732A99 ComputerLab1 Block1

Namita Sharma, Vyshnavi Pisupati, Sara Salahshour

11/24/2019

Assignment 1. Spam classification with nearest neighbors

1.1 Import and divide data into training and test sets

The spambase.xlsx data in imported and divided into training and test sets in the proportion (50% / 50%) using R code detailed in appendix 1.1

1.2 Logistic model to predict spam using classification threshold 0.5

Table 1: Confusion Matrix Training Data

	0	1
0	804	127
1	93	346

Table 2: Confusion Matrix Test Data

	0	1
0	808	143
1	92	327

Misclassification Rates

	Training	Test
Misclassification Rate	0.1605839	0.1715328

The misclassification rate for the training data is slightly better than the misclassification rate for the test data.

1.3 Logistic model to predict spam using classification threshold 0.8

Table 3: Confusion Matrix Training Data

	0	1
0	921	10
1	333	106

Table 4: Confusion Matrix Test Data

	0	1
0	931	20
1	314	105

Misclassification Rates

	Training	Test
Misclassification Rate	0.250365	0.2437956

Although the classification 0.5 has better misclassification rates on both training and test data sets when compared to the classification 0.8, it should be noted that the latter has fewer false positives (non-spam classified as spam) than the former.

Depending on the loss function and the penalty for classifying a non-spam email as spam (false positive) or a spam as non-spam (false negative), either of the models could be useful. If the loss in marking a non-spam email as spam is considered to be much greater than failing to detect a spam email, the classification 0.8 proves to be a better model.

1.4 K-Nearest Neighbours to classify spam emails using K = 30

Misclassification Rates

	Training	Test
Misclassification Rate	0.1671533	0.1715328

Table 5: Confusion Matrix Training Data

	0	1
0	779	152
1	77	362

Table 6: Confusion Matrix Test Data

	0	1
0	808	143
1	92	327

The misclassification rates of the kknn classification with k = 30 is similar to that of the logisticmodel with threshold = 0.5 for both training and test datasets. The two models are comparable in terms of their performance.

On the other hand, the kknn classification with k = 30 has lower misclassifcation rates for both training and test datasets when compared to the logistic model with threshold = 0.8.

As with the previous comparison, even though logistic classification 0.8 has a higher overall misclassification rate than kknn classification k = 30, it should be noted that the former has fewer false positives (detecting a non-spam as spam) and if the loss in classifying a non-spam email as spam is considered to be much greater than failing to detect a spam email, the classification 0.8 proves to be a better model overall.

1.5 K-Nearest Neighbours to classify spam emails using K = 1

Table 7: Confusion Matrix Training Data

	0	1
0	931	0
1	0	439

Table 8: Confusion Matrix Test Data

	0	1
0	644	307
1	185	234

Misclassification Rates

	Training	Test
Misclassification Rate	0	0.3591241

By reducing k, The misclassification rate using the kknn classification with k = 1 is zero for training data but very high for the test data. The model is overfitted to the training data and is poorer compared to the logistic model with threshold = 0.5 or 0.8 because it has higher misclassification rates on the test dataset.

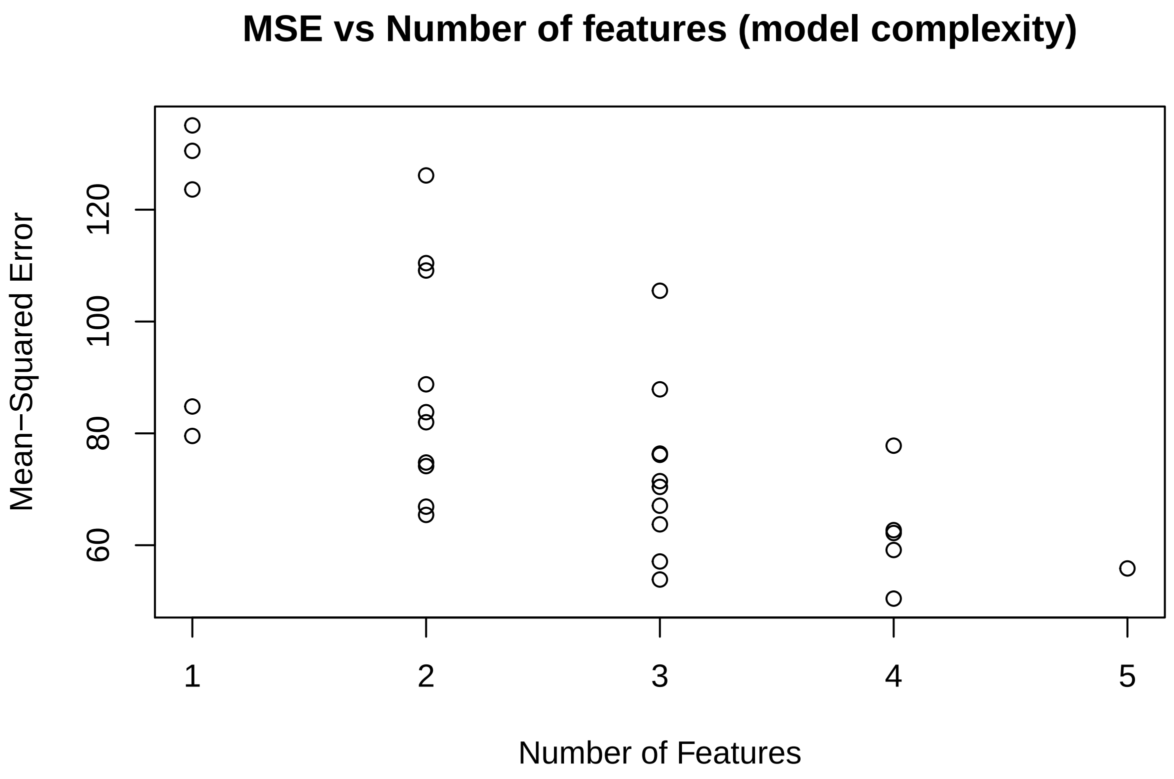
NOTE: The 0 misclassification rate in case of the training dataset can be attributed to the fact that the same dataset is used to train as well as test the model. Hence, every point in the dataset used for testing the model is closest to itself for k = 1 nearest neighbour classification.

Assignment 3. Feature selection by cross-validation in a linear model.

3.1 Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold

Cross validation in linear regression is implemented using R code detailed in appendix 3.1. The custom function calculates the CV scores for all subsets of features using k-fold cross validation in each linear model and determines the model with optimum subset of features having the least MSE.

3.2 Test implementation using swiss dataset



	CV	Features
Best Fit Model	50.44948	1 0 1 1 1

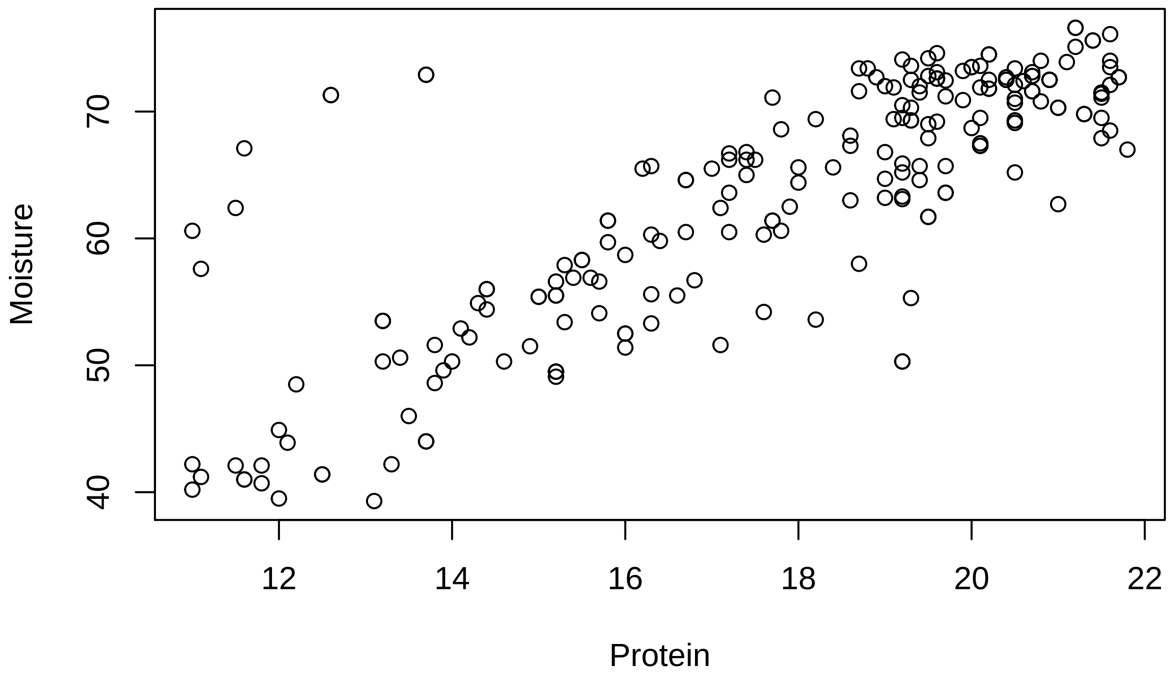
Selected Features = Agriculture, Education, Catholic, Infant.Mortality

The optimal model has a subset of 4 features - Agriculture, Education, Catholic and Infant.Mortality. It is reasonable that this subset of features has the largest impact on the target Fertility because this subset has the lowest MSE. This can be explained because the fifth feature Examination which is dropped in the best model is highly correlated with all the other features (predictors). High multicollinearity reduces the precision of the estimated coefficients (increases standard error) as it becomes harder to determine the effect of individual predictors on the target. Hence, eliminating this highly correlated feature results in better estimates of the target for changes in each of the remaining individual predictors and lowers the MSE of the model predictions.

It can also be observed from the graph that with an increase in the number of features/ predictors in the model (complexity), the MSE reduces.

Assignment 4. Linear regression and regularization

1) Imported the data into R. A plot of protein and moisture is drawn.



It is seen from the graph clearly that moisture and protein follow a linear relation with a few outliers.So the data can be described well with a linear model.

2) Probabilistic model that describes Mi in which Moisture is a polynomial function of Protein including terms up to power i

Since the target variables follow a normal distribution, taking the maximum likelyhood, results in the minimuzation of the MSE.Say, n samples are taken.Here,n=215 which is the number of rows of data set given.

$$P\left(Y_i|\mu,\sigma\right)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\dfrac{\left(Y_i-\mu\right)^2}{2\sigma^2}}$$

$$\mu=w^TX_i$$

Calculating the maximum likelyhood fucntion of Y.

$$l=\prod_{i=1}^{n}\frac{1}{\left(\sqrt{2\pi}\sigma\right)^n}e^{-\dfrac{\sum_{i=1}^{n}\left(Y_i-w^TX_i\right)^2}{2\sigma^2}}$$

To find $\hat{w}$, we maximize $ln\left(l\right)$ or minimize $-ln\left(l\right)$.

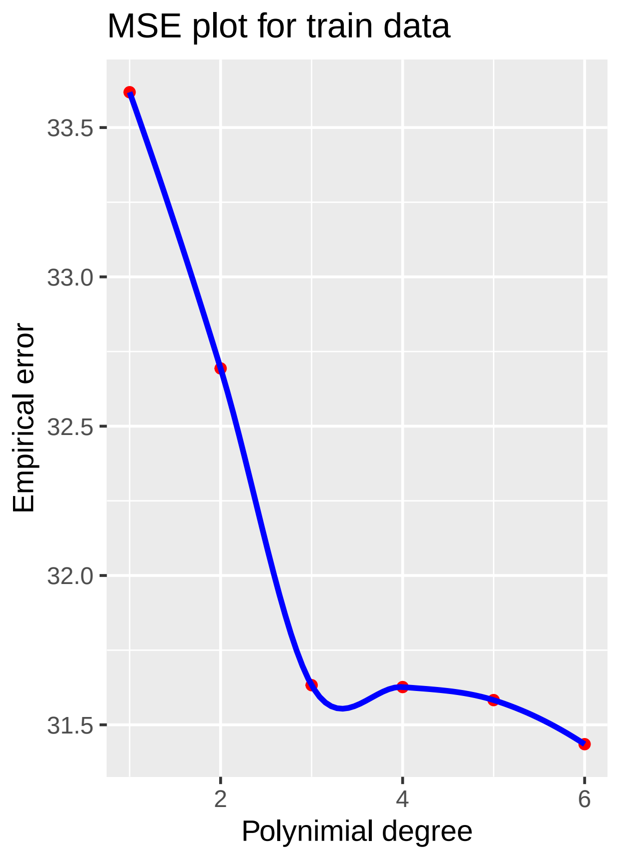
Taking $-ln\left(l\right)$

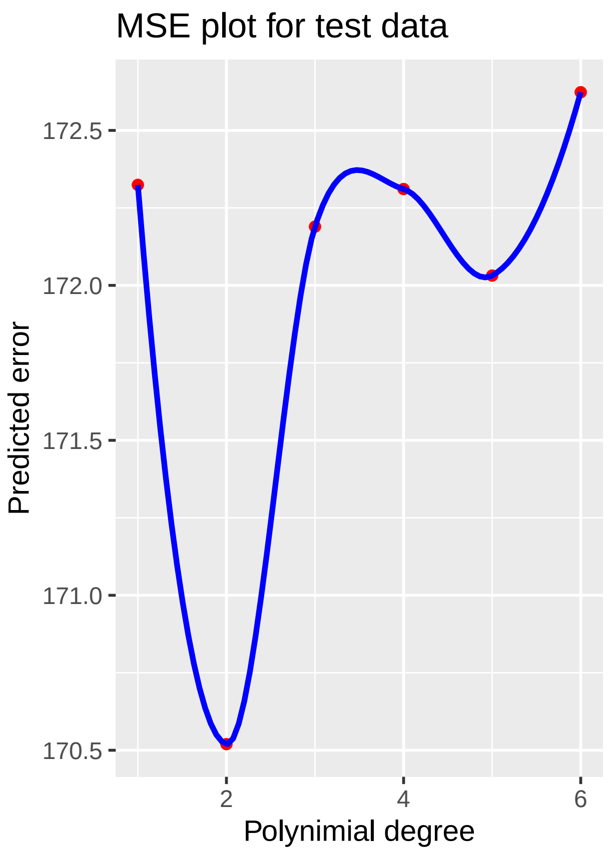
$$-ln\left(l\right) = \frac{n}{2}ln\left(2\pi\sigma^2\right) + \frac{\sum_{i=1}^{n}\left(Y_i - w^TX_i\right)^2}{2\sigma^2}$$

Since $\frac{n}{2}ln\left(2\pi\sigma^2\right)$ consists of constants, in order to minimize the $-ln\left(l\right)$, we must minimize $\sum_{i=1}^{n}\left(Y_i - w^TX_i\right)^2$ which means applying the crieteria of minimizing the MSE is appropriate.

3) Fit Models Mi, i = 1,2,3,4,5,6 for training and test datasets

Dividing the data into training and test data set randomly.Linear models with M_i are fitted with $i = 1...6$ where, i represents the degree of the polynomial used to fit the model though the models are linear in the parameters $\hat{w}$.





From the graphs, it can be seen that when the degree of polynomial is 1 both the traning error and test error are high and also the variance(this can be see by the difference between the Traning and Test error)is high.When the degreee of the polynomial is 2,there is drops and reaches the minimum value, though the train MSE is not at its minumum,the variance is for this model is low.When the polynomial degree is 3,the traning MSE drops drastically, while, there is surge in the test MSE. This trend continues for the all higher polynomial degress. Though the traning MSE is decreasing the test MSE is increasing, indicating that model is overfitting the data.Hence, model 2 with polynomial degree 2 is the best model with minimum variance and no overfitting.

Taking a look at the loss function below, for a given model, as the variance decreases, the bias increases and vice-versa. Hence there is always a trade-off between the bias and variance. Choosing a model with low variance, increases the bias.In this case model 2 is choosen, though the variance is low, the bias is high.

$$R\left(Y\left(x_o\right), \hat{Y}\left(x_o\right)\right) = \sigma^2 + Bias\left(\hat{Y}\left(x_o\right)\right)^2 + Var\left(\hat{Y}\left(x_o\right)\right)$$

4) USing stepAIC() function for feature selection for a linear model.

```
## The below features are identified as the most significant :
```

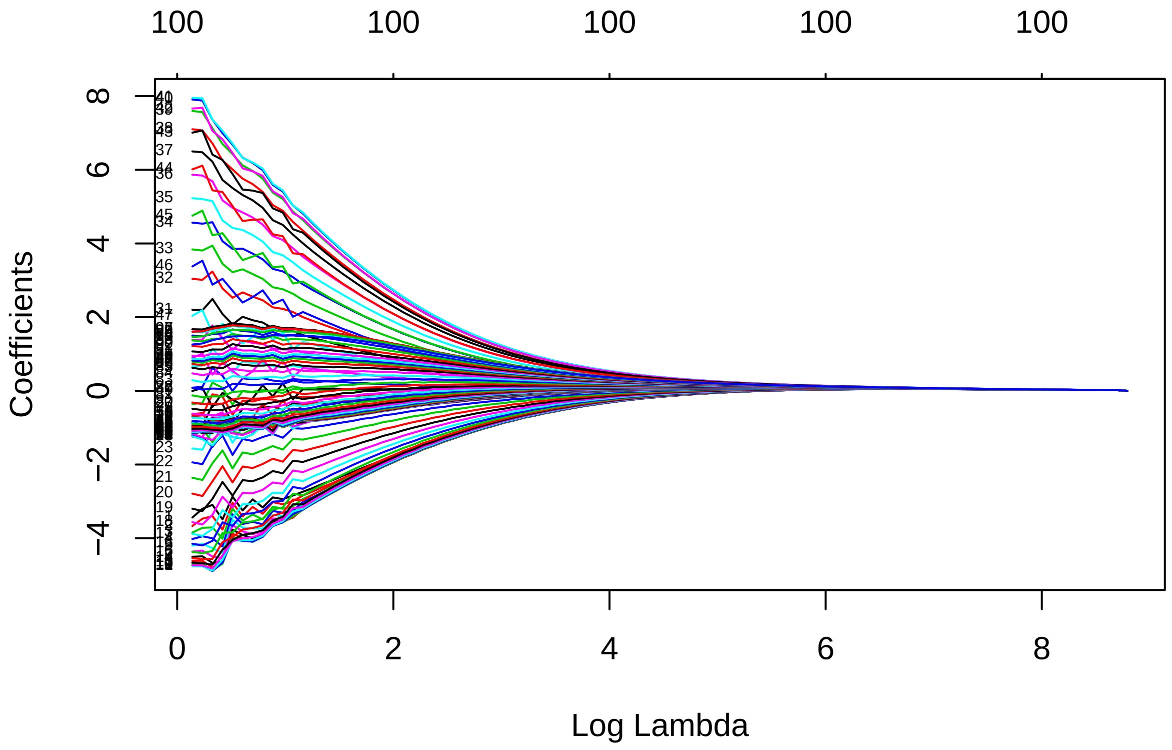
```
##  [1] "Channel1"  "Channel2"  "Channel4"  "Channel5"  "Channel7" 
##  [6] "Channel8"  "Channel11" "Channel12" "Channel13" "Channel14"
## [11] "Channel15" "Channel17" "Channel19" "Channel20" "Channel22"
## [16] "Channel24" "Channel25" "Channel26" "Channel28" "Channel29"
## [21] "Channel30" "Channel32" "Channel34" "Channel36" "Channel37"
## [26] "Channel39" "Channel40" "Channel41" "Channel42" "Channel45"
## [31] "Channel46" "Channel47" "Channel48" "Channel50" "Channel51"
## [36] "Channel52" "Channel54" "Channel55" "Channel56" "Channel59"
## [41] "Channel60" "Channel61" "Channel63" "Channel64" "Channel65"
## [46] "Channel67" "Channel68" "Channel69" "Channel71" "Channel73"
## [51] "Channel74" "Channel78" "Channel79" "Channel80" "Channel81"
## [56] "Channel84" "Channel85" "Channel87" "Channel88" "Channel92"
## [61] "Channel94" "Channel98" "Channel99"
```

```
## Number of fetures selected = 63
```

```
## [1] 0.8598985
```

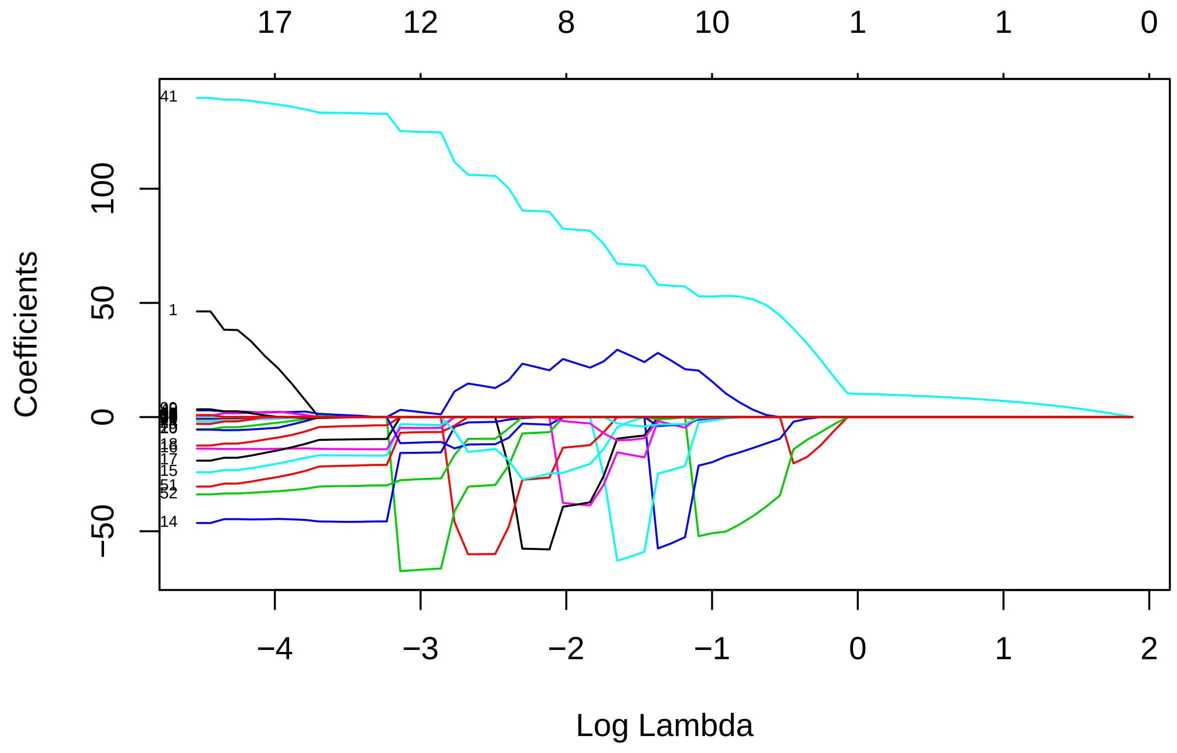
A total of 100 features were given as the input to the stepAIC function which eveluates removes and adds the features based on the AIC values calculated in each iteration.Finally, 63 features were identified to be most significant in determining Fat.

5) Ridge regression.



As λ increases, the number of the coefficients tending to zero increases.After the value $Log(\lambda)$ is greater than 6, all the coeffcients tend to zero. Hence an optimal value of λ needs to be choosen which lets the coefficients of less significant features to become zero.This can be done by cross-validation.

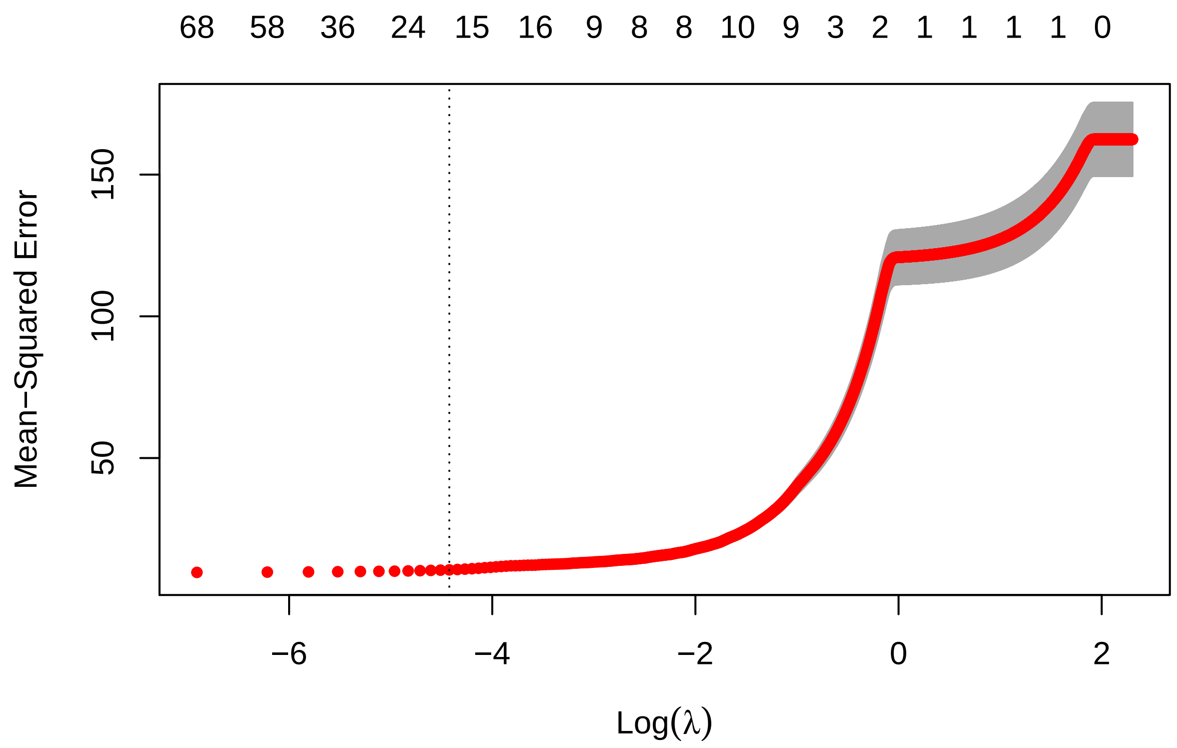
6) LASSO



In LASSO also, as λ increases the number of the coefficients tending to zero increases.In LASSO however, as $Log(\lambda)$ reaches 0, all the coefficients tend to zero.In the case of ridge regression, the decrease is gradual while in LASSO, the decrease is drastic.

Choosing an optimum value of lambda and thus selecting an optimal subset of features can prevent complex models which generally overfit to the training data, as seen in case of simple linear regression in exercise 4.3. When lamda $= 0$, the predicted coefficients are same as that of linear regression and as lambda -> Inf (lambda > 1 in this case), they shrink to zero. The optimum value of lambda can be chosen by performing cross validation lasso regression.

7) Cross-validation for LASSO model.



	Lambda	Measure	SE	Nonzero
Min	0.000	9.488881	1.131688	100
1se	0.012	10.552218	1.013034	17

It can be seen from the graph that as the penalty factor λ increases, the MSE also increases. Combining the results from exercise 4.6, it can be said that as the lambda increases, more and more features are dropped from the model (coefficients shrink to zero) making the model simpler which results in a higher mean squared error.

Hence the lowest MSE recorded is 9.49 for $\lambda = 0$ (simple linear regression) where all of the 100 features are selected in the model. But while $\lambda.min$ gives the least MSE, it is clearly not the most optimum. A good model would be one which is not overly complex, but at the same time has low MSE with an optimal selection of features. It can be observed that $\lambda.1se = 0.012$ gives the largest subset of features such that the MSE is within 1 standard error of the minimum MSE. In other words, for a small increase in MSE value from the minimum MSE, the model is optimized by selecting only 17 features out of 100.

8) Compare results from 4 and 7

	Measure	Features
step AIC	0.8598985	63
CV Lasso	10.5522180	17

In step 4, which uses AIC method for feature selection, 63 features were selected as the most significant features while in step 7 which uses LASSO a coefficient decomposition technique,17 features were selected. LASSO model should be selected as it is less complex when compared to the stepAIC model.

Appendix

```r
##############################################################
# Assignment 1 - Spam classification with nearest neighbors
##############################################################

# 1.1 Import spam data from excel file
spambase <- xlsx::read.xlsx(
              file      = "C:/Users/namit/Downloads/Machine Learning/Lab1/spambase.xlsx",
              sheetName = "spambase_data",
              header    = TRUE)

# No of observations in the dataset
n <- dim(spambase)[1]

# Divide dataset into training and test data
set.seed(12345)
index <- sample(1:n, floor(n / 2))
train <- spambase[index, ]
test  <- spambase[-index, ]

# Create a logistic regression model using the training dataset
logistic.reg <- glm(formula = train$Spam ~ .,
                    data    = train,
                    family  = binomial(link = "logit"))

# Use the logistic model to predict spam in the training dataset
result_train <- predict(object  = logistic.reg,
                        newdata = train,
                        type    = "response")

# Use the logistic model to predict spam in the test dataset
result_test  <- predict(object  = logistic.reg,
                        newdata = test,
                        type    = "response")

#------------------------------------------------------------------------------
# 1.2 using classification threshold 0.5, compare the confusion matrices and
# misclassification rates for training and test datasets

train_pred1          <- ifelse(result_train > 0.5, 1, 0)
train_conf_mat1      <- table(actual = train$Spam, predicted = train_pred1)
train_misclass_rate1 <- 1 - (sum(diag(train_conf_mat1)) / sum(train_conf_mat1))

test_pred1           <- ifelse(result_test > 0.5, 1, 0)
test_conf_mat1       <- table(actual = test$Spam, predicted  = test_pred1)
test_misclass_rate1  <- 1 - (sum(diag(test_conf_mat1)) / sum(test_conf_mat1)))

#------------------------------------------------------------------------------
# 1.3 using classification threshold 0.8, compare the confusion matrices and
# misclassification rates for training and test datasets

train_pred2          <- ifelse(result_train > 0.8, 1, 0)
```

```r
train_conf_mat2      <- table(actual = train$Spam, predicted = train_pred2)
train_misclass_rate2 <- 1 - (sum(diag(train_conf_mat2)) / sum(train_conf_mat2))

test_pred2           <- ifelse(result_test > 0.8, 1, 0)
test_conf_mat2       <- table(actual = test$Spam, predicted  = test_pred2)
test_misclass_rate2  <- 1 - (sum(diag(test_conf_mat2)) / sum(test_conf_mat2))

#------------------------------------------------------------------------------
# 1.4 Use kknn to classify spam emails in training and test datasets
# using k = 30 nearest neighbours

# Classify Spam in training dataset
kknn_train1 <- kknn::kknn(formula = Spam ~ .,
                          train   = train,
                          test    = train,
                          k       = 30)

# Classify Spam in test dataset
kknn_test1  <- kknn::kknn(formula = Spam ~ .,
                          train   = train,
                          test    = test,
                          k       = 30)

# Because there are only two classes, Spam = 1 and No Spam = 0,
# we classify the fitted values as 0 or 1 based on threshold
# probability 0.5
train_pred_1         <- ifelse(kknn_train1$fitted.values > 0.5, 1, 0)
train_conf_mat1      <- table(actual = train$Spam, predicted = train_pred_1)
train_misclass_rate1 <- 1 - (sum(diag(train_conf_mat1)) / sum(train_conf_mat1))

test_pred_1          <- ifelse(kknn_test1$fitted.values > 0.5, 1, 0)
test_conf_mat        <- table(actual = test$Spam, predicted = test_pred_1)
test_misclass_rate1  <- 1 - (sum(diag(test_conf_mat1)) / sum(test_conf_mat1))

#------------------------------------------------------------------------------
# 1.5 Use kknn to classify spam emails in training and test datasets
# using k = 1 nearest neighbours

# Classify Spam in training dataset
kknn_train2 <- kknn::kknn(formula = Spam ~ .,
                          train   = train,
                          test    = train,
                          k       = 1)

# Classify Spam in test dataset
kknn_test2  <- kknn::kknn(formula = Spam ~ .,
                          train   = train,
                          test    = test,
                          k       = 1)

# Because there are only two classes, Spam = 1 and No Spam = 0,
# we classify the fitted values as 0 or 1 based on threshold
# probability 0.5
```

```r
train_pred_2         <- ifelse(kknn_train2$fitted.values > 0.5, 1, 0)
train_conf_mat2      <- table(actual = train$Spam, predicted = train_pred_2)
train_misclass_rate2 <- 1 - (sum(diag(train_conf_mat2)) / sum(train_conf_mat2))

test_pred_2          <- ifelse(kknn_test2$fitted.values > 0.5, 1, 0)
test_conf_mat2       <- table(actual = test$Spam, predicted = test_pred_2)
test_misclass_rate2  <- 1 - (sum(diag(test_conf_mat2)) / sum(test_conf_mat2))

###########################################################################
# Assignment 3 - Feature selection by cross-validation in a linear model
###########################################################################

# 3.1 Implement an R function that performs feature selection (best
# subset selection) in linear regression by using k-fold cross-validation

# Linear regression
mylin = function(X, Y, Xpred) {
  X1     <- cbind(1, X)
  Xpred1 <- cbind(1, Xpred)
  beta   <- solve(t(X1) %*% X1) %*% (t(X1) %*% Y)
  Res    <- Xpred1 %*% beta
  return(Res)
}

# Cross validation
myCV = function(X ,Y, Nfolds) {
  n <-length(Y)
  p <- ncol(X)

  set.seed(12345)
  ind      <- sample(n, n)
  X1       <- X[ind, ]
  Y1       <- Y[ind]
  sF       <- floor(n / Nfolds)
  MSE      <- numeric(2 ^ p - 1)
  Nfeat    <- numeric(2 ^ p - 1)
  Features <- list()
  curr     <- 0

  # We assume 5 features.
  for (f1 in 0:1)
    for (f2 in 0:1)
      for(f3 in 0:1)
        for(f4 in 0:1)
          for(f5 in 0:1) {
            model <- c(f1, f2, f3, f4, f5)
            if (sum(model) == 0) next()
            SSE <- 0

            for (k in 1:Nfolds) {
              # Compute which indices should belong to current fold
              index   <- (((k - 1) * sF) + 1):(k * sF)
              # Implement cross-validation for model with features in "model"
              # and iteration k.
              test_x  <- X1[index, which(model == 1)]
              train_x <- X1[-index, which(model == 1)]
              Yp      <- Y1[index]
              train_y <- Y1[-index]

              # Get the predicted values for fold 'k', Ypred, and the original
              # values for fold 'k', Yp.
              Ypred <- mylin(train_x, train_y, test_x)
              SSE   <- SSE + sum((Ypred - Yp) ^ 2)
            }
            curr             <- curr + 1
            MSE[curr]        <- SSE / n
            Nfeat[curr]      <- sum(model)
            Features[[curr]] <- model
          }

  # Plot MSE against number of features
  plot(Nfeat, MSE)

  # Best fit model
  i <- which.min(MSE)
  return(list(CV = MSE[i], Features = Features[[i]]))
}

#------------------------------------------------------------------------------
# 3.2 Test implementation using swiss dataset
myCV(as.matrix(swiss[, 2:6]), swiss[[1]], 5)

########################################################
## Assignment 4 - Linear Regression and Regularization
########################################################

# 4.1 Import tecator data from excel file
tecator <- xlsx::read.xlsx(
  file      = "C:/Users/namit/Downloads/Machine Learning/Lab1/tecator.xlsx",
  sheetName = "data",
  header    = TRUE)

# Plot Moisture vs Protein
plot(tecator$Moisture, tecator$Protein)

#------------------------------------------------------------------------------
# 4.2 Models Mi where expected moisture is a polynomial function (upto power i)
# of protein and the moisture is normally distributed

for (i in 1:6) {
  plot(tecator$Moisture, tecator$Protein ^ i)
}

#------------------------------------------------------------------------------
# 3. Fit Models Mi, i = 1,2,3,4,5,6 for training and test datasets
```

```r
# No of observations in the dataset
n <- dim(tecator)[1]

# Divide dataset into training and test data
set.seed(12345)
index <- sample(1:n, floor(n / 2))
train <- tecator[index, ]
test  <- tecator[-index, ]

MSE_training   <- numeric(6)
MSE_validation <- numeric(6)

for(i in 1:6) {
  # For Model Mi: Mosture ~ Protein ^ i, calculate the training and
  # validation MSE on training and test datasets

  # Training dataset
  linreg            <- lm(formula = train$Moisture ~ poly(train$Protein, i, raw = TRUE),
                             data    = train)
  Ypred             <- predict(object   = linreg,
                                  newdata = train,
                                  type    = "response")
  Yp                <- train$Moisture
  MSE_training[i]   <- mean((Ypred - Yp) ^ 2)

  # Test dataset
  Ypred             <- predict(object   = linreg,
                                  newdata = test,
                                  type    = "response")
  Yp                <- test$Moisture
  MSE_validation[i] <- mean((Ypred - Yp) ^ 2)
}

# Plot training and validation MSE vs highest degree of protein
# polynomial function
ggplot2::ggplot(mapping = ggplot2::aes(x = 1:6, y = MSE_training)) +
ggplot2::geom_point(shape = 1) +
ggplot2::theme_classic()       +
ggplot2::theme(plot.title   = ggplot2::element_text(hjust = 0.5),
               panel.border = ggplot2::element_rect(fill = NA))  +
ggplot2::geom_smooth(method = stats::loess, se = FALSE, col = 2) +
ggplot2::labs(x = "Highest degree of Protein function",
              y = "Training MSE")

ggplot2::ggplot(mapping = ggplot2::aes(x = 1:6, y = MSE_validation)) +
ggplot2::geom_point(shape = 1) +
ggplot2::theme_classic()       +
ggplot2::theme(plot.title   = ggplot2::element_text(hjust = 0.5),
               panel.border = ggplot2::element_rect(fill = NA))  +
ggplot2::geom_smooth(method = stats::loess, se = FALSE, col = 4) +
ggplot2::labs(x = "Highest degree of Protein function",
              y = "Validation MSE")
```

```r
#------------------------------------------------------------------------------
# 4.4 Variable selection using stepAIC
linreg  <- lm(formula = Fat ~ . - Sample - Protein - Moisture,
              data    = tecator)
stepAIC <- MASS::stepAIC(object    = linreg,
                         direction = "both",
                         trace     = FALSE)

#------------------------------------------------------------------------------
# 4.5 Ridge Regression
ridgereg <- glmnet::glmnet(x      = as.matrix(tecator[ ,2:101]),
                           y      = tecator$Fat,
                           family = "gaussian",
                           alpha  = 0)
plot(ridgereg, xvar="lambda", label=TRUE)

#------------------------------------------------------------------------------
# 4.6 Lasso Regresssion
lassoreg <- glmnet::glmnet(x      = as.matrix(tecator[ ,2:101]),
                           y      = tecator$Fat,
                           family = "gaussian",
                           alpha  = 1)
plot(lassoreg, xvar="lambda", label=TRUE)

#------------------------------------------------------------------------------
# 4.7 Cross validation to find optimal LASSO model
lasso.cv <- glmnet::cv.glmnet(x            = as.matrix(tecator[ ,2:101]),
                              y            = tecator$Fat,
                              family       = "gaussian",
                              alpha        = 1,
                              lambda       = seq(0, 7, 0.002),
                              type.measure = "mse")

# Minimum and Optimum lambda
lambda.min <- which(lasso.cv$lambda == lasso.cv$lambda.min)
lambda.1se <- which(lasso.cv$lambda == lasso.cv$lambda.1se)

# Number of features selected for optimum lambda
feat_optimum <- lasso.cv$nzero[lasso.cv$lambda == lasso.cv$lambda.1se]
cv_optimum   <- lasso.cv$cvm[lasso.cv$lambda == lasso.cv$lambda.1se]

# CV score w.r.t lambda
plot(x    = log(lasso.cv$lambda),
     y    = lasso.cv$cvm,
     main = "CV score w.r.t Lambda",
     xlab = "Log Lambda",
     ylab = "Mean-Squared Error",
     col  = "red" )
```