InsuranceDataSet-HW2-P2

Jeetendra Gan

The dimension of the complete data is:

```
## [1] 5822   86
```

Here are the features in the data. V86 is the response variable, and the rest are predictors.

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11"
## [12] "V12" "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22"
## [23] "V23" "V24" "V25" "V26" "V27" "V28" "V29" "V30" "V31" "V32" "V33"
## [34] "V34" "V35" "V36" "V37" "V38" "V39" "V40" "V41" "V42" "V43" "V44"
## [45] "V45" "V46" "V47" "V48" "V49" "V50" "V51" "V52" "V53" "V54" "V55"
## [56] "V56" "V57" "V58" "V59" "V60" "V61" "V62" "V63" "V64" "V65" "V66"
## [67] "V67" "V68" "V69" "V70" "V71" "V72" "V73" "V74" "V75" "V76" "V77"
## [78] "V78" "V79" "V80" "V81" "V82" "V83" "V84" "V85" "V86"
```

I have split the training data from the trcdata2000.txt file into a train set and a test set. The training set is 75% of the total.

Fit the ordinary least squares solution to the model.

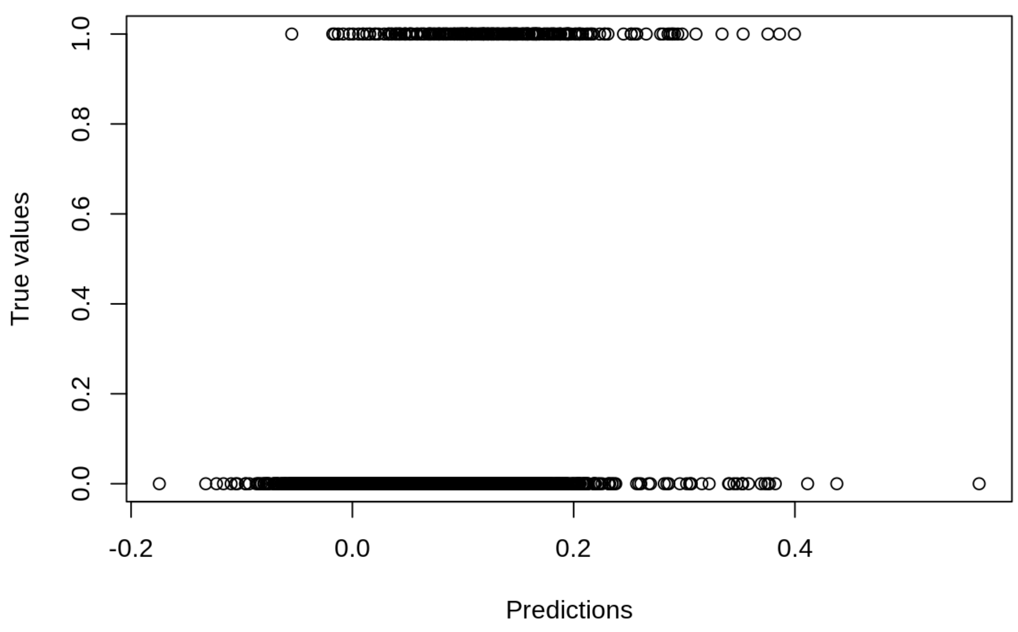
Here is the summary after fitting an ordinary least sequares solution to the model.

```
##
## Call:
## lm(formula = V86 ~ ., data = trainTrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.56649 -0.09176 -0.04766 -0.00524  1.05478
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.6740863  0.5189858   1.299  0.19406
## V1           0.0028396  0.0026725   1.063  0.28806
## V2          -0.0097853  0.0089218  -1.097  0.27280
## V3          -0.0031586  0.0084684  -0.373  0.70918
## V4           0.0100450  0.0058252   1.724  0.08471 .
## V5          -0.0127122  0.0119862  -1.061  0.28895
## V6          -0.0074704  0.0066094  -1.130  0.25843
## V7          -0.0004716  0.0071275  -0.066  0.94725
## V8           0.0026842  0.0064372   0.417  0.67671
## V9          -0.0053216  0.0068250  -0.780  0.43560
## V10          0.0090272  0.0089972   1.003  0.31575
## V11          0.0021400  0.0085660   0.250  0.80273
## V12          0.0063514  0.0090744   0.700  0.48402
## V13         -0.0030948  0.0077206  -0.401  0.68855
## V14         -0.0022258  0.0078028  -0.285  0.77546
## V15          0.0021126  0.0081462   0.259  0.79539
## V16          0.0086951  0.0080439   1.081  0.27978
## V17          0.0022628  0.0084424   0.268  0.78869
## V18         -0.0041721  0.0086358  -0.483  0.62904
## V19          0.0003183  0.0053643   0.059  0.95269
## V20         -0.0018550  0.0061694  -0.301  0.76367
## V21         -0.0052154  0.0059295  -0.880  0.37914
## V22          0.0036525  0.0053373   0.684  0.49380
## V23         -0.0013816  0.0052701  -0.262  0.79322
## V24          0.0004114  0.0052584   0.078  0.93764
## V25          0.0004786  0.0060973   0.078  0.93744
## V26         -0.0014394  0.0059685  -0.241  0.80944
## V27         -0.0013461  0.0052760  -0.255  0.79862
## V28          0.0029136  0.0058418   0.499  0.61798
## V29          0.0011302  0.0055703   0.203  0.83922
## V30         -0.0394718  0.0501241  -0.787  0.43104
## V31         -0.0374001  0.0500970  -0.747  0.45537
## V32          0.0086731  0.0088942   0.975  0.32954
## V33          0.0096759  0.0080864   1.197  0.23154
## V34          0.0051551  0.0085624   0.602  0.54717
## V35         -0.0570658  0.0540136  -1.057  0.29080
## V36         -0.0599969  0.0539404  -1.112  0.26608
## V37          0.0030485  0.0060569   0.503  0.61478
## V38          0.0027502  0.0058239   0.472  0.63678
## V39          0.0004343  0.0059557   0.073  0.94188
## V40          0.0004944  0.0062300   0.079  0.93675
## V41         -0.0118761  0.0079618  -1.492  0.13587
## V42          0.0083151  0.0052357   1.588  0.11232
## V43          0.0022160  0.0026571   0.834  0.40432
## V44          0.0262624  0.0192118   1.367  0.17170
## V45         -0.0092622  0.0234190  -0.396  0.69249
## V46         -0.0247973  0.0452933  -0.547  0.58408
## V47          0.0078538  0.0032962   2.383  0.01723 *
## V48          0.0027836  0.0166754   0.167  0.86743
## V49         -0.0053178  0.0087891  -0.605  0.54518
## V50         -0.0328564  0.0523928  -0.627  0.53062
## V51          0.0882920  0.0646431   1.366  0.17206
## V52          0.0149693  0.0160836   0.931  0.35205
## V53         -0.0071507  0.0405924  -0.176  0.86018
## V54          0.0089008  0.0182050   0.489  0.62492
## V55         -0.0161979  0.0075667  -2.141  0.03236 *
## V56          0.0033504  0.0382028   0.088  0.93012
## V57          0.1455986  0.0948578   1.535  0.12488
## V58          0.0769592  0.0417431   1.844  0.06531 .
## V59          0.0122850  0.0042072   2.920  0.00352 **
## V60         -0.0251206  0.1675100  -0.150  0.88080
## V61         -0.0111426  0.0335701  -0.332  0.73997
## V62         -0.0380381  0.0615868  -0.618  0.53685
## V63         -0.0667484  0.0406090  -1.644  0.10031
## V64         -0.0310711  0.0379610  -0.819  0.41312
## V65         -0.0337581  0.0373908  -0.903  0.36666
## V66          0.0177618  0.0582802   0.305  0.76056
## V67          0.0196391  0.1598355   0.123  0.90222
## V68          0.0184974  0.0162324   1.140  0.25454
## V69         -0.0260767  0.0697336  -0.374  0.70846
## V70          0.0065174  0.0331176   0.197  0.84400
## V71          0.0901771  0.2163196   0.417  0.67679
## V72         -0.1248875  0.1074250  -1.163  0.24507
## V73         -0.0414336  0.0387917  -1.068  0.28553
## V74         -0.0014296  0.0781164  -0.018  0.98540
## V75         -0.0257911  0.0553738  -0.466  0.64141
## V76          0.0405513  0.0178340   2.274  0.02303 *
## V77         -0.0280636  0.1082137  -0.259  0.79539
## V78         -0.2642048  0.2300835  -1.148  0.25091
## V79         -0.2744572  0.2274640  -1.207  0.22765
## V80         -0.0156600  0.0135581  -1.155  0.24814
## V81         -0.0235207  0.3736589  -0.063  0.94981
## V82          0.2672272  0.1077425   2.480  0.01317 *
## V83          0.0513991  0.0461256   1.114  0.26520
## V84          0.1646178  0.0913585   1.802  0.07163 .
## V85          0.1619878  0.1288226   1.257  0.20866
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2348 on 4280 degrees of freedom
## Multiple R-squared:  0.07472,    Adjusted R-squared:  0.05634
## F-statistic: 4.066 on 85 and 4280 DF,  p-value: < 2.2e-16
```

Pass in the test data and find out the output

The R-Squared is very less. The poor performance of OLS on a classification problem was expected.

As this is a classification problem, there is a need to set some cap on the predicted values. After passing the training data to the model, I got a set of predicted values. This is how they span out. Firstly there aren't many observations with samples equal to 1. Secondly, the OSL model classifies a lot of true ones close to 0.3-0.4. Which is not a surprise as OLS is not optimizing on the classification, it is trying to reduce the quantitative error.



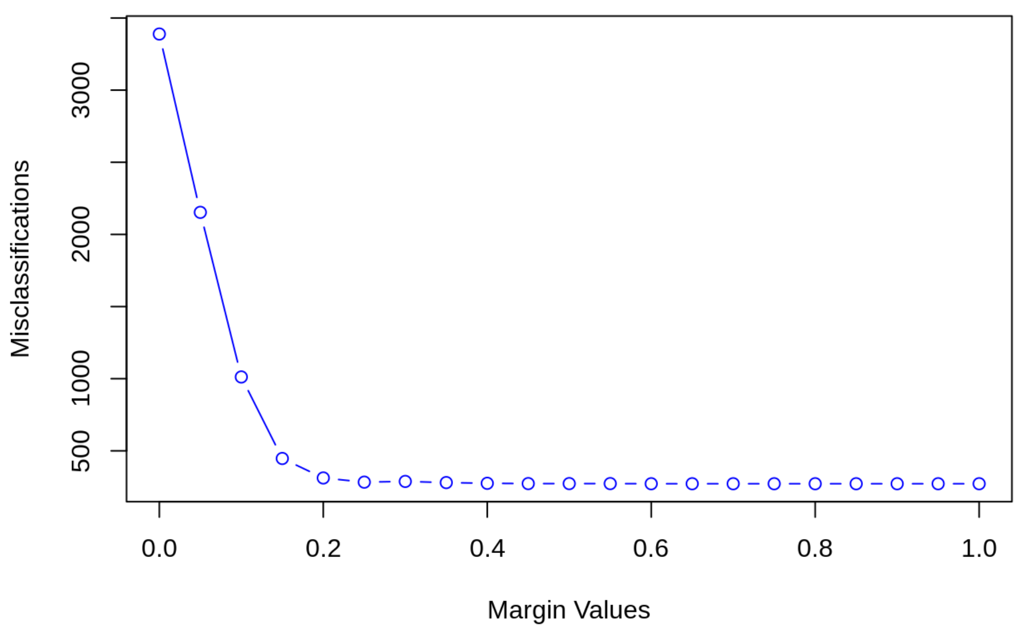
I ran a linear regression on the model to try to get the best value of the margin. Unfortunately the result wasn't good either. The model is as follows -

True value ~ Predicteed Value.

```
##   (Intercept)          pred
## -3.89813e-16  1.00000e+00
```

The margin fit tells us that the best fit is for a value close to 1. Graph above shows that there aren't many values that are above 1 or even close to 1 for that matter. I will not use the value of 1 as the margin. To select the value of margin/cap, I have selected 21 equally spaced values from 0 and 1. For each of these 21 values, I have computed the training error and the test error. The margin value, training misclassifications, and test misclassifications are shown in the columns 1, 2, and 3 respectively.

2	0.05	2153	737
3	0.1	1012	366
4	0.15	447	142
5	0.2	312	88
6	0.25	283	78
7	0.3	288	74
8	0.35	280	75
9	0.4	275	75
10	0.45	273	76
11	0.5	273	75
12	0.55	273	75
13	0.6	272	75
14	0.65	272	75
15	0.7	272	75
16	0.75	272	76
17	0.8	272	76
18	0.85	272	76
19	0.9	272	76



As can be seen from the plot above, the number of misclassifications decrease with an increase in the value of the margin upto 0.4 very quickly. After 0.4 the number of misclassifications does not change much. I have selected 0.45 as the classification margin.

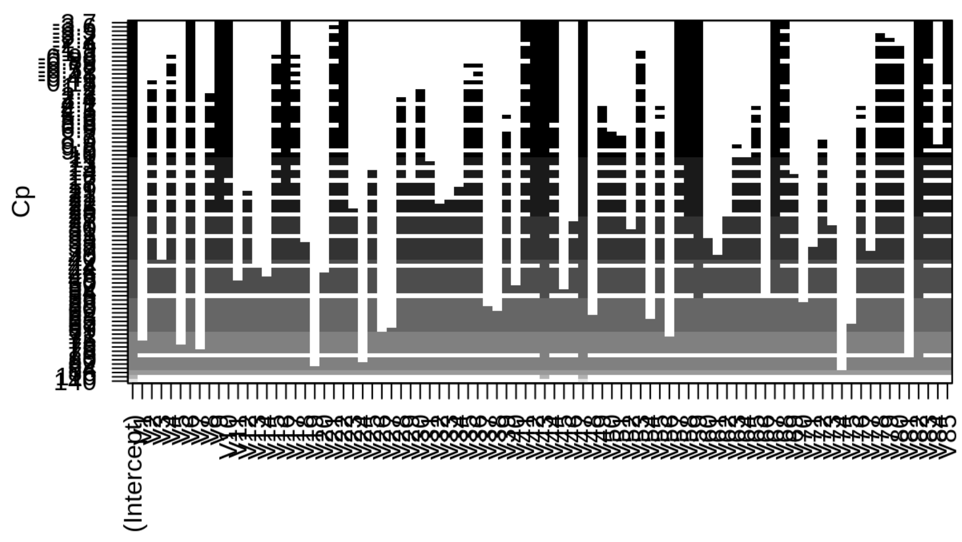
So, the number of misclassifications in the OLS solution is 273 for the training set, and 75 for the test set.

```
## [1] "Training accuracy:  0.93747136967476"
```

```
## [1] "Test accuracy:  0.947802197802198"
```

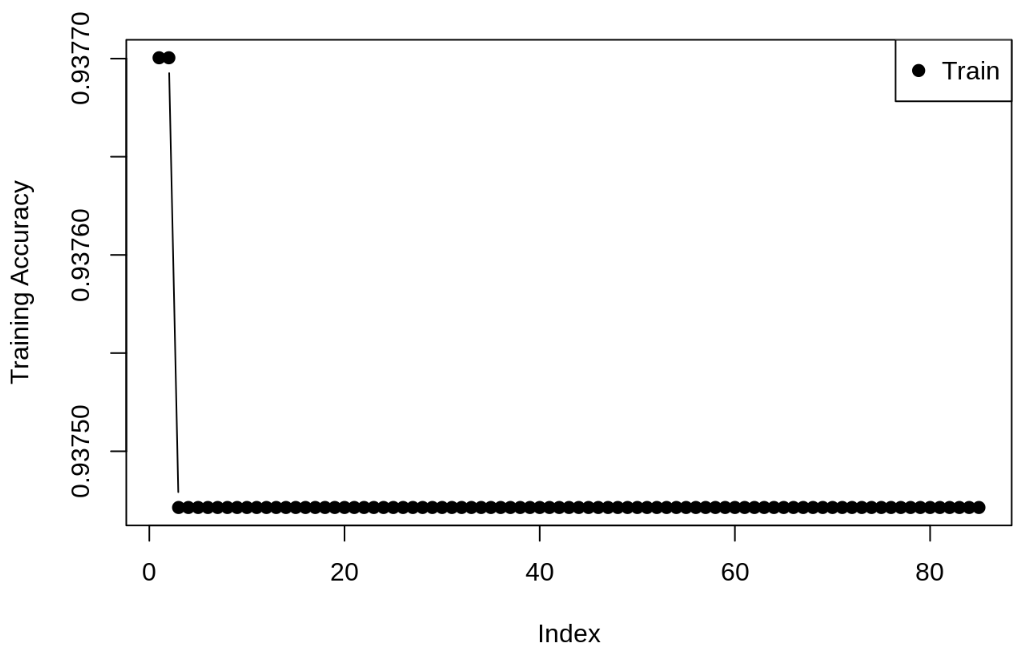
Forward Subset selection

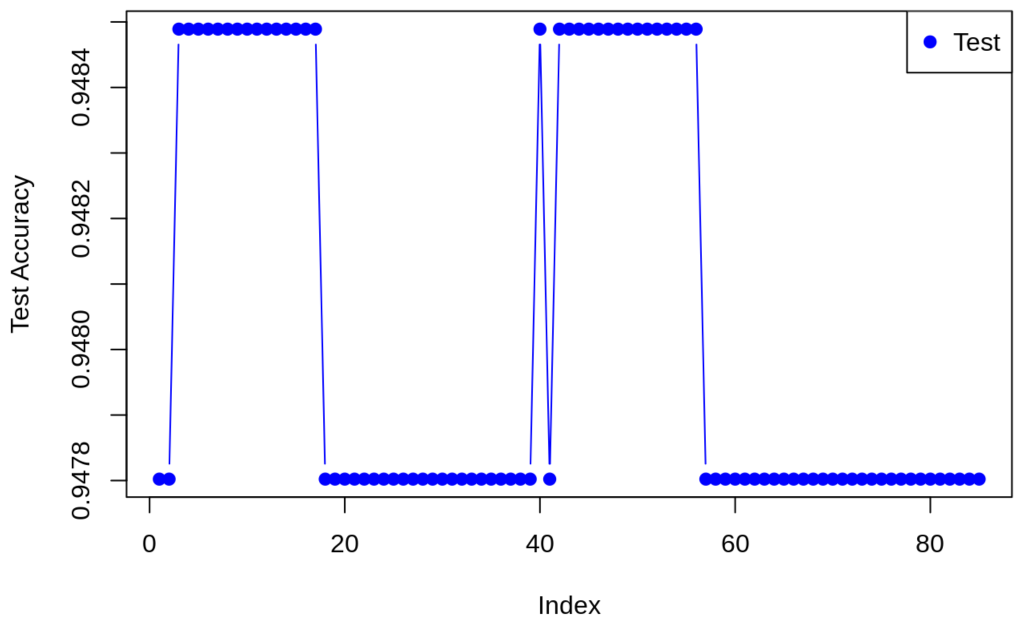
85 subsets have been generated using forward stepwise selection. The margin is assumed to 0.45. For train and test data misclassifications have been calculated.



The plot for CP indicates that

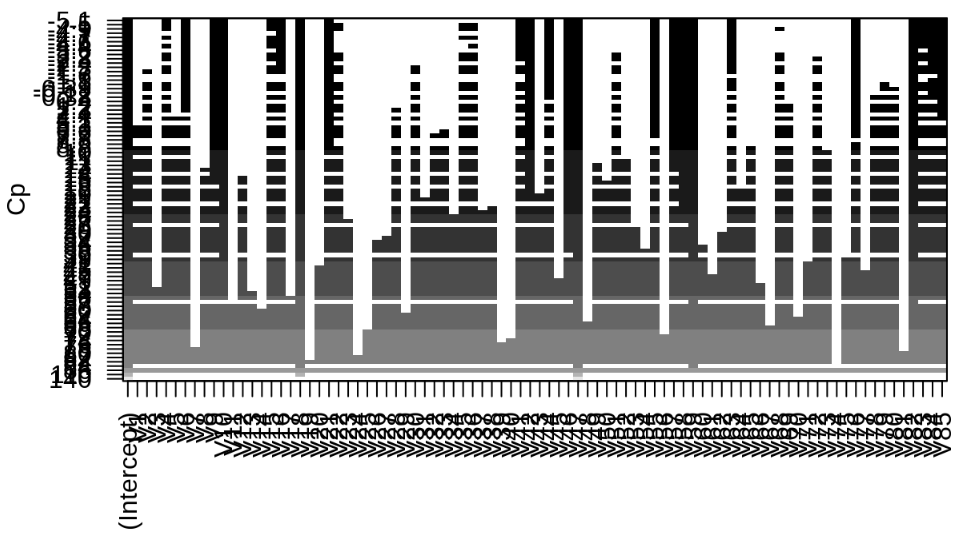
as we add more variables to the data set, the test error may increase.



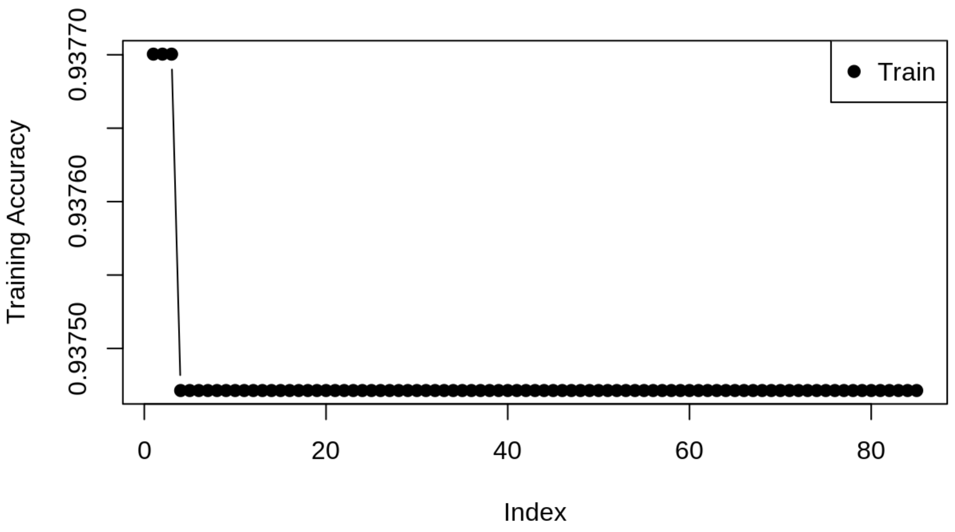


The train accuracy is less than the test accuracy. But the values do not change much accross different subsets. A lot of the subsets have minimum accuracy. I have selected a model with 10 subsets as it has a high test accuracy. There is a range of values from 2 to 18 that have a heigher accuracy than the rest. Also there is a simular such peak from 42-58. But selecting a model with lower number of predictors is always better if we are getting the same accuracy.

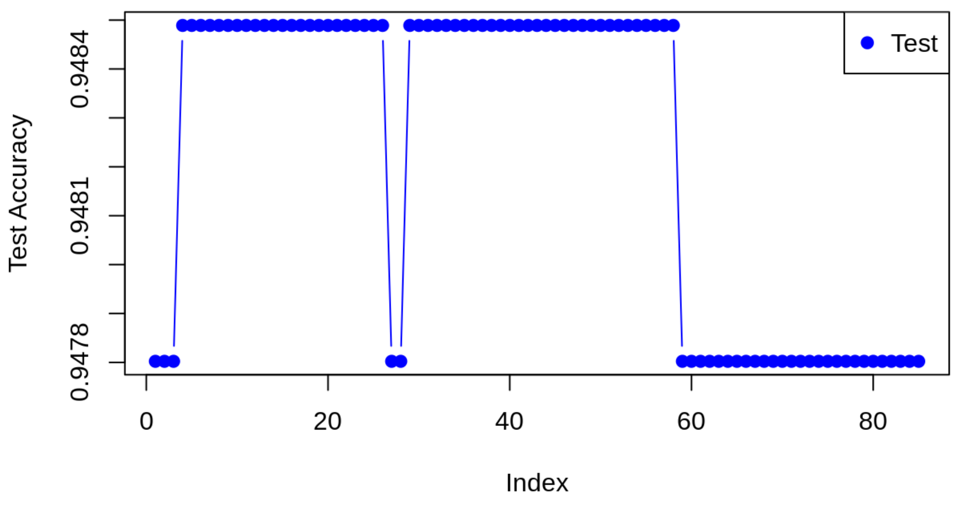
Using backward selection



Cp
(Intercept)



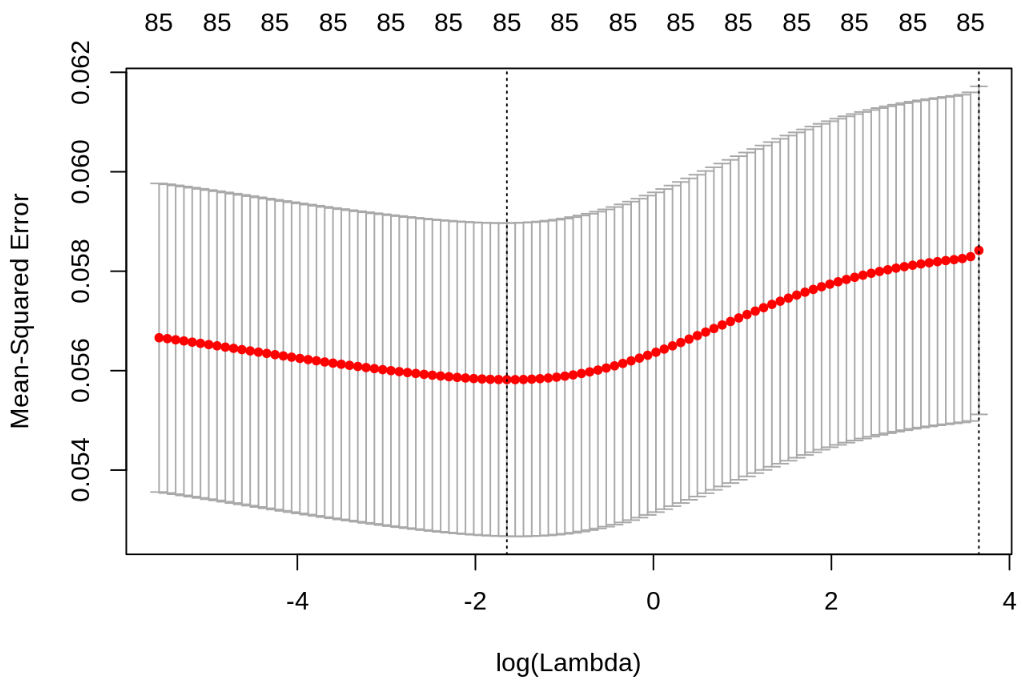
Training Accuracy
0.93750
0.93760
0.93770
Index
0
20
40
60
80
● Train



For backward selection, we will select the model with 10 features due to the same reasons mentioned for forward selection.

Ridge regression

The graph shows the penalization vs MSE obtained.



The best lambda is:

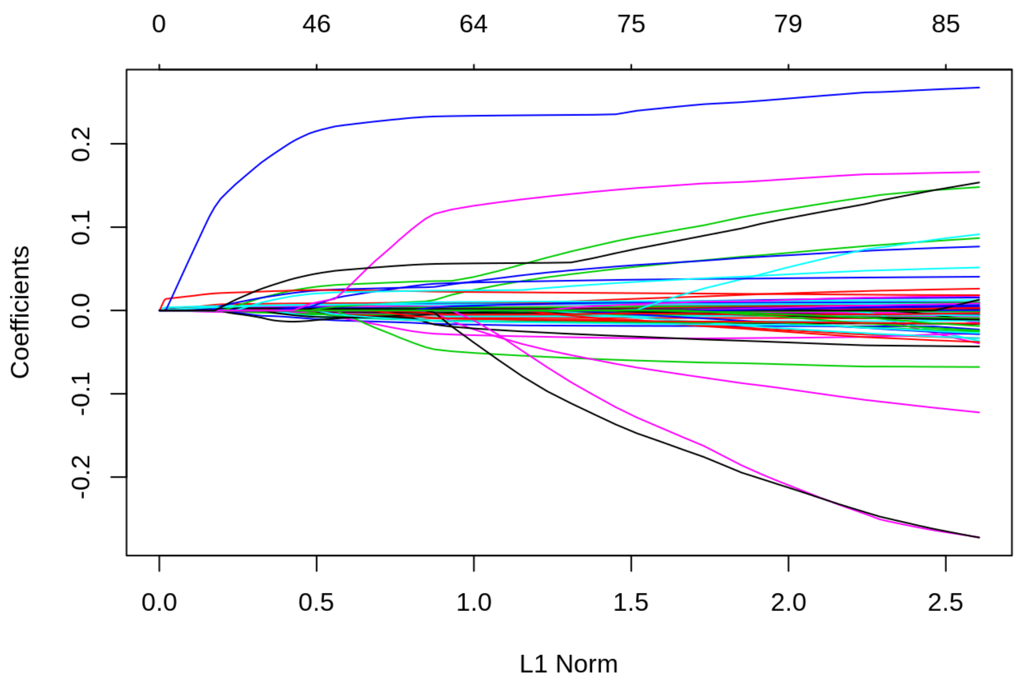
```
## [1] 0.19
```

```
## [1] 0.9377004
```

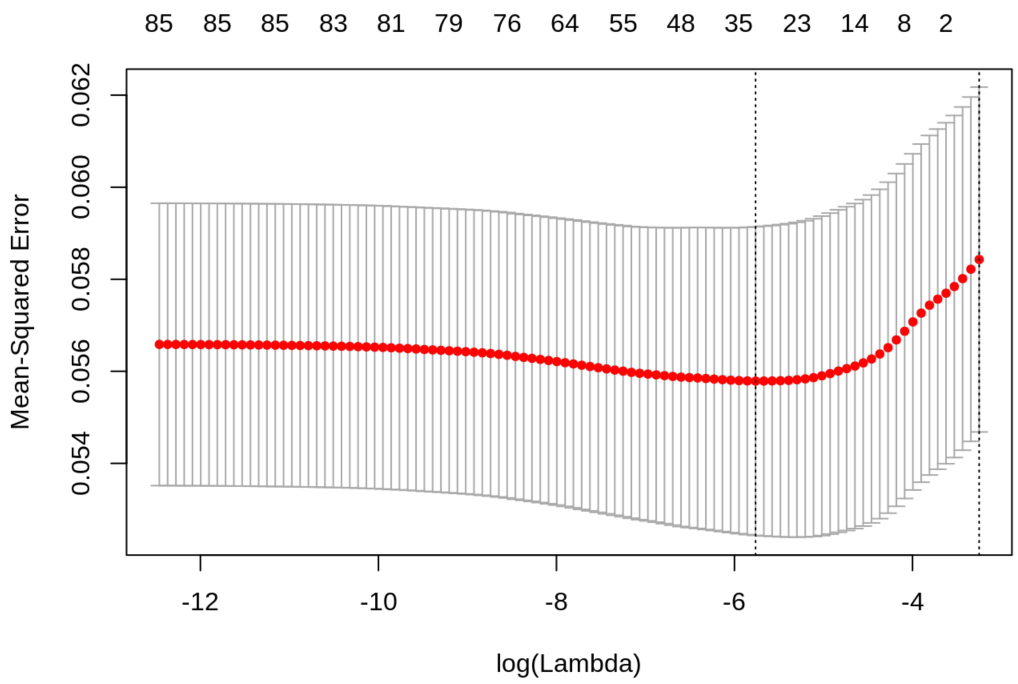
The accuracy of ridge regression on test data is:

```
## [1] 0.948489
```

The LASSO



Graph below shows how the model performs(MSE) on the training data with increase in lambda.



The best lambda for lasso is:

```
## [1] 0.003141387
```

```
## 86 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -3.331868e-02
## V1            .
## V2          -5.253181e-05
## V3            .
## V4           4.991761e-04
## V5            .
## V6          -3.165080e-03
## V7            .
## V8           1.359830e-03
## V9          -3.450580e-03
## V10           3.350334e-03
## V11          -9.416380e-04
## V12           .
## V13           .
## V14           .
## V15           1.085635e-03
## V16           4.222063e-03
## V17           .
## V18          -2.103643e-03
## V19           .
## V20           .
## V21          -3.655689e-03
## V22           1.929118e-03
## V23           .
## V24           .
## V25           .
## V26           .
## V27           .
## V28           .
## V29           .
## V30          -9.428736e-04
## V31           .
## V32           7.372119e-04
## V33           .
## V34          -1.553975e-04
## V35           .
## V36           .
## V37           .
## V38           .
## V39           .
## V40           .
## V41          -4.444920e-03
## V42           5.176550e-03
## V43           2.297444e-03
## V44           8.322689e-03
## V45           .
## V46          -7.981429e-03
## V47           6.379072e-03
## V48           .
## V49           .
## V50           .
## V51           .
## V52           .
## V53          -3.397767e-03
## V54           .
## V55           .
## V56           .
## V57           2.328407e-02
## V58           2.044832e-02
## V59           6.495851e-03
## V60           .
## V61           .
## V62           .
## V63           .
## V64           .
## V65           .
## V66           .
## V67          -1.339116e-02
## V68           2.376710e-02
## V69           .
## V70           .
## V71           .
## V72           .
## V73          -5.171417e-03
## V74           .
## V75           .
## V76           2.216473e-03
## V77           .
## V78           .
## V79           .
## V80           .
## V81           .
## V82           2.006145e-01
## V83          1.659849e-02
## V84          .
## V85          3.692245e-02
```

The train accuracy for the LASSO is:

```
## [1] 0.9374714
```

The test accuracy for the lasso is:

```
## [1] 0.9478022
```

Predicting the models on the unseen data.

Data was read from the files ticeval2000.txt and tictgts2000.txt. The data set was treated as unseen and its accuracy was calculated using all the methods used so far.

There are 4000 observations in the data set.

Ordinary Least Squares

The accuracy obtained by using the ordinary least squares model is as follows:

```
## [1] 0.94075
```

Forward selection We selected 10 for forward selection. So that will be used to compute the prediction accuracy on the new data.

The accuracy using forward selection is:

```
## [1] 0.94025
```

Backward selection We selected 10 predictors in backward selection as well. Here is the accuracy prodced by the model.

```
## [1] 0.94025
```

Ridge Ridge has the following accuracy:

```
## [1] 0.8578297
```

Lasso Lasso has the following accuracy:

```
## [1] 0.94025
```

There is not much difference between the estimates produced by each of the variables. They all roughly perform the same.

Can you predict who will be interested in buying a caravan insurance policy and give an explanation why? Even the accuracy is very high the models actually do not do very well for the positive cases - i.e. does the model predict correctly when the true value is 1(person may buy a caravan insurance policy). AS the number of 0's in the data is very large, it is difficult to correctly predict the positive case.