Lab1ML

Omkar Bhutra(omkbh878)

November 26, 2018

Assignment 1

Spam classification with nearest neighbors

```
##
## Call:
## glm(formula = Spam ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5205  -0.4402  -0.0005   0.6584   3.6196
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.508e+00  2.011e-01   7.499 6.44e-14 ***
## Word1       -7.192e-01  5.015e-01  -1.434 0.151520
## Word2        3.994e-02  3.014e-01   0.133 0.894580
## Word3       -3.529e-01  1.839e-01  -1.919 0.055019 .
## Word4        1.370e-01  1.117e-01   1.226 0.220230
## Word5        1.221e-01  1.413e-01   0.864 0.387400
## Word6        2.887e-01  4.153e-01   0.695 0.486888
## Word7       -2.948e-01  3.348e-01  -0.880 0.378676
## Word8       -1.034e-01  3.510e-01  -0.295 0.768337
## Word9       -1.085e-01  4.053e-01  -0.268 0.788983
## Word10      -3.091e-02  1.668e-01  -0.185 0.852991
## Word11      -6.088e-01  6.790e-01  -0.897 0.369902
## Word12       1.614e-01  1.073e-01   1.505 0.132378
## Word13       7.811e-01  3.572e-01   2.187 0.028771 *
## Word14      -4.819e-01  3.003e-01  -1.605 0.108598
## Word15      -1.305e-01  3.884e-01  -0.336 0.736861
## Word16       3.171e-01  2.332e-01   1.360 0.173891
## Word17      -9.201e-02  2.823e-01  -0.326 0.744479
## Word18       2.330e-02  2.322e-01   0.100 0.920074
## Word19       3.694e-04  5.746e-02   0.006 0.994871
## Word20       1.688e-02  3.181e-01   0.053 0.957687
## Word21      -2.821e-02  1.072e-01  -0.263 0.792524
## Word22      -4.767e-01  3.200e-01  -1.490 0.136337
## Word23       2.541e-01  3.413e-01   0.745 0.456525
## Word24      -2.483e-01  6.255e-01  -0.397 0.691372
## Word25      -7.828e-02  5.852e-02  -1.338 0.181027
## Word26       3.733e-03  1.358e-01   0.027 0.978071
## Word27      -2.238e-01  1.077e-01  -2.078 0.037749 *
## Word28       1.320e-01  1.880e-01   0.702 0.482776
## Word29      -8.131e-02  9.119e-02  -0.892 0.372564
## Word30      -1.815e+00  6.195e-01  -2.930 0.003391 **
## Word31      -4.694e+00  1.853e+00  -2.533 0.011296 *
## Word32      -1.194e+02  1.513e+04  -0.008 0.993703
```

```
## Word33      -2.899e+00  6.794e-01  -4.268 1.98e-05 ***
## Word34      -3.710e+00  4.352e+00  -0.852 0.394004
## Word35      -7.033e+00  1.996e+00  -3.522 0.000428 ***
## Word36      -1.678e+00  3.810e-01  -4.404 1.06e-05 ***
## Word37      -8.583e-01  2.175e-01  -3.947 7.92e-05 ***
## Word38      -6.043e-01  1.279e+00  -0.472 0.636575
## Word39      -1.877e+00  4.282e-01  -4.384 1.16e-05 ***
## Word40       7.393e-02  3.400e-01   0.217 0.827885
## Word41      -3.326e+02  1.656e+04  -0.020 0.983978
## Word42      -5.352e+00  1.302e+00  -4.109 3.98e-05 ***
## Word43      -2.592e+00  7.353e-01  -3.525 0.000423 ***
## Word44      -2.931e+00  6.601e-01  -4.441 8.96e-06 ***
## Word45      -1.141e+00  1.681e-01  -6.785 1.16e-11 ***
## Word46      -3.288e+00  5.178e-01  -6.350 2.15e-10 ***
## Word47      -3.741e+00  2.030e+00  -1.843 0.065399 .
## Word48      -4.390e+00  1.473e+00  -2.980 0.002878 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1696.82  on 1369  degrees of freedom
## Residual deviance:  928.54  on 1321  degrees of freedom
## AIC: 1026.5
##
## Number of Fisher Scoring iterations: 23
```

```
## p_class2
## Not Spam     Spam
##      888      482
```

```
##    p_class2
##     Not Spam Spam
##   0      791  146
##   1       97  336
```

```
## [1] 0.1773723
```

```
## p_class1
## Not Spam     Spam
##      884      486
```

```
##    p_class1
##     Not Spam Spam
##   0      803  142
##   1       81  344
```

```
## [1] 0.1627737
```

```
## p_class3
## Not Spam     Spam
##     1363        7
```

```
##    p_class3
##     Not Spam Spam
##   0      936    1
##   1      427    6

## [1] 0.3124088

## p_class3.1
## Not Spam     Spam
##     1363        7

##    p_class3.1
##     Not Spam Spam
##   0      944    1
##   1      419    6

## [1] 0.3065693

##
## Call:
## kknn(formula = Spam ~ ., train = train, test = test, k = 30)
##
## Response: "continuous"

## p_class4.1
## Not Spam     Spam
##      859      511

##    p_class4.1
##     Not Spam Spam
##   0      672  265
##   1      187  246

## [1] 0.329927

##
## Call:
## kknn(formula = Spam ~ ., train = train, test = train, k = 30)
##
## Response: "continuous"

## p_class4.2
## Not Spam     Spam
##      905      465

##    p_class4.2
##     Not Spam Spam
##   0      807  138
##   1       98  327

## [1] 0.1722628
```

```
##
## Call:
## kknn(formula = Spam ~ ., train = train, test = test, k = 1)
##
## Response: "continuous"

## p_class5.1
## Not Spam     Spam
##      817      553

##    p_class5.1
##     Not Spam Spam
##   0      640  297
##   1      177  256

## [1] 0.3459854

##
## Call:
## kknn(formula = Spam ~ ., train = train, test = train, k = 1)
##
## Response: "continuous"

## p_class5.2
## Not Spam     Spam
##      945      425

##    p_class5.2
##     Not Spam Spam
##   0      945    0
##   1        0  425

## [1] 0
```

1.2) For the train dataset: From the coefficients of the linear model we can say that the Intercept,word 3,13,27,30,31,33,35,36,37,39,42,43,44,45,46,47 and 48 are significant. The Confusion matrix showcases the model on the Train data set. The possible values are 'Not spam'=0, and 'Spam'=1. Out of 1370 emails, the classifier has predicted Spam mails 486 times and predicted Not Spam mails 884 times. But the actual 'Spam' and 'Not spam' mails are 425 and 945 emails respectively. There are 344 True positive values and 803 True negative values. And there are 142 False positive values and 81 false negative values. Misclassification rate which is an error rate is a measure to test the accuracy, it stands at 16.27%.

For the test dataset: The Confusion matrix showcases the model on the Train data set. The possible values are 'Not spam'=0, and 'Spam'=1. Out of 1370 emails, the classifier has predicted Spam mails 888 times and predicted Not Spam mails 482 times. But the actual 'Spam' and 'Not spam' mails are 433 and 937 emails respectively. There are 336 True positive values and 791 True negative values. And there are 146 False positive values and 97 false negative values. Misclassification rate which is an error rate is a measure to test the accuracy, it stands at 17.7%.

1.3) With the change in the probability from 0.5 to 0.9 in the classification criterion, the misclassification rate has increased from 16.27% to 31.67% for train data set and 17.7% to 30.6% for the test dataset and the number of spam mails correctly predicted has reduced.

1.4) The misclassification rate for train data set is 17.22% and for test data set it is 32.99% whereas in step 2 it was 16.27% and 17.73% for train and test data set respectively.

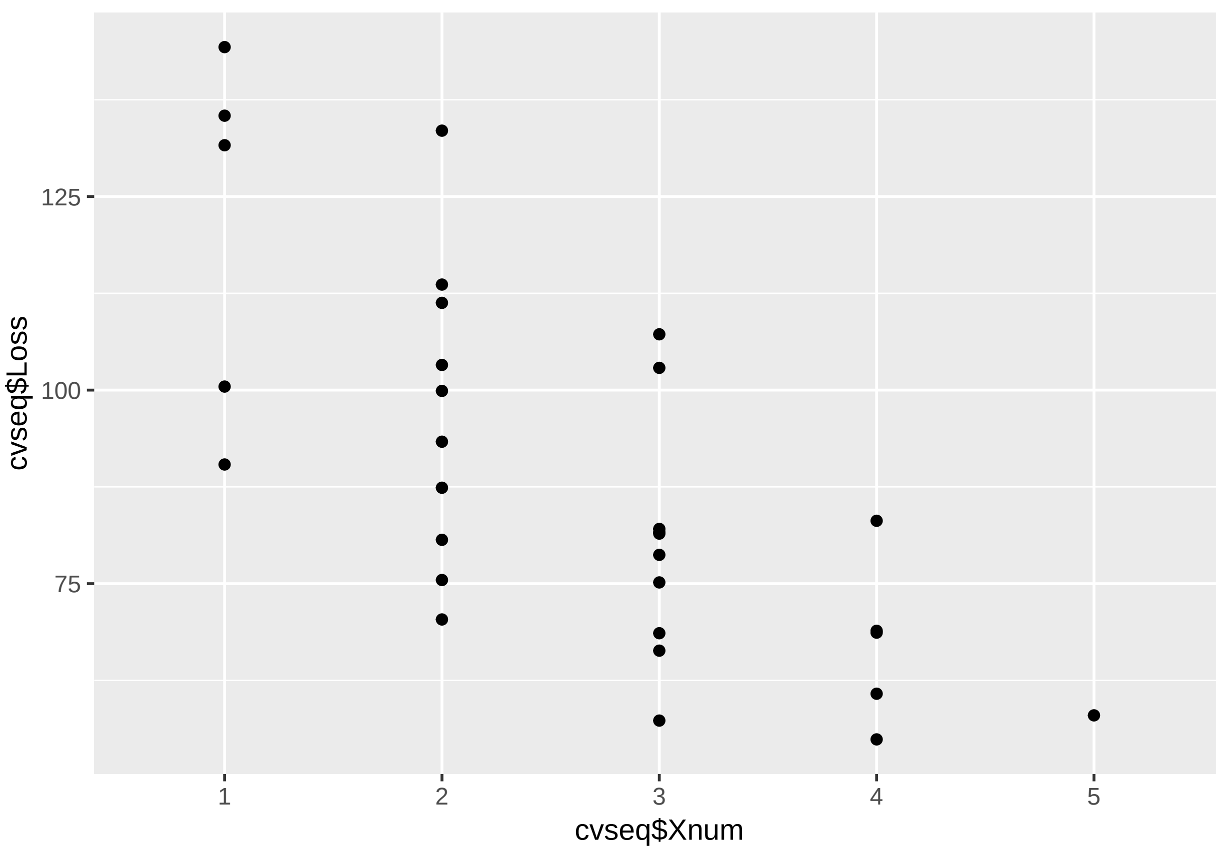
1.5) The misclassification rate using KNN with the training dataset and test dataset is 0% and 34.59% respectively. In step 2, the misclassification rate stood at 16.2% for the training data and 17.7% for the test data. With decrease in the K value , the misclassification rate increases. With the 1-nearest neighbor model, each observation of the training dataset is potentially the center of an area predicting the outcome, with the other neighbors potentially predicting the other possible outcomes and hence making it highly complex. With k=30, the area's predicting each outcome will be more smooth as its the majority of the 30 nearest neighbors that predict the outcome of any observation. The area's are hence in lesser number, larger sizes and less complex. The variance term is simply the variance of an average here, and decreases as the inverse of k. So as k varies, there is a bias-variance tradeoff. More generally, as the model complexity of our procedure is increased, the variance tends to increase and the squared bias tends to decrease. The opposite behavior occurs as the model complexity is decreased. For k-nearest neighbors, the model complexity is controlled by k.

Assignment 3

Feature selection by cross-validation in a linear model

```
## Optimal subset of features:  1,3,4,5
```

```
## Cross validation score:  54.88725
```

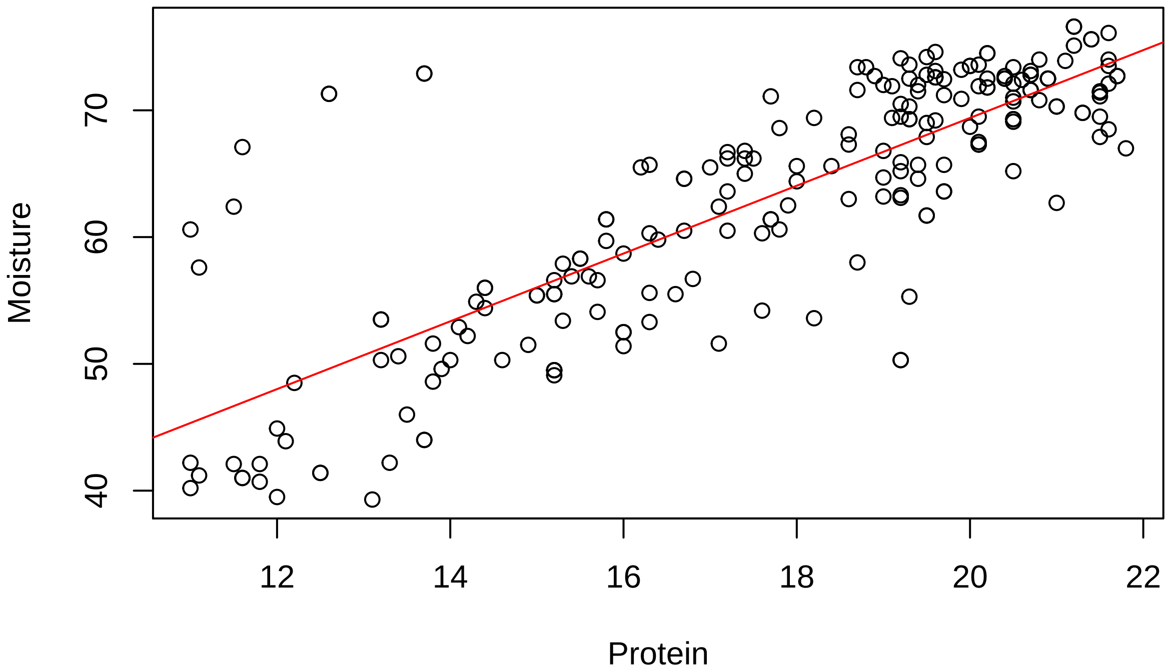


The optimal subset of X variables is Agriculture, Education, Catholic and Infant Mortality. The minimum cv score is 54.88 using a 4 variable subset. It is quite evident that Examination results do not affect the fertility measures. It is reasonable that these subset variable's will have a large impact on the fertility measure.

Assignment 4

Linear regression and regularization

Q1




```
## (Intercept)     Protein
##   15.924560    2.673778
```

```
## [1] "y = 2.7*x +15.9"
```

```
##
## Call:
## lm(formula = Moisture ~ Protein, data = tecator)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.9611  -3.1660   0.0255   2.5836  21.6858
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.9246     2.3405   6.804 1.01e-10 ***
## Protein       2.6738     0.1305  20.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 5.758 on 213 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6619
## F-statistic: 419.9 on 1 and 213 DF,  p-value: < 2.2e-16
```

```
## [1] 32.85021
```

Yes , the data is well described by a linear model as Protein is directly propotional to Moisture values, with only a few outliers with very high values of Moisture.

Q2

M_i,i is 1,2,3,..,i y^ is the expected moisture x is protein M_i : p(y|x,w) is $y_{N(aplha}0_{+aplha}1_{.x}1_{+aplha}2_{.x}2_{+\ldots+aplha}i_{.x}i$~,std.dev^2)

It is appropriate to use the Mean Squared Error criterion when fitting this model to the training data as the model is fitted using the Least Sqaures Method, where the fitted line is chosen such that the vertical distances from the data are the least. MSE consists of two components, the squared bias and variance. MSE is also used due to the curse of dimensionality, when there is no noise in the target function , the MSE approximates to squared bias. When the target function is constant in all but one dimension, The variance dominates and is approximated by MSE.

There is a big drawback in the simple error metric. This is because the positive and the negative errors cancel out. This can happen in the real scenario too, where the errors across all samples of observed data can cancel out each other. To get around this problem,Mean Squared Error. Now, no matter what the sign of error is, the squaring operation always amplifies the errors in the positive direction.

Q3

```
## (Intercept)     Protein
##   17.685838    2.582355
```

```
## [1] "y = 2.7*x +15.9"
```

```
##
## Call:
## lm(formula = Moisture ~ Protein, data = train, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.9671  -2.9550   0.0494   2.7753  21.0765
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.686      3.402   5.199 9.95e-07 ***
## Protein        2.582      0.188  13.739  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.853 on 105 degrees of freedom
## Multiple R-squared:  0.6426, Adjusted R-squared:  0.6391
## F-statistic: 188.7 on 1 and 105 DF,  p-value: < 2.2e-16
```

```
## [1] 33.61836
```

```
## (Intercept)     Protein    Protein2
##  49.6471841  -1.3920468   0.1189638
```

```
## [1] "y = -1.4*x +0.1*x^2 +49.6"
```

```
##
## Call:
## lm(formula = Moisture ~ Protein + Protein2, data = train, x = TRUE,
##     y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.4747  -2.7866   0.1618   2.9023  20.3059
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.64718   18.93534   2.622   0.0101 *
## Protein     -1.39205    2.32448  -0.599   0.5506
## Protein2     0.11896    0.06935   1.715   0.0893 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.8 on 104 degrees of freedom
## Multiple R-squared:  0.6524, Adjusted R-squared:  0.6457
## F-statistic: 97.59 on 2 and 104 DF,  p-value: < 2.2e-16
```

```
## [1] 32.69342
```

```
##  (Intercept)      Protein     Protein2     Protein3
## 226.91446170 -35.02882584   2.19476262  -0.04178493
```

```
## [1] "y = -35*x +2.2*x^2 +0*x^2 +226.9"
```

```
##
## Call:
## lm(formula = Moisture ~ Protein + Protein2 + Protein3, data = train,
##     x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.3893  -2.9224   0.4621   3.0396  21.3893
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 226.91446   97.20177   2.334   0.0215 *
## Protein     -35.02883   18.24432  -1.920   0.0576 .
## Protein2      2.19476    1.11904   1.961   0.0525 .
## Protein3     -0.04178    0.02248  -1.858   0.0660 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.732 on 103 degrees of freedom
## Multiple R-squared:  0.6637, Adjusted R-squared:  0.6539
## F-statistic: 67.75 on 3 and 103 DF,  p-value: < 2.2e-16
```

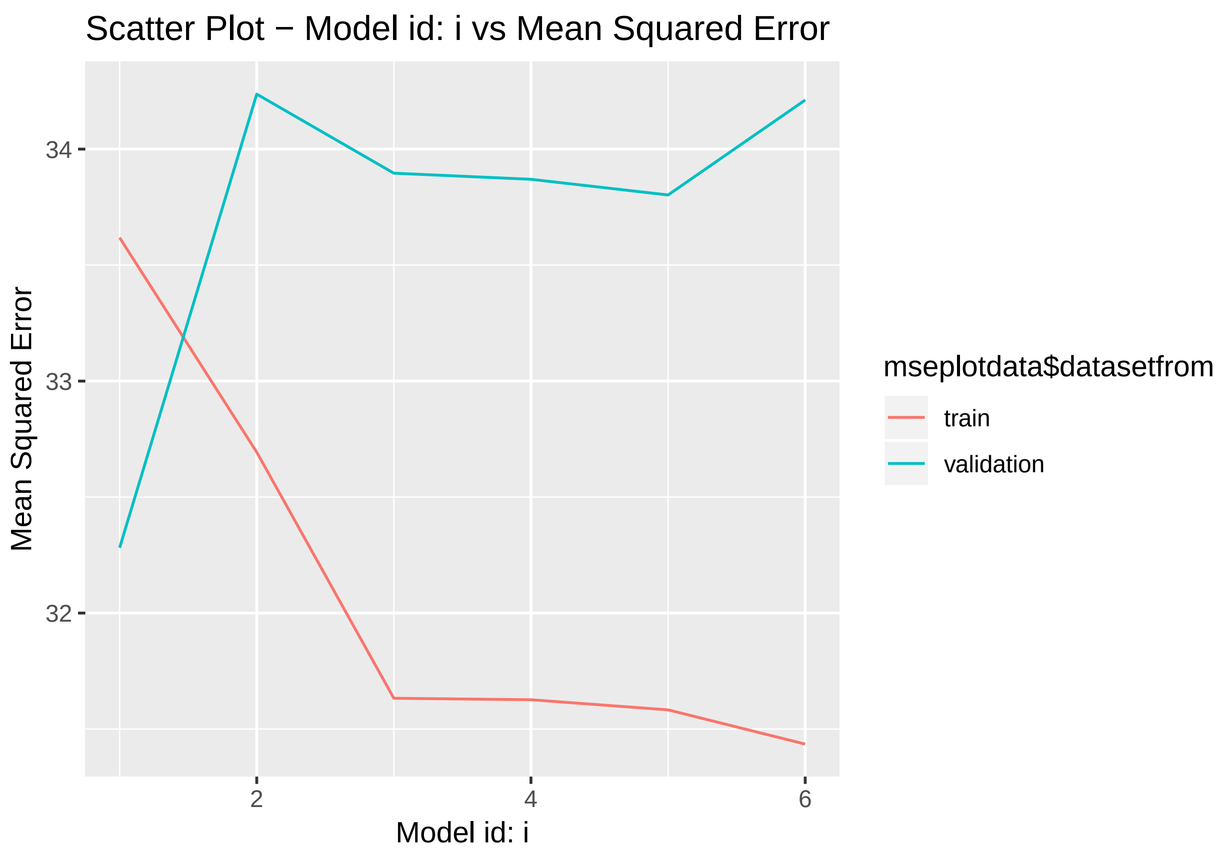
```
## [1] 31.63266

##  (Intercept)      Protein     Protein2     Protein3     Protein4
## 147.20755297 -14.71009881   0.28880722   0.03626647  -0.00117897

##
## Call:
## lm(formula = Moisture ~ Protein + Protein2 + Protein3 + Protein4,
##     data = train, x = TRUE, y = TRUE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.413  -2.961   0.442   3.086  21.293
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 147.207553 569.806627   0.258    0.797
## Protein     -14.710099 144.273696  -0.102    0.919
## Protein2      0.288807  13.470609   0.021    0.983
## Protein3      0.036266   0.550178   0.066    0.948
## Protein4     -0.001179   0.008303  -0.142    0.887
##
## Residual standard error: 5.76 on 102 degrees of freedom
## Multiple R-squared:  0.6637, Adjusted R-squared:  0.6505
## F-statistic: 50.33 on 4 and 102 DF,  p-value: < 2.2e-16

## [1] 31.62641

##   (Intercept)       Protein      Protein2      Protein3      Protein4
##  1.504946e+03 -4.491577e+02  5.509783e+01 -3.373110e+00  1.034524e-01
##      Protein5
## -1.268213e-03

##
## Call:
## lm(formula = Moisture ~ Protein + Protein2 + Protein3 + Protein4 +
##     Protein5, data = train, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.2813  -3.0122   0.4175   2.8961  21.2493
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.505e+03  3.678e+03   0.409    0.683
## Protein     -4.492e+02  1.171e+03  -0.383    0.702
## Protein2     5.510e+01  1.473e+02   0.374    0.709
## Protein3    -3.373e+00  9.139e+00  -0.369    0.713
## Protein4     1.035e-01  2.801e-01   0.369    0.713
## Protein5    -1.268e-03  3.393e-03  -0.374    0.709
##
## Residual standard error: 5.784 on 101 degrees of freedom
## Multiple R-squared:  0.6642, Adjusted R-squared:  0.6476
## F-statistic: 39.95 on 5 and 101 DF,  p-value: < 2.2e-16
```

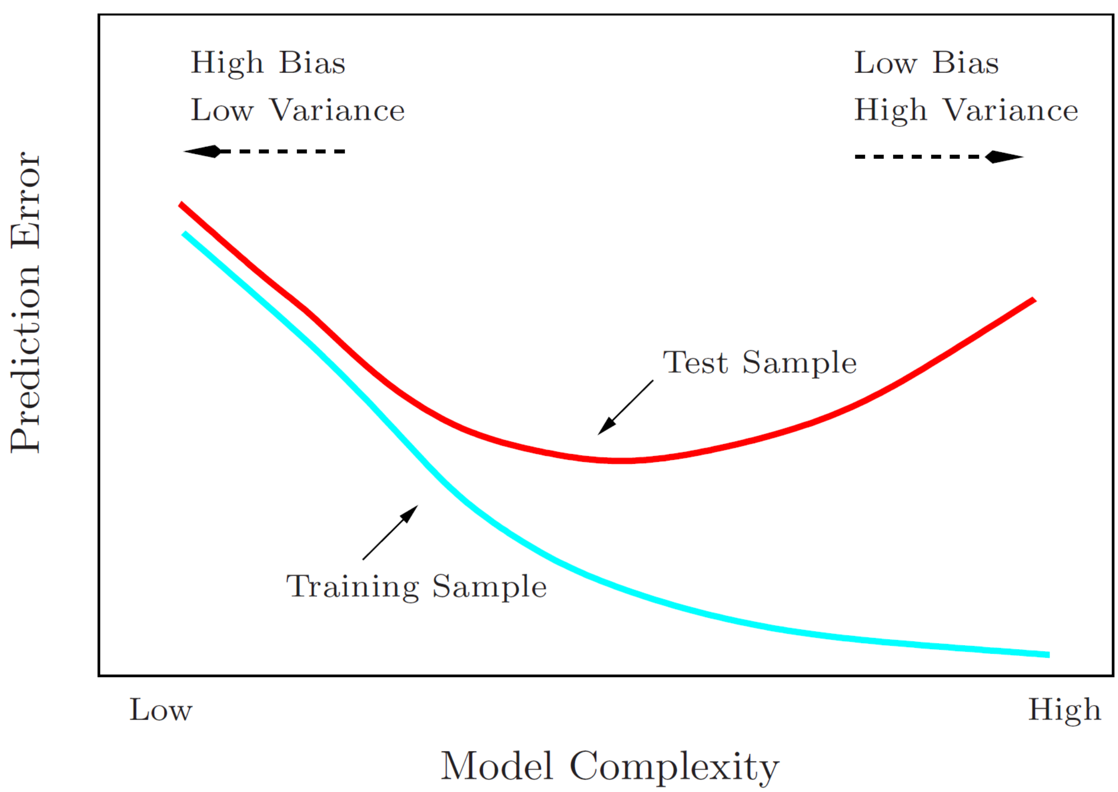
```
## [1] 31.58273
```

```
##   (Intercept)       Protein      Protein2      Protein3      Protein4 
##  1.478818e+04 -5.554997e+03  8.636619e+02 -7.091387e+01  3.243092e+00 
##      Protein5      Protein6 
## -7.830290e-02  7.797287e-04
```

```
## 
## Call:
## lm(formula = Moisture ~ Protein + Protein2 + Protein3 + Protein4 + 
##     Protein5 + Protein6, data = train, x = TRUE, y = TRUE)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -17.6110  -2.9827   0.5657   2.9141  21.5539 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.479e+04  1.973e+04   0.749    0.455
## Protein     -5.555e+03  7.543e+03  -0.736    0.463
## Protein2     8.637e+02  1.189e+03   0.726    0.469
## Protein3    -7.091e+01  9.899e+01  -0.716    0.475
## Protein4     3.243e+00  4.590e+00   0.706    0.482
## Protein5    -7.830e-02  1.125e-01  -0.696    0.488
## Protein6     7.797e-04  1.138e-03   0.685    0.495
## 
## Residual standard error: 5.8 on 100 degrees of freedom
## Multiple R-squared:  0.6658, Adjusted R-squared:  0.6457 
## F-statistic:  33.2 on 6 and 100 DF,  p-value: < 2.2e-16
```

```
## [1] 31.43513
```


Scatter Plot − Model id: i vs Mean Squared Error
Mean Squared Error
34
33
32
2
4
6
Model id: i
mseplotdata$datasetfrom
train
validation


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

The model with 1st order polynomial (Linear model) is the best model as there is the right amount of bias variance tradeoff. The training error decreases at first and then stabilises beyond 3rd order polynomial while the validation error increases at first and stabilises beyond 2nd order polynomial function. Bias-Variance tradeoff: As the more generalised view of the prediction error vs the model complexity is shown in the picture attached. In the Mean Squared Error, the variance is the average of the variances and decreases as the inverse of k. So as the k varies, there is a bias variance tradeoff. For the training sample, As the Model complexity increases, the prediction error decreases. For the testing sample, as the model complexity increases the the prediction error decreases at first but reaches a minima and starts to increase as the model complexity goes higher. Typically we would like to choose our model complexity to trade biasoff with variance in such a way as to minimize the test error. An obvious estimate of test error is the training error, 1/N * sum(i(yi ??? $^{yi)}$2), But the training error is not a good estimate of the testing error.

Q4

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## Fat ~ (Channel1 + Channel2 + Channel3 + Channel4 + Channel5 +
##     Channel6 + Channel7 + Channel8 + Channel9 + Channel10 + Channel11 +
##     Channel12 + Channel13 + Channel14 + Channel15 + Channel16 +
##     Channel17 + Channel18 + Channel19 + Channel20 + Channel21 +
##     Channel22 + Channel23 + Channel24 + Channel25 + Channel26 +
##     Channel27 + Channel28 + Channel29 + Channel30 + Channel31 +
##     Channel32 + Channel33 + Channel34 + Channel35 + Channel36 +
##     Channel37 + Channel38 + Channel39 + Channel40 + Channel41 +
##     Channel42 + Channel43 + Channel44 + Channel45 + Channel46 +
##     Channel47 + Channel48 + Channel49 + Channel50 + Channel51 +
##     Channel52 + Channel53 + Channel54 + Channel55 + Channel56 +
##     Channel57 + Channel58 + Channel59 + Channel60 + Channel61 +
##     Channel62 + Channel63 + Channel64 + Channel65 + Channel66 +
##     Channel67 + Channel68 + Channel69 + Channel70 + Channel71 +
##     Channel72 + Channel73 + Channel74 + Channel75 + Channel76 +
##     Channel77 + Channel78 + Channel79 + Channel80 + Channel81 +
##     Channel82 + Channel83 + Channel84 + Channel85 + Channel86 +
##     Channel87 + Channel88 + Channel89 + Channel90 + Channel91 +
##     Channel92 + Channel93 + Channel94 + Channel95 + Channel96 +
##     Channel97 + Channel98 + Channel99 + Channel100 + Protein +
##     Moisture) - Moisture - Protein
##
## Final Model:
## Fat ~ Channel1 + Channel2 + Channel4 + Channel5 + Channel7 +
##     Channel8 + Channel11 + Channel12 + Channel13 + Channel14 +
##     Channel15 + Channel17 + Channel19 + Channel20 + Channel22 +
##     Channel24 + Channel25 + Channel26 + Channel28 + Channel29 +
##     Channel30 + Channel32 + Channel34 + Channel36 + Channel37 +
##     Channel39 + Channel40 + Channel41 + Channel42 + Channel45 +
##     Channel46 + Channel47 + Channel48 + Channel50 + Channel51 +
##     Channel52 + Channel54 + Channel55 + Channel56 + Channel59 +
##     Channel60 + Channel61 + Channel63 + Channel64 + Channel65 +
##     Channel67 + Channel68 + Channel69 + Channel71 + Channel73 +
##     Channel74 + Channel78 + Channel79 + Channel80 + Channel81 +
##     Channel84 + Channel85 + Channel87 + Channel88 + Channel92 +
```

```
##     Channel94 + Channel98 + Channel99
##
##
##           Step Df     Deviance Resid. Df Resid. Dev       AIC
## 1                                   114   169.8123 151.27203
## 2   - Channel70  1 5.580758e-05       115   169.8124 149.27210
## 3   - Channel89  1 6.338934e-04       116   169.8130 147.27290
## 4   - Channel66  1 4.350148e-04       117   169.8135 145.27345
## 5  - Channel100  1 9.526559e-04       118   169.8144 143.27466
## 6   - Channel57  1 1.512331e-03       119   169.8159 141.27657
## 7   - Channel38  1 4.235150e-03       120   169.8202 139.28193
## 8   - Channel58  1 7.141818e-03       121   169.8273 137.29098
## 9   - Channel53  1 2.509829e-02       122   169.8524 135.32275
## 10   - Channel9  1 3.771904e-02       123   169.8901 133.37049
## 11  - Channel91  1 3.178511e-02       124   169.9219 131.41071
## 12  - Channel77  1 5.501288e-02       125   169.9769 129.48030
## 13  - Channel49  1 9.282875e-02       126   170.0698 127.59769
## 14  - Channel33  1 1.137405e-01       127   170.1835 125.74143
## 15  - Channel96  1 1.838591e-01       128   170.3674 123.97358
## 16  - Channel93  1 1.204802e-01       129   170.4878 122.12557
## 17  - Channel82  1 2.012906e-01       130   170.6891 120.37927
## 18  - Channel86  1 2.608049e-01       131   170.9499 118.70753
## 19  - Channel72  1 3.340581e-01       132   171.2840 117.12725
## 20  - Channel35  1 4.539629e-01       133   171.7380 115.69633
## 21  - Channel43  1 3.667681e-01       134   172.1047 114.15500
## 22  - Channel44  1 3.686336e-01       135   172.4734 112.61502
## 23  - Channel90  1 4.430432e-01       136   172.9164 111.16659
## 24  - Channel83  1 4.636039e-01       137   173.3800 109.74225
## 25   - Channel3  1 4.495464e-01       138   173.8295 108.29899
## 26  - Channel23  1 4.393963e-01       139   174.2689 106.84177
## 27   - Channel6  1 6.745513e-01       140   174.9435 105.67238
## 28  - Channel62  1 6.873639e-01       141   175.6309 104.51547
## 29  - Channel10  1 6.770690e-01       142   176.3079 103.34272
## 30  - Channel18  1 5.551316e-01       143   176.8631 102.01861
## 31  - Channel27  1 8.012085e-01       144   177.6643 100.99038
## 32  - Channel16  1 8.124404e-01       145   178.4767  99.97132
## 33  - Channel21  1 9.726859e-01       146   179.4494  99.13987
## 34  - Channel95  1 8.809590e-01       147   180.3304  98.19277
## 35  - Channel97  1 6.630855e-01       148   180.9934  96.98189
## 36  - Channel76  1 1.451145e+00       149   182.4446  96.69882
## 37  - Channel75  1 7.506552e-01       150   183.1952  95.58160
## 38  - Channel31  1 1.682931e+00       151   184.8782  95.54769
```

```
##
## Call:
## lm(formula = Fat ~ Channel1 + Channel2 + Channel4 + Channel5 +
##     Channel7 + Channel8 + Channel11 + Channel12 + Channel13 +
##     Channel14 + Channel15 + Channel17 + Channel19 + Channel20 +
##     Channel22 + Channel24 + Channel25 + Channel26 + Channel28 +
##     Channel29 + Channel30 + Channel32 + Channel34 + Channel36 +
##     Channel37 + Channel39 + Channel40 + Channel41 + Channel42 +
##     Channel45 + Channel46 + Channel47 + Channel48 + Channel50 +
##     Channel51 + Channel52 + Channel54 + Channel55 + Channel56 +
##     Channel59 + Channel60 + Channel61 + Channel63 + Channel64 +
```

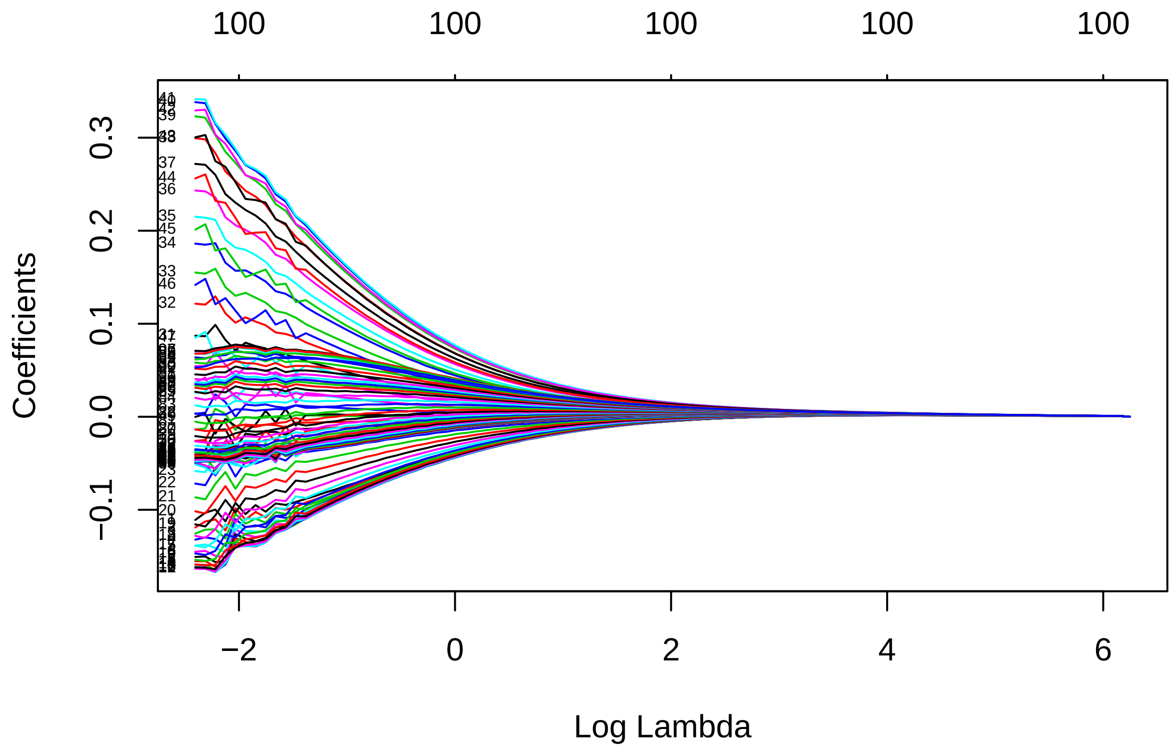
```
##     Channel65 + Channel67 + Channel68 + Channel69 + Channel71 +
##     Channel73 + Channel74 + Channel78 + Channel79 + Channel80 +
##     Channel81 + Channel84 + Channel85 + Channel87 + Channel88 +
##     Channel92 + Channel94 + Channel98 + Channel99, data = data.frame(tecator1))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.82961 -0.57129 -0.00696  0.58152  2.86375
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.093      1.453   4.882 2.64e-06 ***
## Channel1     10559.894   2333.430   4.525 1.21e-05 ***
## Channel2    -12636.967   3467.995  -3.644 0.000369 ***
## Channel4      8489.323   4637.993   1.830 0.069164 .
## Channel5    -10408.967   4771.350  -2.182 0.030689 *
## Channel7     -5376.018   3851.782  -1.396 0.164847
## Channel8      7215.595   4246.489   1.699 0.091342 .
## Channel11    -9505.520   5721.115  -1.661 0.098692 .
## Channel12    37240.918  12290.648   3.030 0.002878 **
## Channel13   -41564.547  15892.375  -2.615 0.009817 **
## Channel14    34938.179  13290.454   2.629 0.009454 **
## Channel15   -23761.451   6584.006  -3.609 0.000417 ***
## Channel17     4296.572   3189.730   1.347 0.179998
## Channel19    14279.808   5017.407   2.846 0.005042 **
## Channel20   -23855.616   5153.161  -4.629 7.85e-06 ***
## Channel22    18444.906   3381.683   5.454 1.97e-07 ***
## Channel24   -20138.426   4946.417  -4.071 7.52e-05 ***
## Channel25    18137.432   5374.094   3.375 0.000938 ***
## Channel26    -7670.318   3859.006  -1.988 0.048660 *
## Channel28    20079.898   4991.631   4.023 9.06e-05 ***
## Channel29   -36351.014   7655.223  -4.749 4.72e-06 ***
## Channel30    18071.276   5863.802   3.082 0.002446 **
## Channel32     3838.013   2722.862   1.410 0.160729
## Channel34    -9242.884   2225.926  -4.152 5.48e-05 ***
## Channel36     8070.938   3317.588   2.433 0.016152 *
## Channel37    -9045.588   3536.621  -2.558 0.011522 *
## Channel39    18664.454   5986.730   3.118 0.002183 **
## Channel40   -20069.709  10701.902  -1.875 0.062677 .
## Channel41    22257.776  11122.533   2.001 0.047169 *
## Channel42   -21760.853   5833.811  -3.730 0.000270 ***
## Channel45    18145.804   2985.416   6.078 9.50e-09 ***
## Channel46    -8225.696   3715.367  -2.214 0.028330 *
## Channel47    -4986.549   2558.694  -1.949 0.053165 .
## Channel48     2876.075   2014.985   1.427 0.155546
## Channel50   -13009.410   4535.797  -2.868 0.004720 **
## Channel51    29251.161   6554.297   4.463 1.57e-05 ***
## Channel52   -26833.976   4389.473  -6.113 7.97e-09 ***
## Channel54    30954.862   4392.339   7.047 6.06e-11 ***
## Channel55   -35183.287   5646.314  -6.231 4.39e-09 ***
## Channel56    14912.986   2810.889   5.305 3.93e-07 ***
## Channel59    -8030.278   1887.431  -4.255 3.66e-05 ***
## Channel60    13071.416   2629.374   4.971 1.79e-06 ***
## Channel61    -7850.189   2246.864  -3.494 0.000625 ***
```

```
## Channel63    15059.275   3231.692   4.660 6.90e-06 ***
## Channel64   -19909.466   4727.696  -4.211 4.35e-05 ***
## Channel65     4190.184   3486.766   1.202 0.231346
## Channel67    13850.508   3909.121   3.543 0.000526 ***
## Channel68   -25873.365   5304.223  -4.878 2.69e-06 ***
## Channel69    18362.385   3331.483   5.512 1.50e-07 ***
## Channel71    -9223.910   1558.752  -5.917 2.11e-08 ***
## Channel73    12456.498   2386.255   5.220 5.82e-07 ***
## Channel74    -5624.411   1933.590  -2.909 0.004177 **
## Channel78    -7927.105   2176.860  -3.642 0.000372 ***
## Channel79    15473.188   3812.200   4.059 7.89e-05 ***
## Channel80   -22391.895   4490.714  -4.986 1.67e-06 ***
## Channel81    13852.453   3105.934   4.460 1.59e-05 ***
## Channel84   -11442.630   3457.064  -3.310 0.001167 **
## Channel85    20228.671   4081.863   4.956 1.91e-06 ***
## Channel87   -15938.315   4102.273  -3.885 0.000153 ***
## Channel88     5647.072   3236.286   1.745 0.083033 .
## Channel92     6595.995   1864.595   3.537 0.000537 ***
## Channel94    -5497.846   1847.113  -2.976 0.003397 **
## Channel98    -8728.596   2489.314  -3.506 0.000598 ***
## Channel99     8554.587   1898.010   4.507 1.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.107 on 151 degrees of freedom
## Multiple R-squared:  0.9947, Adjusted R-squared:  0.9925
## F-statistic: 447.9 on 63 and 151 DF,  p-value: < 2.2e-16

##   (Intercept)      Channel1      Channel2      Channel4      Channel5
##      7.093133  10559.893784 -12636.966607   8489.323117 -10408.966948
##      Channel7      Channel8     Channel11     Channel12     Channel13
##  -5376.017738   7215.595409  -9505.520235  37240.918374 -41564.546571
##     Channel14     Channel15     Channel17     Channel19     Channel20
##  34938.179314 -23761.450875   4296.572462  14279.808102 -23855.616123
##     Channel22     Channel24     Channel25     Channel26     Channel28
##  18444.905722 -20138.426065  18137.431996  -7670.318234  20079.898191
##     Channel29     Channel30     Channel32     Channel34     Channel36
## -36351.013717  18071.275531   3838.013358  -9242.884498   8070.938452
##     Channel37     Channel39     Channel40     Channel41     Channel42
##  -9045.587624  18664.454171 -20069.708579  22257.776227 -21760.853228
##     Channel45     Channel46     Channel47     Channel48     Channel50
##  18145.803786  -8225.696060  -4986.549169   2876.074542 -13009.409717
##     Channel51     Channel52     Channel54     Channel55     Channel56
##  29251.160946 -26833.976402  30954.861519 -35183.287363  14912.986496
##     Channel59     Channel60     Channel61     Channel63     Channel64
##  -8030.277501  13071.415506  -7850.189324  15059.274961 -19909.466348
##     Channel65     Channel67     Channel68     Channel69     Channel71
##   4190.183533  13850.508143 -25873.365427  18362.384676  -9223.909939
##     Channel73     Channel74     Channel78     Channel79     Channel80
##  12456.497755  -5624.411385  -7927.104791  15473.187794 -22391.894812
##     Channel81     Channel84     Channel85     Channel87     Channel88
##  13852.452651 -11442.629734  20228.671387 -15938.315283   5647.072201
##     Channel92     Channel94     Channel98     Channel99
##   6595.995241  -5497.846381  -8728.596111   8554.587048
```

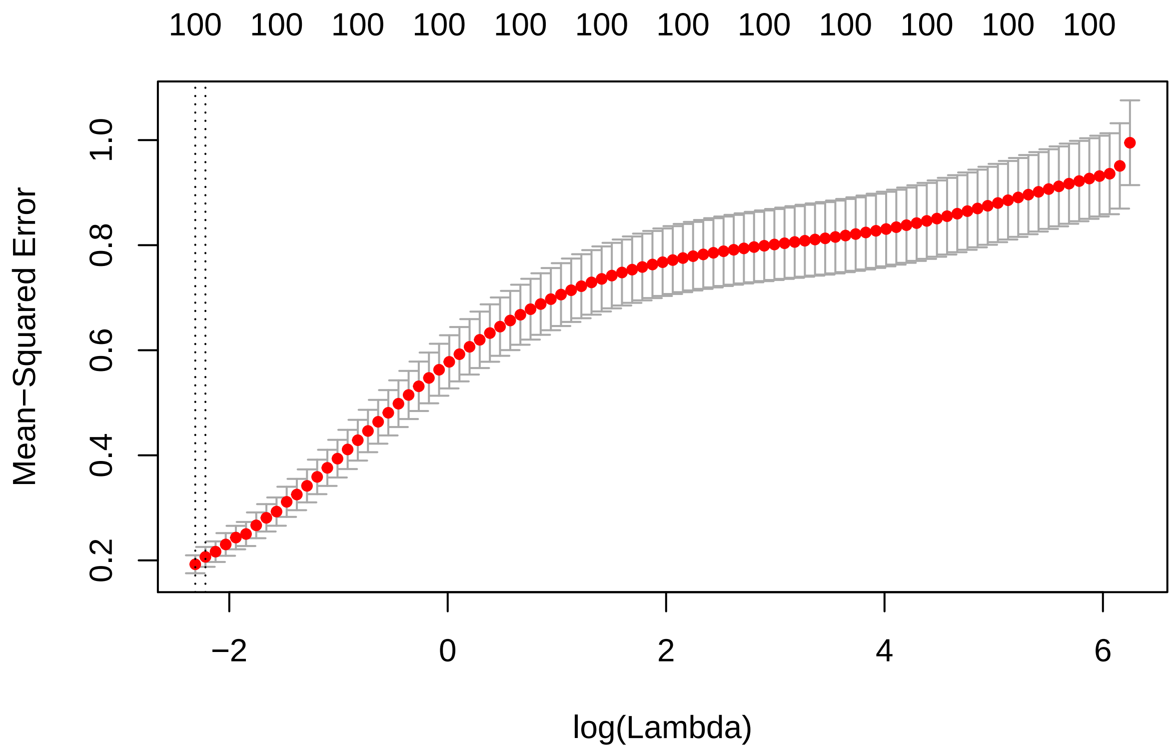
```
## [1] 64
```

63 Variables were selected and a model of the adjusted R squared of 99.25% was produced.

Q5




```
## [1] 0.09910954
```




```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -5.984460e-17
## Channel1    -1.031894e-01
## Channel2    -1.126216e-01
## Channel3    -1.214028e-01
## Channel4    -1.296478e-01
## Channel5    -1.372993e-01
## Channel6    -1.442254e-01
## Channel7    -1.502639e-01
## Channel8    -1.552611e-01
## Channel9    -1.591851e-01
## Channel10   -1.618359e-01
## Channel11   -1.633326e-01
## Channel12   -1.635877e-01
## Channel13   -1.624216e-01
## Channel14   -1.596083e-01
## Channel15   -1.550072e-01
## Channel16   -1.485948e-01
## Channel17   -1.404129e-01
## Channel18   -1.302440e-01
## Channel19   -1.180202e-01
## Channel20   -1.039948e-01
## Channel21   -8.880577e-02
## Channel22   -7.361606e-02
## Channel23   -5.957436e-02
```

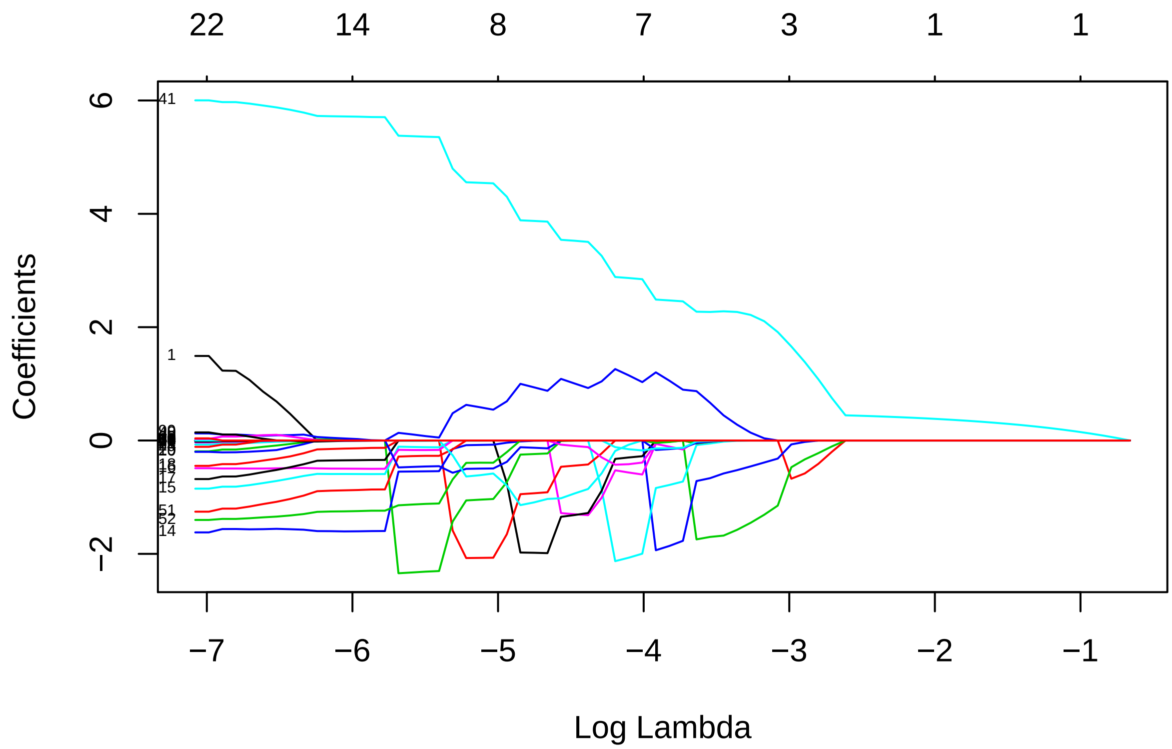
```
## Channel24   -4.721711e-02
## Channel25   -3.619736e-02
## Channel26   -2.534224e-02
## Channel27   -1.256519e-02
## Channel28    4.344829e-03
## Channel29    2.662692e-02
## Channel30    5.448642e-02
## Channel31    8.658293e-02
## Channel32    1.205339e-01
## Channel33    1.538239e-01
## Channel34    1.849436e-01
## Channel35    2.140008e-01
## Channel36    2.423180e-01
## Channel37    2.707945e-01
## Channel38    2.981771e-01
## Channel39    3.215073e-01
## Channel40    3.370674e-01
## Channel41    3.409983e-01
## Channel42    3.300942e-01
## Channel43    3.028880e-01
## Channel44    2.605030e-01
## Channel45    2.069027e-01
## Channel46    1.482506e-01
## Channel47    9.123195e-02
## Channel48    4.203255e-02
## Channel49    3.162517e-03
## Channel50   -2.490854e-02
## Channel51   -4.292261e-02
## Channel52   -5.225467e-02
## Channel53   -5.472501e-02
## Channel54   -5.253630e-02
## Channel55   -4.809605e-02
## Channel56   -4.348703e-02
## Channel57   -4.014073e-02
## Channel58   -3.842608e-02
## Channel59   -3.841251e-02
## Channel60   -3.933553e-02
## Channel61   -4.055333e-02
## Channel62   -4.184981e-02
## Channel63   -4.296650e-02
## Channel64   -4.398656e-02
## Channel65   -4.493753e-02
## Channel66   -4.583276e-02
## Channel67   -4.675245e-02
## Channel68   -4.748998e-02
## Channel69   -4.790305e-02
## Channel70   -4.780024e-02
## Channel71   -4.714768e-02
## Channel72   -4.592782e-02
## Channel73   -4.407423e-02
## Channel74   -4.169460e-02
## Channel75   -3.889640e-02
## Channel76   -3.574401e-02
## Channel77   -3.212053e-02
```

```
## Channel78   -2.774564e-02
## Channel79   -2.225572e-02
## Channel80   -1.532490e-02
## Channel81   -7.268784e-03
## Channel82    1.411225e-03
## Channel83    1.013919e-02
## Channel84    1.823082e-02
## Channel85    2.496689e-02
## Channel86    2.969929e-02
## Channel87    3.252596e-02
## Channel88    3.435886e-02
## Channel89    3.649918e-02
## Channel90    3.989390e-02
## Channel91    4.484971e-02
## Channel92    5.090499e-02
## Channel93    5.715961e-02
## Channel94    6.283132e-02
## Channel95    6.729909e-02
## Channel96    7.010424e-02
## Channel97    7.056400e-02
## Channel98    6.805063e-02
## Channel99    6.235443e-02
## Channel100   5.406867e-02
```

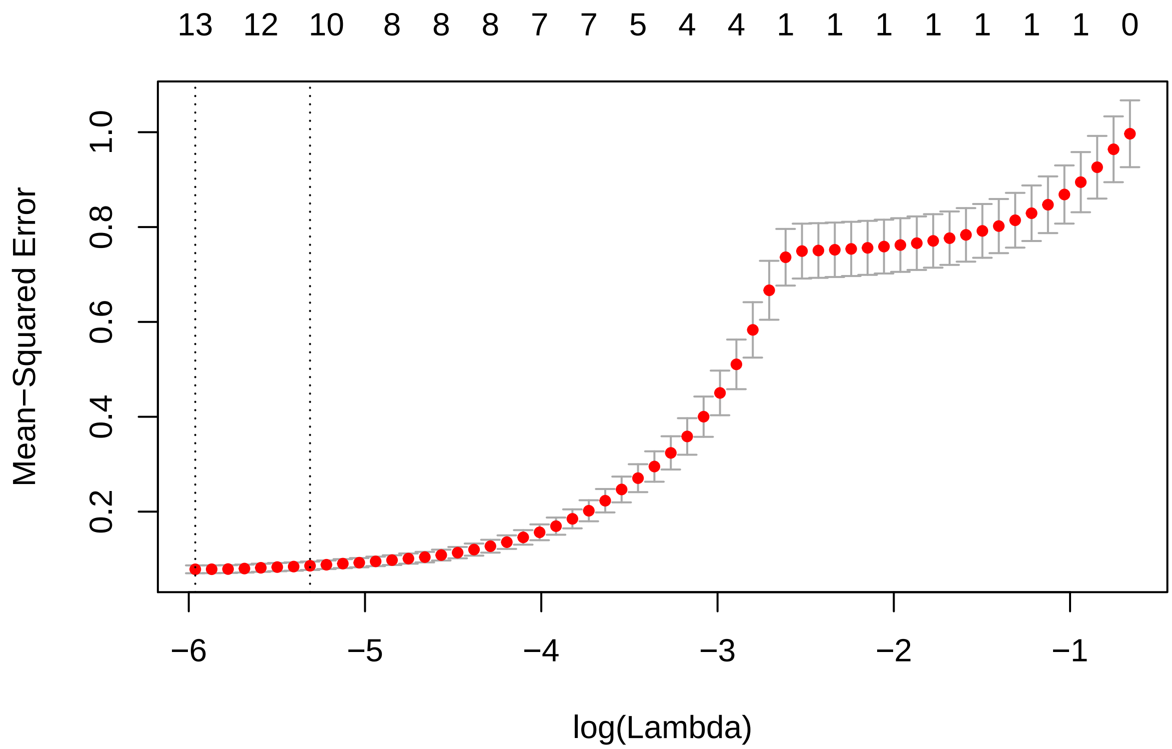
```
## [1] 0.09910954
```

The coefficients converge to 0 as the log lambda increases. They range between -0.1 and 0.3 approximately and almost converge to 0 by reaching the value 4 of log lambda. The least Mean Squared Error is under log lambda of -2.

Q6




```
## [1] 0.002571916
```




```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  2.430320e-15
## Channel1     .
## Channel2     .
## Channel3     .
## Channel4     .
## Channel5     .
## Channel6     .
## Channel7     .
## Channel8     .
## Channel9     .
## Channel10    .
## Channel11    .
## Channel12    .
## Channel13    .
## Channel14   -1.602254e+00
## Channel15   -5.920293e-01
## Channel16   -4.978047e-01
## Channel17   -3.471019e-01
## Channel18   -1.377282e-01
## Channel19    .
## Channel20    .
## Channel21    .
## Channel22    .
## Channel23    .
```

```
## Channel24    .
## Channel25    .
## Channel26    .
## Channel27    .
## Channel28    .
## Channel29    .
## Channel30    .
## Channel31    .
## Channel32    .
## Channel33    .
## Channel34    .
## Channel35    .
## Channel36    .
## Channel37    .
## Channel38    .
## Channel39    5.881678e-03
## Channel40    2.533079e-02
## Channel41    5.714424e+00
## Channel42    .
## Channel43    .
## Channel44    .
## Channel45    .
## Channel46    .
## Channel47    .
## Channel48    .
## Channel49   -2.466071e-05
## Channel50   -4.082507e-03
## Channel51   -8.735307e-01
## Channel52   -1.245545e+00
## Channel53   -1.069154e-03
## Channel54    .
## Channel55    .
## Channel56    .
## Channel57    .
## Channel58    .
## Channel59    .
## Channel60    .
## Channel61    .
## Channel62    .
## Channel63    .
## Channel64    .
## Channel65    .
## Channel66    .
## Channel67    .
## Channel68    .
## Channel69    .
## Channel70    .
## Channel71    .
## Channel72    .
## Channel73    .
## Channel74    .
## Channel75    .
## Channel76    .
## Channel77    .
```

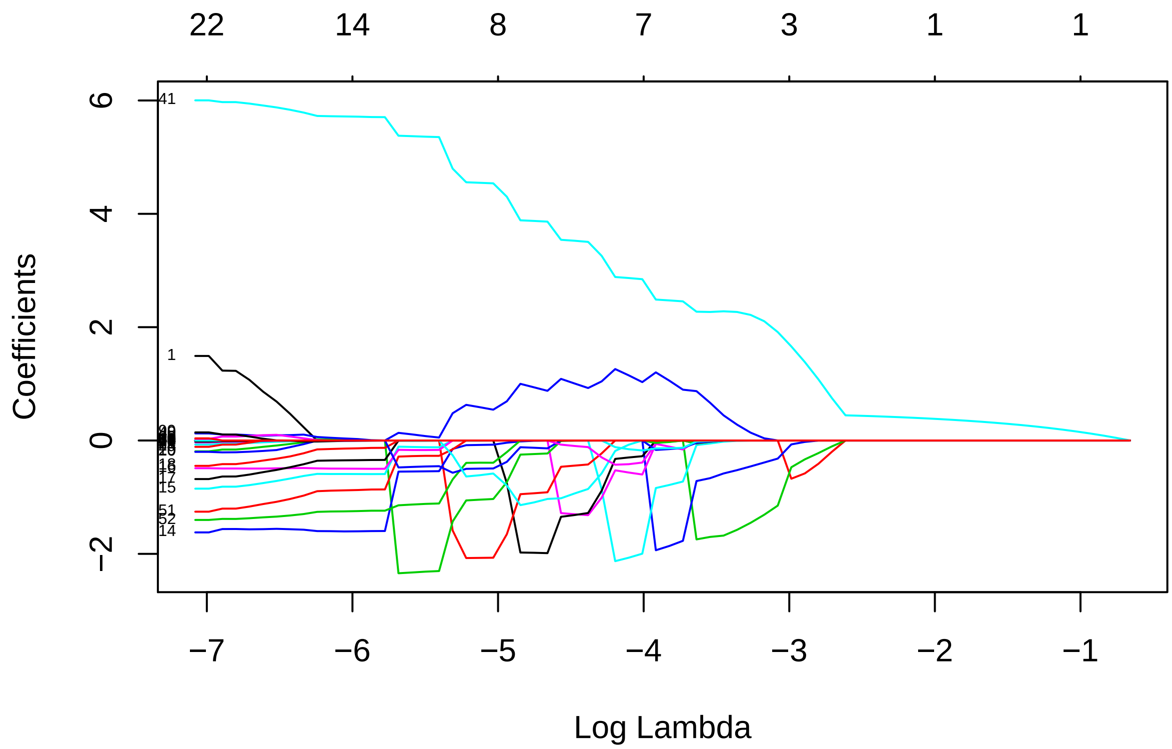
```
## Channel78     .
## Channel79     .
## Channel80     .
## Channel81     .
## Channel82     .
## Channel83     .
## Channel84     .
## Channel85     .
## Channel86     .
## Channel87     .
## Channel88     .
## Channel89     .
## Channel90     .
## Channel91     .
## Channel92     .
## Channel93     .
## Channel94     .
## Channel95     .
## Channel96     .
## Channel97     .
## Channel98     .
## Channel99     .
## Channel100    .
```

```
## [1] 0.002571916
```

The coefficient values for the 13 variables selected converge to 0 at the log lambda of -3. the least mse is found between log lambda of -6 to -5.5.

There are 100 variables that have been used in the ridge regression model and only 13 variables used in the lasso regression model. In Ridge regression there is propotional shrinkage that is done while Lasso translates each coefficient by a factor lambda truncating at 0. In the Ridge regression both shrinkage and

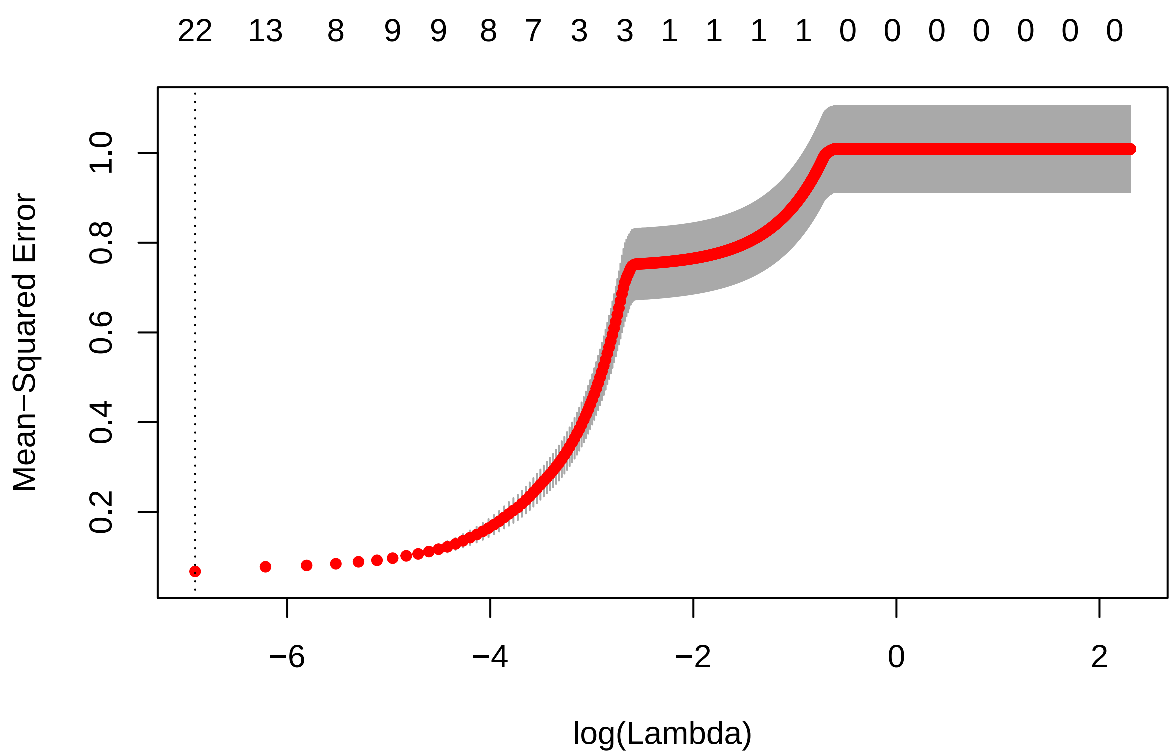
Q7




```
## 101 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  3.423729e-15
## Channel1     2.389874e+00
## Channel2     1.988995e-06
## Channel3     1.193300e-06
## Channel4    -3.161787e-06
## Channel5    -5.893738e-06
## Channel6     3.484921e-01
## Channel7    -7.283755e-02
## Channel8    -6.794308e-02
## Channel9     1.422145e-01
## Channel10   -7.818637e-02
## Channel11   -9.791948e-03
## Channel12   -6.749591e-02
## Channel13    8.343128e-03
## Channel14   -1.159804e-02
## Channel15   -1.105447e+00
## Channel16   -1.617016e-01
## Channel17   -9.932520e-01
## Channel18   -7.298657e-01
## Channel19   -2.581660e+00
## Channel20   -1.919222e-01
## Channel21   -1.431399e-01
## Channel22   -4.840860e-02
## Channel23   -3.651574e-02
```

```
## Channel24   -1.864789e-02
## Channel25    7.570411e-06
## Channel26    1.491368e-05
## Channel27    1.536979e-05
## Channel28    1.409084e-05
## Channel29    1.370188e-05
## Channel30    6.678097e-06
## Channel31   -6.166308e-06
## Channel32   -1.319432e-05
## Channel33   -8.494726e-06
## Channel34    1.139730e-05
## Channel35    3.363313e-05
## Channel36    5.203122e-05
## Channel37    6.456039e-05
## Channel38    6.255091e-05
## Channel39    4.143816e-05
## Channel40   -2.162560e-01
## Channel41    6.734801e+00
## Channel42   -1.260505e-05
## Channel43   -1.435354e-05
## Channel44   -2.446549e-05
## Channel45   -3.925484e-05
## Channel46   -5.191104e-05
## Channel47   -5.284406e-05
## Channel48   -2.916115e-05
## Channel49    1.534122e-01
## Channel50   -1.216789e-01
## Channel51   -1.601908e+00
## Channel52   -1.271089e+00
## Channel53   -7.792935e-02
## Channel54   -1.232931e-01
## Channel55   -8.068550e-02
## Channel56    1.832617e-02
## Channel57    1.809941e-01
## Channel58   -5.985316e-03
## Channel59   -4.819876e-02
## Channel60    6.648840e-02
## Channel61    1.663522e-01
## Channel62   -1.320815e-01
## Channel63   -9.210347e-02
## Channel64   -5.989582e-02
## Channel65   -9.942104e-05
## Channel66   -9.768134e-05
## Channel67   -8.517783e-05
## Channel68   -7.996782e-05
## Channel69   -8.946121e-05
## Channel70   -9.530787e-05
## Channel71   -8.319355e-05
## Channel72   -6.480279e-05
## Channel73   -6.565899e-05
## Channel74   -7.081182e-05
## Channel75   -5.436526e-05
## Channel76   -2.522736e-05
## Channel77   -1.878956e-05
```

```
## Channel78    -1.046285e-05
## Channel79    -4.135561e-06
## Channel80     7.920261e-06
## Channel81     1.224882e-05
## Channel82     1.636592e-05
## Channel83     1.766812e-05
## Channel84     1.797741e-05
## Channel85     2.643758e-05
## Channel86     3.403708e-05
## Channel87     4.460767e-05
## Channel88     5.152972e-05
## Channel89     5.712904e-05
## Channel90     5.687547e-05
## Channel91     5.839429e-05
## Channel92     6.316667e-05
## Channel93     6.450139e-05
## Channel94     6.473916e-05
## Channel95     6.403011e-05
## Channel96    -1.350519e-02
## Channel97     1.275633e-01
## Channel98     9.754263e-02
## Channel99     6.085942e-02
## Channel100    1.261776e-01
```




```
## number of variables chosen = 100
```

```
## optimal lambda= 0
```

```
##            Length Class  Mode
## lambda     10001  -none- numeric
## cvm        10001  -none- numeric
## cvsd       10001  -none- numeric
## cvup       10001  -none- numeric
## cvlo       10001  -none- numeric
## nzero      10001  -none- numeric
## name           1  -none- character
## glmnet.fit    12  elnet  list
## lambda.min     1  -none- numeric
## lambda.1se     1  -none- numeric
```

```
## [1] 1.770297
```

```
## [1] 48146
```

The value lambda minimum is found to be 0.This would make the MSE 100% which make the model and hence not optimal. The number of variables chosen are 100. The least MSE can be however founda at log lambda value -7.

Q8

The linear model made in step 4 selects 63 variables using StepAIC while the lasso model in step 7 no variable selection is done due to lambda value of 0 being considered in the procedure. 'Both' directional selection was used to perform this analysis.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(plotly)
library(ggplot2)
library(seriation)
library(glm2)
library(kknn)
library(cvTools)
library(xlsx)
library(MASS)
library(glmnet)
spambase = read.xlsx("spambase.xlsx", sheetName = "spambase_data", header = TRUE)
n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spambase[id,]
test=spambase[-id,]

spambase$Spam<-as.factor(spambase$Spam)
#build model on training dataset
```

```r
model <- glm(Spam ~ . , data=train, family = "binomial")
summary(model)

#predict Y
predictedtest <- predict(model,newdata=test ,type = "response")

p_class2 <- ifelse(predictedtest > 0.5, "Spam", "Not Spam")
table(p_class2)

tab2 <- table(test$Spam, p_class2)
tab2

#misClassError
1 - sum(diag(tab2))/sum(tab2) # for test data


#predict Y
predictedtrain<- predict(model,data=train, type = "response")
p_class1 <- ifelse(predictedtrain > 0.5, "Spam", "Not Spam")
table(p_class1)

#confusionMatrix
tab1 <- table(train$Spam, p_class1)
tab1

#misClassError
1 - sum(diag(tab1))/sum(tab1) # for training data


#3.

#predict Y
predictedY3 <- predict(model, newdata=test,type = "response")

p_class3 <- ifelse(predictedY3 > 0.90, "Spam", "Not Spam")
table(p_class3)

tab3 <- table(test$Spam, p_class3)
tab3

#misClassError
1 - sum(diag(tab3))/sum(tab3) # for test data

#predict Y
predictedY3.1 <- predict(model,data=train, type = "response")

p_class3.1 <- ifelse(predictedY3.1 > 0.90, "Spam", "Not Spam")
table(p_class3.1)

tab3.1 <- table(train$Spam, p_class3.1)
tab3.1

#misClassError
```

```r
1 - sum(diag(tab3.1))/sum(tab3.1) # for test data


#Q4 KNN K nearest neighbors

model4.1 <- kknn(Spam ~ .,train,test,k=30)
summary(model4.1)
predictedY4.1 <- predict(model4.1)

p_class4.1 <- ifelse(predictedY4.1 > 0.50, "Spam", "Not Spam")
table(p_class4.1)

tab4.1 <- table(test$Spam, p_class4.1)
tab4.1
1 - sum(diag(tab4.1))/sum(tab4.1)

model4.2 <- kknn(Spam ~ .,train,train,k=30)
summary(model4.2)
predictedY4.2 <- predict(model4.2)

p_class4.2 <- ifelse(predictedY4.2 > 0.50, "Spam", "Not Spam")
table(p_class4.2)

tab4.2 <- table(train$Spam, p_class4.2)
tab4.2
1 - sum(diag(tab4.2))/sum(tab4.2)

#5
model5.1 <- kknn(Spam ~ .,train,test,k=1)
summary(model5.1)
predictedY5.1 <- predict(model5.1)

p_class5.1 <- ifelse(predictedY5.1 > 0.50, "Spam", "Not Spam")
table(p_class5.1)

tab5.1 <- table(test$Spam, p_class5.1)
tab5.1
1 - sum(diag(tab5.1))/sum(tab5.1)


model5.2 <- kknn(Spam ~ .,train,train,k=1)
summary(model5.2)
predictedY5.2 <- predict(model5.2)

p_class5.2 <- ifelse(predictedY5.2 > 0.50, "Spam", "Not Spam")
table(p_class5.2)

tab5.2 <- table(train$Spam, p_class5.2)
tab5.2
1 - sum(diag(tab5.2))/sum(tab5.2)
data <- swiss
y <- as.vector(data[,1])
x <- as.matrix(data[,c(2:6)])
```

```r
Nfolds <- 5

weighter<- function(X,Y)
{
  w <- ginv(t(X)%*%X)%*%t(X)%*%Y
}
n <- dim(data)[1]
set.seed(12345)
sample_n <- sample(1:n)
ids <- list()
cvscore <- c()

cv_func <- function(X,Y,Nfolds)
{
  start <- 1
  for(i in 1:Nfolds)
    {
      if(i<Nfolds)
      {
        end <- start+(as.integer(n/Nfolds)-1)
        ids[[i]] <- sample_n[start:end]
        start <- end+1
      }
      else if(i==Nfolds)
      {
        end <- n
        ids[[i]] <- sample_n[start:end]
      }


      X_test <- X[as.vector(ids[[i]]),]
      X_train <- X[-as.vector(ids[[i]]),]
      Y_test <- Y[as.vector(ids[[i]])]
      Y_train <- Y[-as.vector(ids[[i]])]

      weights <- as.matrix(weighter(X=X_train,Y=Y_train))
      y_cap <- X_test%*%weights
      loss <- y_cap-Y_test
      cv <- sum(loss*loss)/length(Y_test)
      cvscore[i] <- cv
  }
  average_cv <- sum(cvscore)/Nfolds
}

cvseq<- matrix(0, nrow = 0, ncol = 3)
for(k in 1:ncol(x))
{
  combs <- combn(1:ncol(x),k)
  for(j in 1:ncol(combs))
  {
    x_new <- as.matrix(x[,combs[,j]])
    x_new <- cbind(x_new, 1) #adding the incercept
    seq <- paste(combs[,j], collapse = ",") #converting the sequence into string
```

```
    avg_cvscore <- cv_func(X = x_new,Y = y,Nfolds)
    cvseq <- rbind(cvseq, c(seq,avg_cvscore, k))
  }
}

cvseq = as.data.frame(cvseq)
colnames(cvseq) = c("Seq", "Loss","Xnum")
cvseq$Loss = as.numeric(as.character(cvseq$Loss))
cvseq$Seq = as.character(cvseq$Seq)

cv_func(X= x,Y= y,Nfolds)

cat("Optimal subset of features: ",cvseq$Seq[which.min(cvseq$Loss)])
cat("Cross validation score: ",min(cvseq$Loss))

save1<-ggplot()+geom_point(data= cvseq,aes(x= cvseq$Xnum,y= cvseq$Loss))
save1
#spambase = read.xlsx("spambase.xlsx", sheetName = "spambase_data", header = TRUE)

tecator = read.xlsx("tecator.xlsx", sheetName = "data", header = TRUE,row.names = 1)
#tecator=read.csv("tecator.csv", sep=',')
tecator1=as.data.frame(tecator)

n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=tecator[id,]
test=tecator[-id,]

fit= lm(Moisture~Protein,data=tecator)
plot(Moisture~Protein,data=tecator)
abline(fit,col="red")
coeff=coefficients(fit)
coeff
eq = paste0("y = ", round(coeff[2],1), "*x +", round(coeff[1],1))
eq
sm<-summary(fit)
sm
mse <-mean(sm$residuals^2)
mse
tecator$Protein2<-(tecator$Protein)^2
tecator$Protein3<-(tecator$Protein)^3
tecator$Protein4<-(tecator$Protein)^4
tecator$Protein5<-(tecator$Protein)^5
tecator$Protein6<-(tecator$Protein)^6

n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=tecator[id,]
validation=tecator[-id,]
```

```r
fit1.1= lm(Moisture~Protein,data=train,x=TRUE,y=TRUE)
coeff1.1=coefficients(fit1.1)
coeff1.1
eq1.1 = paste0("y = ", round(coeff[2],1), "*x +", round(coeff[1],1))
eq1.1
sm1.1<-summary(fit1.1)
sm1.1
mse1.1 <-mean((fit1.1$fitted.values-fit1.1$y)^2)
mse1.1
predict.validation1<-predict(fit1.1,validation)
mse1.1.1<-mean((predict.validation1-validation$Moisture)^2)


fit2.1= lm(Moisture~Protein+Protein2,data=train,x=TRUE,y=TRUE)
coeff2.1=coefficients(fit2.1)
coeff2.1
eq2.1 = paste0("y = ", round(coeff2.1[2],1), "*x +", round(coeff2.1[3],1),
               "*x^2 +", round(coeff2.1[1],1))
eq2.1
sm2.1<-summary(fit2.1)
sm2.1
mse2.1 <-mean((fit2.1$fitted.values-fit2.1$y)^2)
mse2.1
predict.validation2<-predict(fit2.1,validation)
mse2.1.1<-mean((predict.validation2-validation$Moisture)^2)


fit3.1= lm(Moisture~Protein+Protein2+Protein3,data=train,x=TRUE,y=TRUE)
coeff3.1=coefficients(fit3.1)
coeff3.1
eq3.1 = paste0("y = ", round(coeff3.1[2],1), "*x +", round(coeff3.1[3],1),
               "*x^2 +",round(coeff3.1[4],1), "*x^2 +",round(coeff3.1[1],1))
eq3.1
sm3.1<-summary(fit3.1)
sm3.1
mse3.1 <-mean((fit3.1$fitted.values-fit3.1$y)^2)
mse3.1
predict.validation3<-predict(fit3.1,validation)
mse3.1.1<-mean((predict.validation3-validation$Moisture)^2)


fit4.1= lm(Moisture~Protein+Protein2+Protein3+Protein4,data=train,x=TRUE,y=TRUE)
coeff4.1=coefficients(fit4.1)
coeff4.1
sm4.1<-summary(fit4.1)
sm4.1
mse4.1 <-mean((fit4.1$fitted.values-fit4.1$y)^2)
mse4.1
predict.validation4<-predict(fit4.1,validation)
mse4.1.1<-mean((predict.validation4-validation$Moisture)^2)


fit5.1= lm(Moisture~Protein+Protein2+Protein3+Protein4+Protein5,data=train
```

```
           ,x=TRUE,y=TRUE)
coeff5.1=coefficients(fit5.1)
coeff5.1
sm5.1<-summary(fit5.1)
sm5.1
mse5.1 <-mean((fit5.1$fitted.values-fit5.1$y)^2)
mse5.1
predict.validation5<-predict(fit5.1,validation)
mse5.1.1<-mean((predict.validation5-validation$Moisture)^2)


fit6.1= lm(Moisture~Protein+Protein2+Protein3+Protein4+Protein5+Protein6,data=train
           ,x=TRUE,y=TRUE)
coeff6.1=coefficients(fit6.1)
coeff6.1
sm6.1<-summary(fit6.1)
sm6.1
mse6.1 <-mean((fit6.1$fitted.values-fit6.1$y)^2)
mse6.1
predict.validation6<-predict(fit6.1,validation)
mse6.1.1<-mean((predict.validation6-validation$Moisture)^2)

msetrain<-c(mse1.1,mse2.1,mse3.1,mse4.1,mse5.1,mse6.1)
msevalidation<-c(mse1.1.1,mse2.1.1,mse3.1.1,mse4.1.1,mse5.1.1,mse6.1.1)
msevalues<-c(msetrain,msevalidation)
mseplotdata<-data.frame(id=1:6,msevalues,datasetfrom=c("train","train","train","train","train","train",

scattermse<-ggplot(mseplotdata,aes(y=mseplotdata$msevalues,x=mseplotdata$id,color=mseplotdata$datasetfr

scattermse

knitr::include_graphics("biasvar.png")

library(MASS)
attach(tecator1)
fit <- lm(Fat~.-Moisture-Protein,data=data.frame(tecator1))
step <- stepAIC(fit, direction="both",trace = FALSE)
step$anova
summary(step)
step$coefficients
length(step$coefficients)
covariates1=scale(tecator1[,1:100])
response1=scale(tecator1[, 101])
model1=glmnet(as.matrix(covariates1),response1, alpha=0,family="gaussian")
plot(model1, xvar="lambda", label=TRUE)

model=cv.glmnet(as.matrix(covariates1),response1, alpha=0,family="gaussian")
model$lambda.min
plot(model)
coef(model, s="lambda.min")
model$lambda.min
covariates2=scale(tecator1[,1:100])
response2=scale(tecator1[, 101])
```

```r
model2=glmnet(as.matrix(covariates2),response2, alpha=1,family="gaussian")
plot(model2, xvar="lambda", label=TRUE)

model=cv.glmnet(as.matrix(covariates2),response2, alpha=1,family="gaussian")
model$lambda.min
plot(model)
coef(model, s="lambda.min")
model$lambda.min
#sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
#sum((ynew-y)^2)
plot(model2, xvar="lambda", label=TRUE)
tecator2<-scale(tecator1)
covariates3=(tecator2[,1:100])
response3=(tecator2[, 101])
model9=cv.glmnet(as.matrix(covariates3), response3, alpha=1,family="gaussian",lambda=seq(0,10,0.001))
y=validation[,101]
ynew=predict(model9, newx=as.matrix(validation[, 1:100]), type="response")

coef(model9, s="lambda.min")
plot(model9)
cat(paste("number of variables chosen =",length(coef(model9,s="lambda.min"))-1))
optimal<-model9$lambda.min
cat(paste("optimal lambda=",optimal))
summary(model9)

#Coefficient of determination
sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
sum((ynew-y)^2)
```