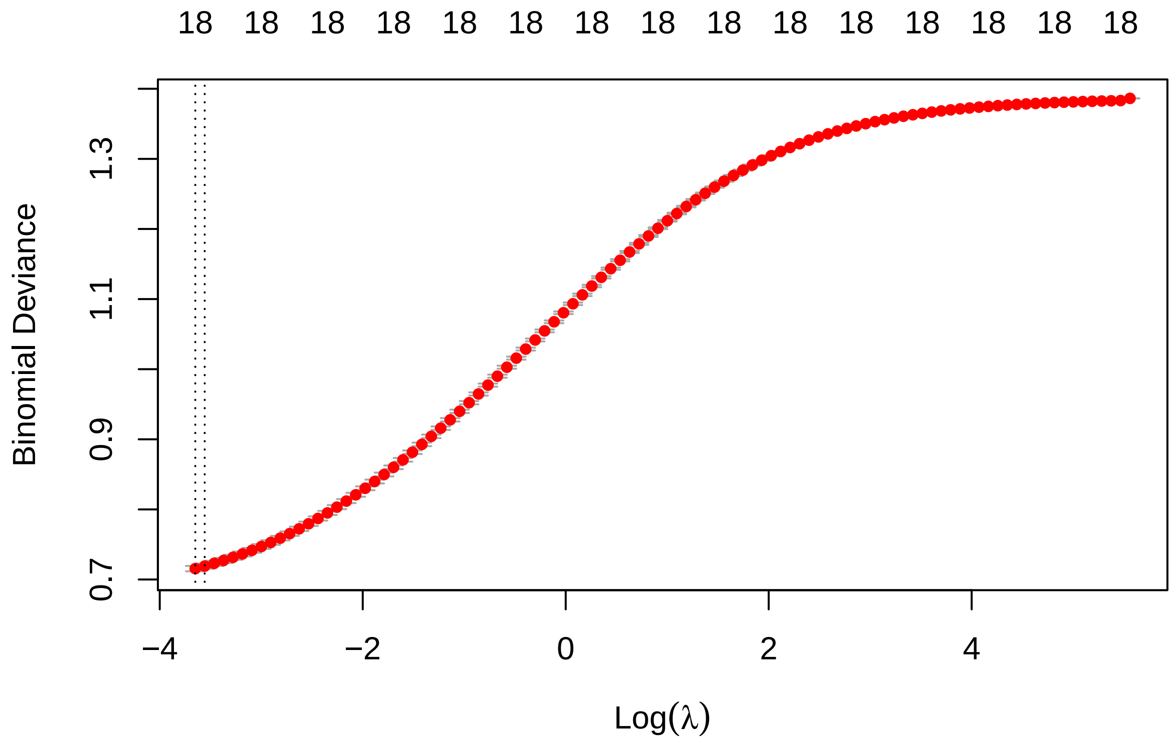
- $\alpha = 0$ it is Ridge Logistic Regression.
- In Ridge regression, we use all the variables to test then chose the best one which has the smallest value of *lambda*.

```r
set.seed(1234)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-3
```

```r
mod_ridge <- cv.glmnet(x=as.matrix(X[,1:18]),
                       y=medHousePriceBin, alpha = 0, n_fold=5, family = "binomial")
plot(mod_ridge)
```



- Binomial deviance $\approx$ error of the model.
- Log(λ) = penalizing coefficients
- When the penalizing coefficients are low, so are the binomial deviances, meaning that the model is probably better.

The penalizing coefficient giving the lowest binomial deviance:

```
mod_ridge$lambda.min
```

```
## [1] 0.02599242
```

```
library("glmnet")
mod_ridge_best <- glmnet(as.matrix(X_train[,1:18]),
                         Y_train, family = "binomial", alpha = 0, lambda = mod_ridge$lambda.min)
coef(mod_ridge_best)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  -2.061742e+02
## bedrooms     -8.187667e-02
## bathrooms     3.111216e-01
## sqft_living   4.467194e-04
## sqft_lot      2.823002e-06
## floors        4.504263e-01
## waterfront    1.226646e+00
## view          3.193111e-01
## condition     2.687099e-01
## grade         6.914330e-01
```

```
## sqft_above      3.951695e-04
## sqft_basement  5.473239e-04
## yr_built      -1.747249e-02
## yr_renovated   1.369549e-04
## zipcode       -1.392924e-03
## lat            7.199961e+00
## long          -1.993721e-01
## sqft_living15  6.972992e-04
## sqft_lot15     1.106278e-06
```

```r
preds_ridge <- round(predict(mod_ridge_best,
                             as.matrix(X_test[,1:18]), type = "response"))

conf_ridge_best <- confusionMatrix(data = as.factor(preds_ridge),
                                   reference = as.factor(Y_test), positive = '1')
conf_ridge_best
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1857  380
##          1  306 1780
##
##                Accuracy : 0.8413
##                  95% CI : (0.8301, 0.8521)
##     No Information Rate : 0.5003
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6826
##
##  Mcnemar's Test P-Value : 0.005317
##
##             Sensitivity : 0.8241
##             Specificity : 0.8585
##          Pos Pred Value : 0.8533
##          Neg Pred Value : 0.8301
##              Prevalence : 0.4997
##          Detection Rate : 0.4118
##    Detection Prevalence : 0.4825
##       Balanced Accuracy : 0.8413
##
##        'Positive' Class : 1
##
```

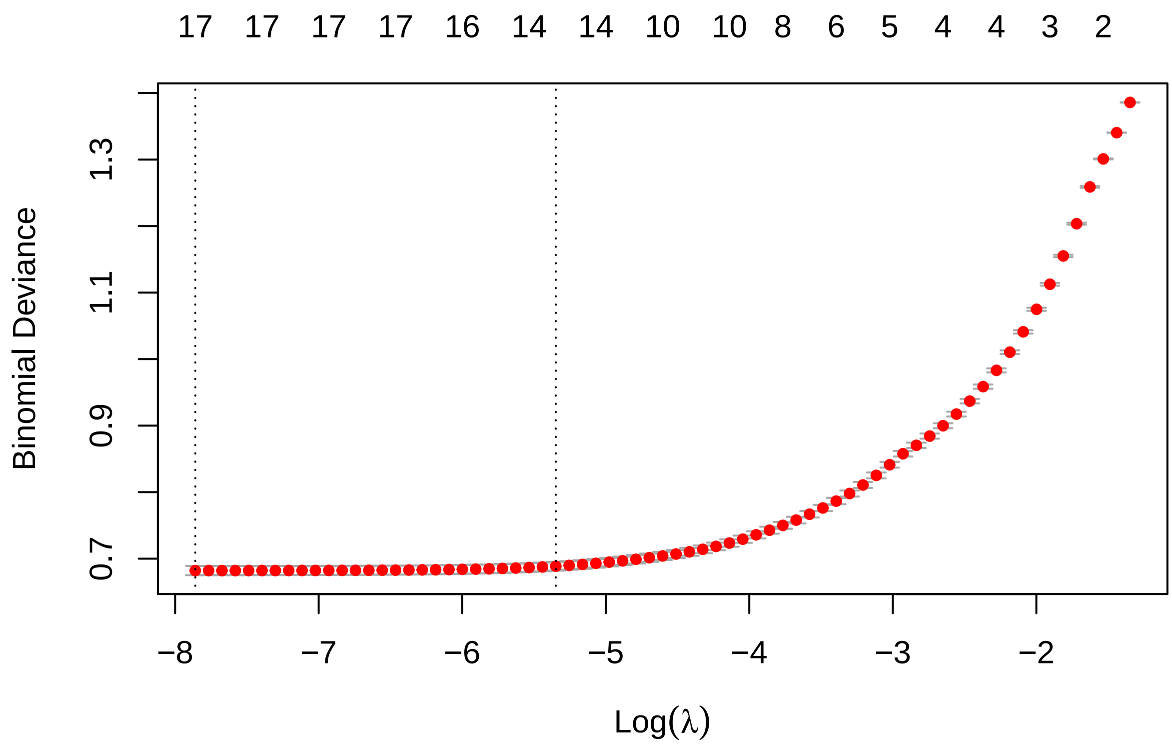
Ridge Logistic Regression:

- Accuracy = 0.8413
- Sensitivity (true positive rate) = 0.8241
- Specificity (true negative rate) = 0.8585

b) Logistic regression with l penalization (Lasso)

- $\alpha = 1$ it is: Lasso Logistic Regression.

```
library("glmnet")
mod_lasso <- cv.glmnet(x=as.matrix(X[,1:18]), y=medHousePriceBin,
                       alpha = 1, n_fold=5, family = "binomial")
plot(mod_lasso)
```




```
mod_lasso$lambda.min
```

```
## [1] 0.0003859791
```

```
mod_lasso$lambda.1se
```

```
## [1] 0.004758531
```

Let's run a lasso logit with the penalizing coefficient = lambda.1se:

```
library("glmnet")
mod_lasso_1se <- glmnet(as.matrix(X_train[,1:18]), Y_train,
                        family = "binomial", alpha = 1, lambda = mod_lasso$lambda.1se)
coef(mod_lasso_1se)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                           s0
## (Intercept)    -2.482034e+02
## bedrooms       -8.282872e-02
## bathrooms       3.287398e-01
## sqft_living     8.994067e-04
## sqft_lot        3.486541e-06
```

```
## floors          4.863792e-01
## waterfront      1.044707e+00
## view            3.696376e-01
## condition       2.505922e-01
## grade           1.120778e+00
## sqft_above      .
## sqft_basement   1.154542e-04
## yr_built       -2.655256e-02
## yr_renovated    .
## zipcode        -1.501893e-03
## lat             9.122108e+00
## long            .
## sqft_living15   7.977641e-04
## sqft_lot15      .
```

```
preds_lasso <- round(predict(mod_lasso_1se, as.matrix(X_test[,1:18]), type = "response"))

conf_best_lasso <- confusionMatrix(data = as.factor(preds_lasso),
                                   reference = as.factor(Y_test),positive = '1')

conf_best_lasso
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1843  356
##          1  320 1804
##
##                Accuracy : 0.8436
##                  95% CI : (0.8325, 0.8543)
##     No Information Rate : 0.5003
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6873
##
##  Mcnemar's Test P-Value : 0.1783
##
##             Sensitivity : 0.8352
##             Specificity : 0.8521
##          Pos Pred Value : 0.8493
##          Neg Pred Value : 0.8381
##              Prevalence : 0.4997
##          Detection Rate : 0.4173
##    Detection Prevalence : 0.4913
##       Balanced Accuracy : 0.8436
##
##        'Positive' Class : 1
##
```

Lasso Logistic Regression:

- Accuracy = 0.8448

- Sensitivity (true positive rate) = 0.8366
- Specificity (true negative rate) = 0.8530 .

Therefore, Logistic Regression above are the almost the same result:

- Accuracy = 84%
- Sensitivity (true positive rate) = 83%
- Specificity (true negative rate) = 85%

We can say that: 2% Model better on 0 than 1.

6. Conclusion:

```
set.seed(1234)
errors_full <- sum((Y_pred - Y_test)^2)
errors_forward <- sum((Y_forward - Y_test)^2)
errors_ridge <- sum((preds_ridge - Y_test)^2)
errors_lasso <- sum((preds_lasso - Y_test)^2)

c(errors_full, errors_forward, errors_ridge, errors_lasso)
```

```
## [1] 675 677 686 676
```

```
length(Y_test)
```

```
## [1] 4323
```

```
# the errors made during the test
c(errors_full, errors_forward, errors_ridge, errors_lasso)/length(Y_test)
```

```
## [1] 0.1561416 0.1566042 0.1586861 0.1563729
```