LAb1_machine_learning

Yash Pawar

22/11/2019

Assignment 1.1

Dataset is divided into two parts. 50% is used for training and the other 50%

Assignment 1.2

The misclassification rate for the training data is less as compared to the test data. This is due to the fact that classification and testing of the model is done on the same data in case of Training data.

```
##    condition_1
##       0   1
##   0 808 143
##   1  92 327
```

```
## [1] 0.1715328
```

```
##    condition_3
##       0   1
##   0 804 127
##   1  93 346
```

```
## [1] 0.1605839
```

Assignment 1.3

For the particular classification principle P(Y=1|X) > 0.8, the misclassification rate for training data is low as compared to test data. However, it is higher as compared to the classification principle P(Y=1|X) > 0.5 in both the cases of training and testing data.

What effect did the new rule have?

Significantly less mails have been classified as spam for this classification principle as the threshold for classification is high.

Assignment 1.4

As compared to the logistic regression models, the misclassification rate for the K- nearest neighbours model with k = 30 is significantly low. The misclassification rate for training data is lower as compared to the testing data.

```
##    pred.knn
##       0   1
##   0 702 249
##   1 180 239
```

```
## [1] 0.3131387
```

```
##    pred.knn_1
##       0   1
##   0 779 152
##   1  77 362
```

```
## [1] 0.1671533
```

Assignment 1.5

When $k = 1$, the misclassification rate for the test data is very high. From this we can conclude that higher is the k, better is the classification. However the misclassification rate for training data in this case is *zero* as it classifies all the nearest neighbours correctly (i.e. 1 neighbour) and the testing is done on the same dataset.

```
##    pred.knn_2
##       0   1
##   0 560 371
##   1 269 170
```

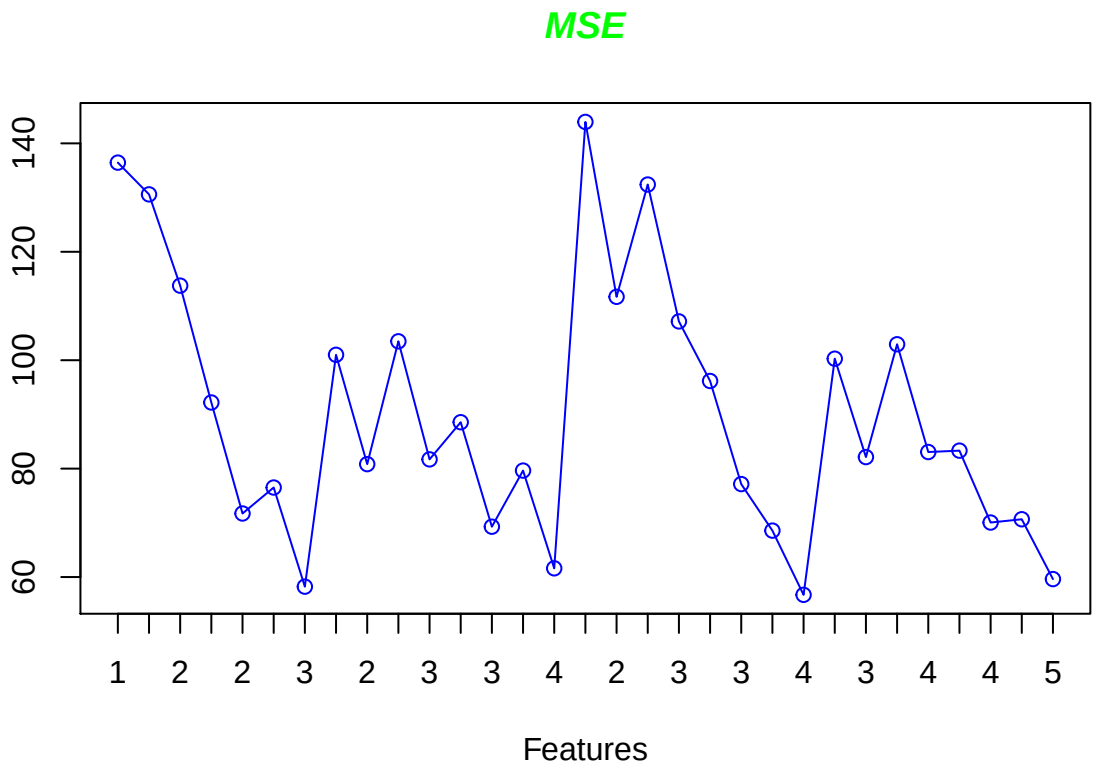
```
## [1] 0.4671533
```

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

```
##    pred.knn_3
##       0   1
##   0 931   0
##   1   0 439
```

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

Assignment 3:



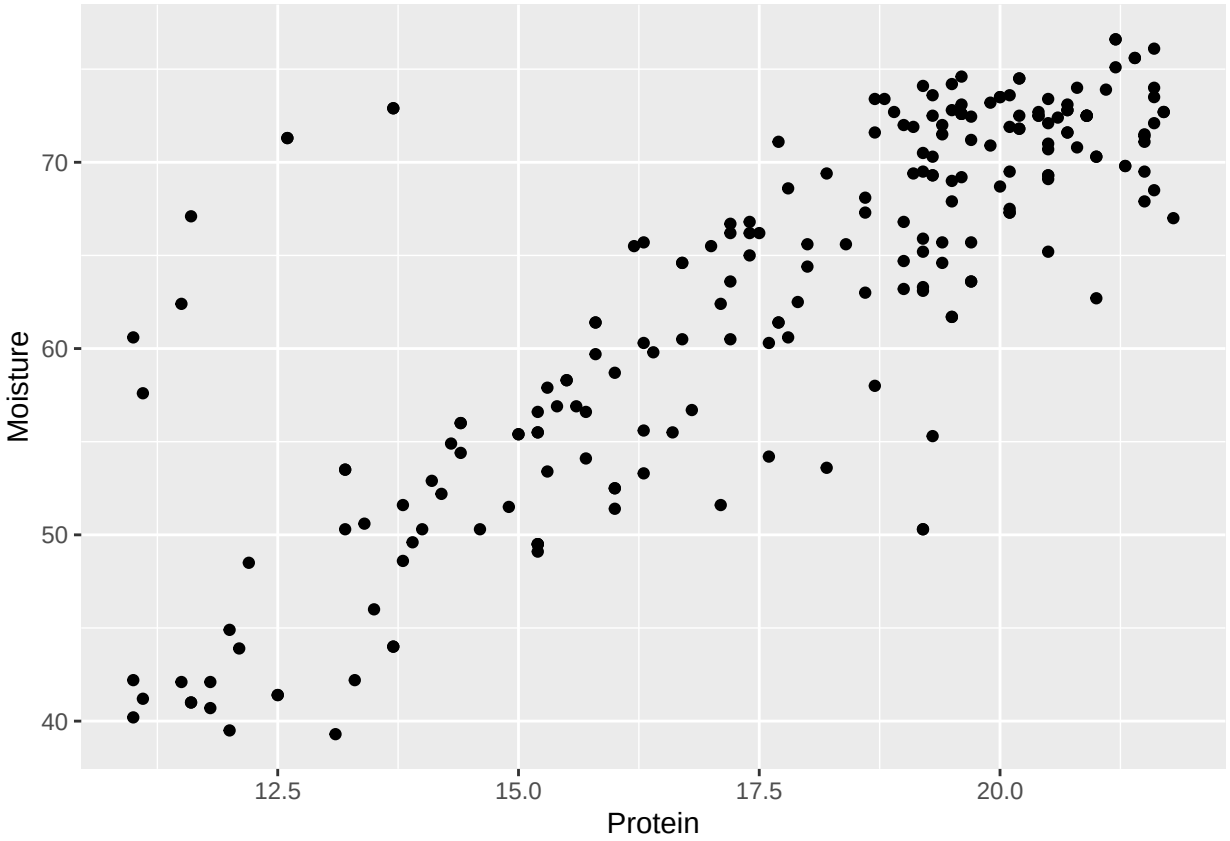

```
## $min_MSE
## [1] 56.72245
##
## $Features
## [1] "Agriculture"      "Education"        "Catholic"         "Infant.Mortality"
```

The cross validation performed on the dataset *swiss* returns the minimum MSE of *56.72245*. The features corresponding to this MSE are *Agriculture* , *Education*, *Catholic*, *Infant.Mortality*. The feature Examination had the least impact on the model and thus this feature is eliminated.

Assignment 4.1

Import data to R and create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?

The linear model for this particular data set has a correlation of 0.8145 which is very high. Thus, it can be said that the data can be described well by a linear model and the approximations can be better.

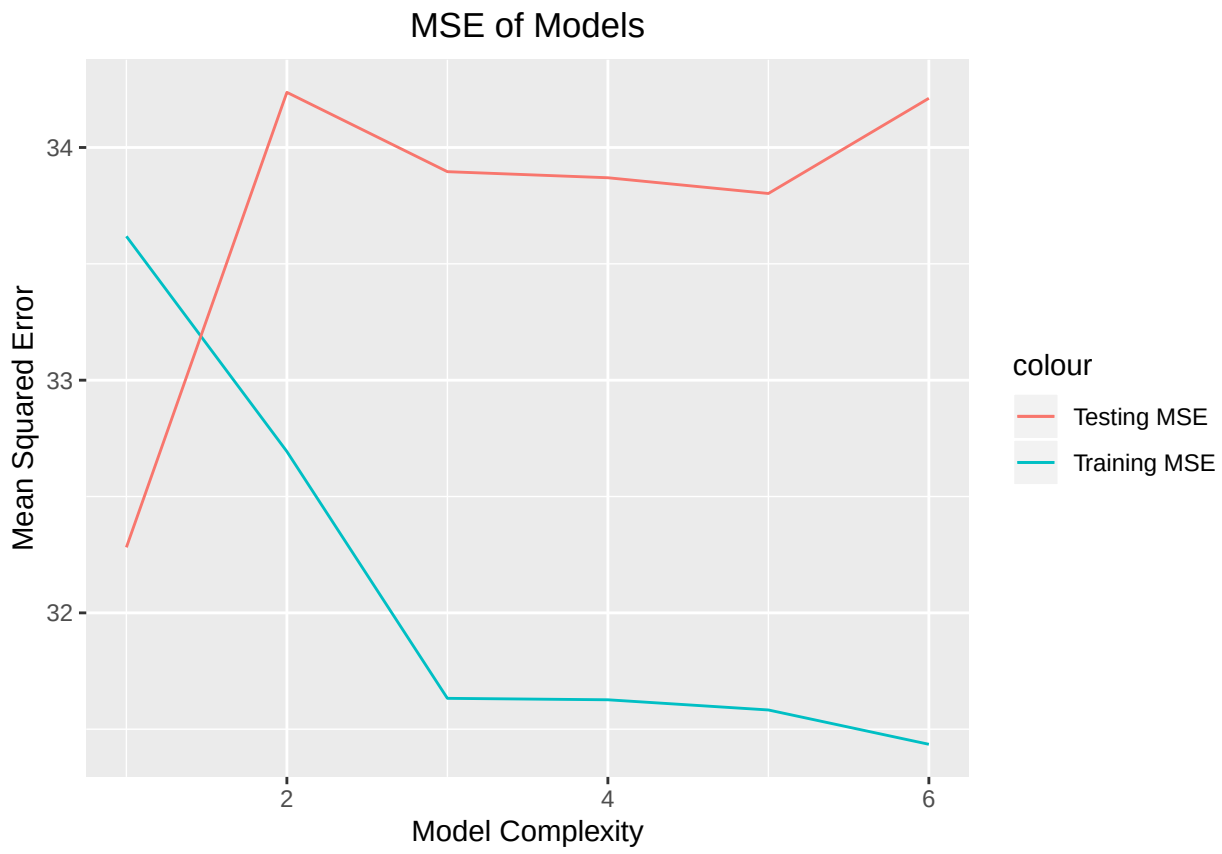


Assignment 4.2

The model with normal distribution is: $M_i \sim \ N(\beta_0 + \beta_1 prot^1 + ... + \beta_i prot^i, \sigma^2)$

Assignment 4.3

The MSE for the training model is the highest for the model M_1. However, it is constant for the models M_3 to M_5. It is lowest for the model M_6. This happens because of overfitting of data for this paricular model. In case of prediction model the MSE is highest for the model M_2. The prediction error for the model M_6 is high the coressponding training model was overfitted.



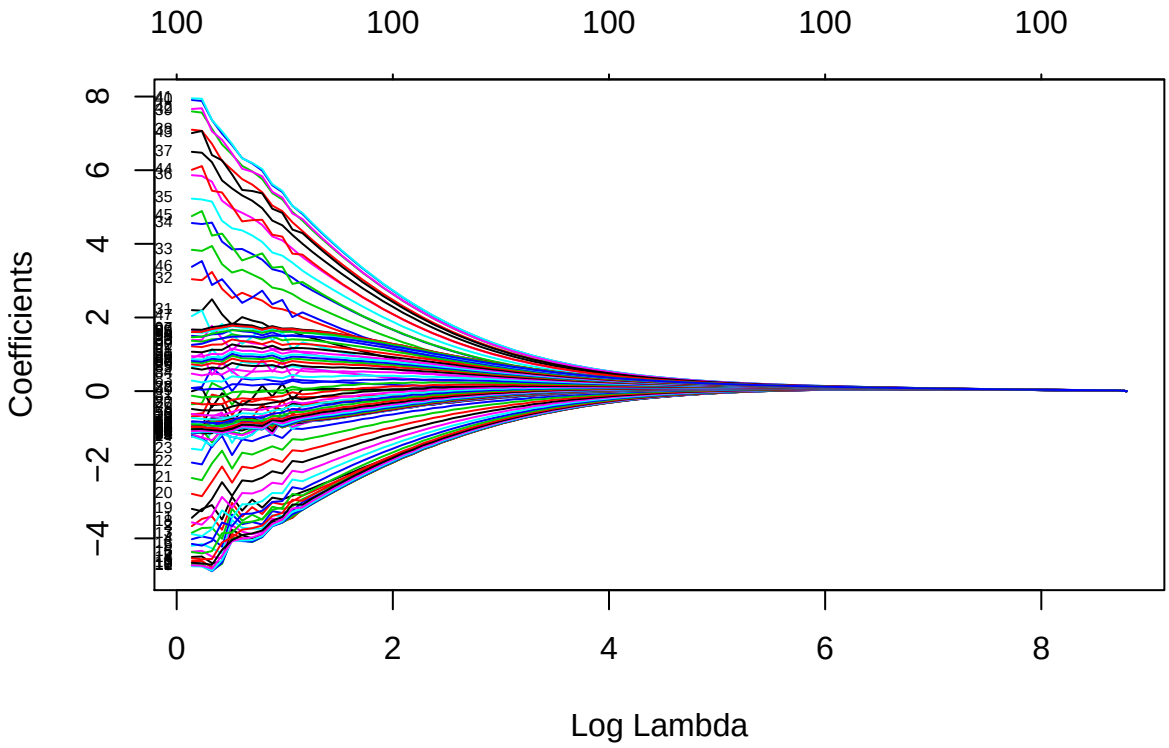
Assignment 4.4

How many variables were selected ?

In step AIC variable selection *63* variables were selected.

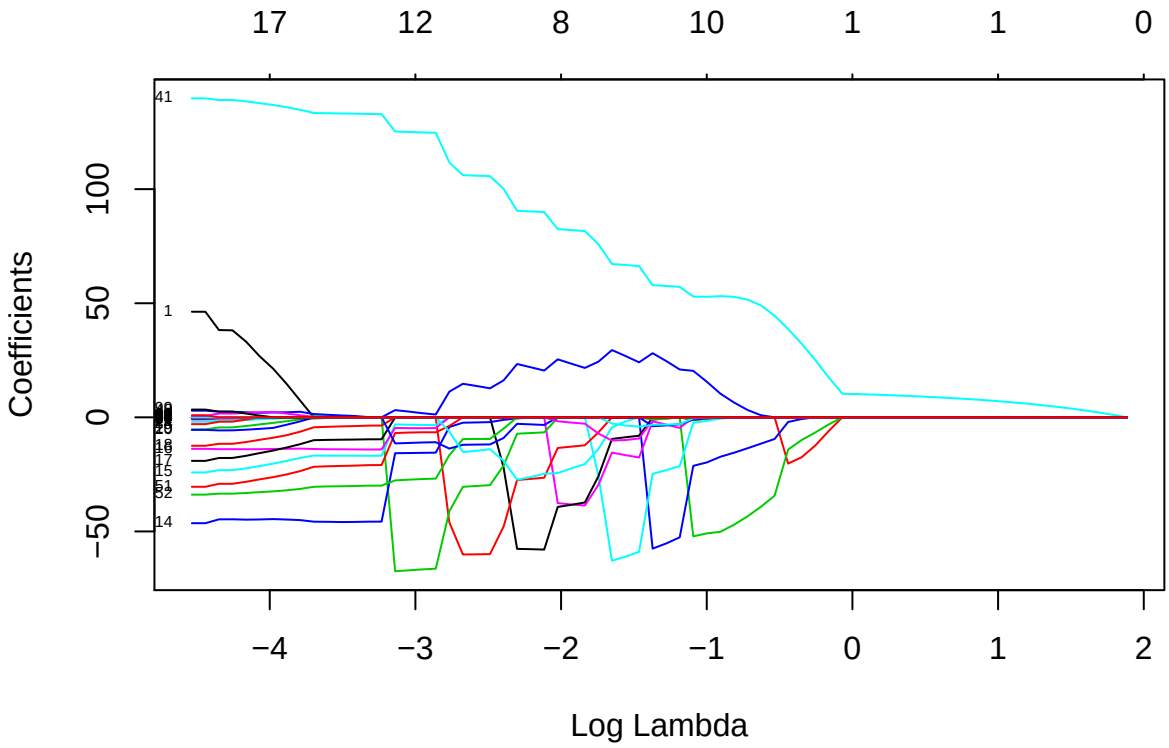
Assignment 4.5

As the value of λ increases, the coefficients shrink. However, the number of predictors are not eliminated.



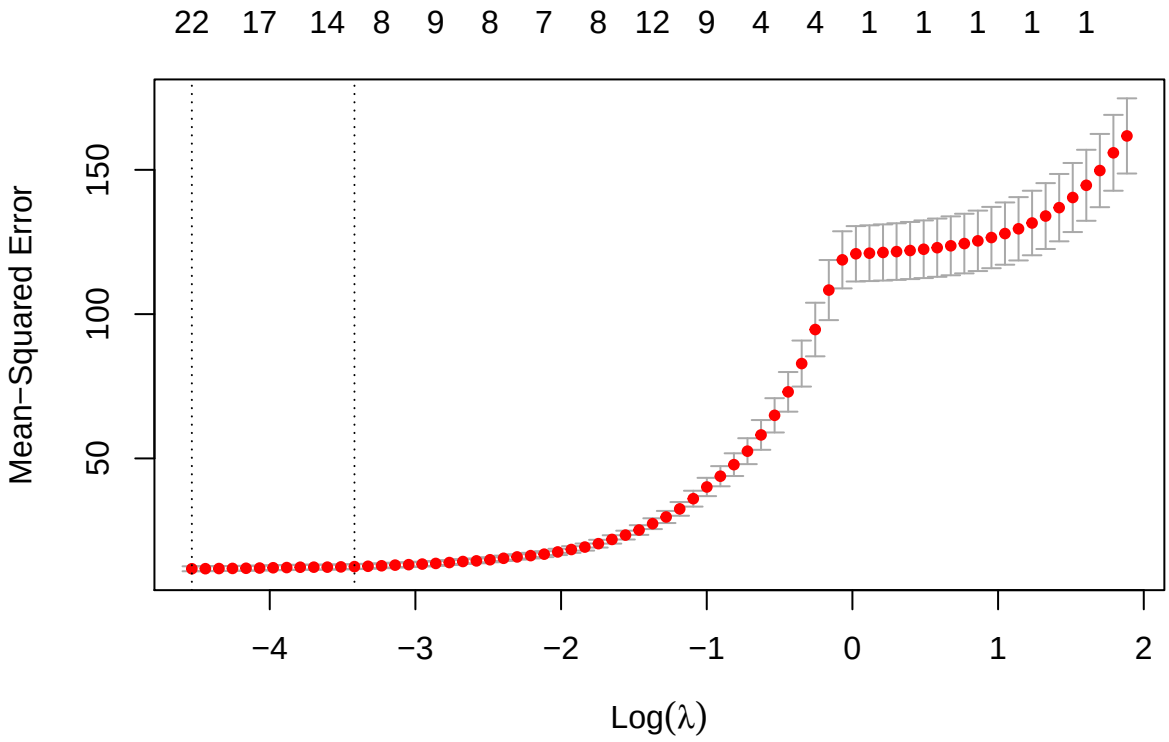
Assignment 4.6

In case of Ridge regression the shrinkage in coefficients is gradual with the increasing λ. However, the shrinkage of coefficients in case of LASSO is sudden for a particular value of λ.



Assignment 4.7

The CV scores in case of this model is the MSE which is observed to be lowest when the number of coefficients are *22*.




```
## $number_of_coefficients
## [1] 22
##
## $min_lambda
## [1] 0.01072971
##
## $min_MSE
## [1] 11.76905
```

Assignment 4.8

The coefficient selection in case of step AIC resulted in 63 variables whereas in case of the cross validation for LASSO the number of coefficients are reduced down to 22. This is due to the addition of the penalty factor λ which minimizes the effect of coefficients in LASSO model.

Appendix

```
library(readxl)
library(kknn)
library(cvTools)
library(readxl)
library(ggplot2)
data = read_excel("C:/Users/DELL PC/Desktop/Machine learning/spambase.xlsx")
library(glmnet)
#Splitting the data
```

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

#training using logistic regression
log_1 = glm(formula = Spam ~. , data = train, family = "binomial")
# Prediciton on test data
pred_1 = predict(log_1, newdata = test, type = "response")
condition_1 = ifelse(pred_1 > 0.5,1,0)
#confusion matrix for condition 1
t1 = table(train$Spam, condition_1)
misClasificError_log1 = 1-(sum(diag(t1))/length(test$Spam))
#summary(pred_1)

condition_2 = ifelse(pred_1 > 0.8,1,0)
#summary(pred_2)

#confusion matrix for condition 2
t2 = table(test$Spam, condition_2)
misClasificError_log2 = 1-(sum(diag(t2))/length(test$Spam))


# Prediciton on train data
pred_2 = predict(log_1, newdata = train, type = "response")
condition_3 = ifelse(pred_2 > 0.5,1,0)

#confusion matrix for condition 3
t3 = table(train$Spam, condition_3)
misClasificError_log3 = 1-(sum(diag(t3))/length(train$Spam))

#confusion matrix for condition 4
condition_4 = ifelse(pred_2 > 0.8,1,0)
t4 = table(train$Spam, condition_4)
misClasificError_log4 = 1-(sum(diag(t4))/length(train$Spam))




##using k-nearest neighbours
# for k = 30, test data
spam.knn = kknn(formula = Spam ~. ,train, test, k = 30)
summary(spam.knn)
fit.spam = fitted(spam.knn)
pred.knn = ifelse(fit.spam>0.5,1,0)
t.knn = table(test$Spam, pred.knn)
t1
t.knn
misClasificError = 1-(sum(diag(t.knn))/length(test$Spam))
```

```
# for k = 30, train data
spam.knn_1 = kknn(formula = Spam ~. ,train, train, k = 30)
summary(spam.knn_1)
fit.spam_1 = fitted(spam.knn_1)
pred.knn_1 = ifelse(fit.spam_1>0.5,1,0)
t.knn_1 = table(train$Spam, pred.knn_1)
t1
t.knn_1
misClasificError_1 = 1-(sum(diag(t.knn_1))/length(train$Spam))

# for k = 1, test data
spam.knn_2 = kknn(formula = Spam ~. ,train, test, k = 1)
summary(spam.knn_1)
fit.spam_2 = fitted(spam.knn_2)
pred.knn_2 = ifelse(fit.spam_2>0.5,1,0)
t.knn_2 = table(train$Spam, pred.knn_2)
t1
t.knn_2
misClasificError_2 = 1-(sum(diag(t.knn_2))/length(train$Spam))

# for k = 1, train data
spam.knn_3 = kknn(formula = Spam ~. ,train, train, k = 1)
summary(spam.knn_3)
fit.spam_3 = fitted(spam.knn_3)
pred.knn_3 = ifelse(fit.spam_3>0.5,1,0)
t.knn_3 = table(train$Spam, pred.knn_3)
t1
t.knn_3
misClasificError_3 = 1-(sum(diag(t.knn_3))/length(train$Spam))









#problem 3
#linear regression
mylin=function(X,Y, Xpred){
  X1 = (cbind(1,X))
  X1 = as.matrix(X1)
  Xpred1=cbind(1,Xpred)
  Xpred1 = as.matrix(Xpred1)
  Y = as.matrix(Y)
  #MISSING: check formulas for linear regression and compute beta
  t2 = solve(t(X1) %*% X1)
  beta = as.matrix(t2 %*% t(X1) %*% Y)
  Res=Xpred1 %*% beta
  return(Res)
}


myCV = function(X,Y,Nfolds){
```

```
n=length(Y)
p=ncol(X)
suppressWarnings(RNGkind(sample.kind = "Rounding"))
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){
            elements = floor(nrow(X)/Nfolds)
            if(k < Nfolds){
              current_fold = ((k-1)*elements + 1):(k*elements)
            } else{
              current_fold = ((k-1)*elements + 1):((k*elements) + nrow(X)%%Nfolds)
            }
            #MISSING: compute which indices should belong to current fold


            #print(current_fold)
            x_pred = X1[current_fold,]
            x_train = X1[-current_fold,]
            Yp = Y1[current_fold]


            #MISSING: implement cross-validation for model with features in "model" and iteration i.
              Ypred = mylin(x_train[,which(model==1)], Y1[-current_fold], x_pred[,which(model==1)])

            #MISSING: Get the predicted values for fold 'k', Ypred, and the original values for folf

            SSE=SSE+sum((Ypred-Yp)^2)
          }
          curr=curr+1
          MSE[curr]=SSE/n
          scaled_MSE = scale(MSE)
          Nfeat[curr]=sum(model)
          Features[[curr]]=model
```

```
            # plot(Features, MSE)
            #print(Nfeat)
            min_MSE_features = which(MSE == min(MSE))

            #print([min_MSE])
          }
  plot(MSE, type = "o", col = "blue", xaxt = "n", ann = FALSE)
  title(main = "MSE", xlab = "Features", col.main = "Green", font.main = 4)
  axis(1, at = 1:31, labels = Nfeat)
  result = list("min_MSE" = min(MSE), "Features" = names(x_train[,which(Features[[min_MSE_features]]==1
  return(result)

}

# x = swiss
# y = swiss$Fertility
# n = length(y)
# ind = sample(n,n)
# fit1=lm(Fertility~Agriculture+, data=data)
# fit2=lm(Fertility~V3+V4+V5+V6+V7, data=data)
# fit3=lm(Fertility~V3+V4+V5+V6, data=data)
# f1=cvFit(fit1, y=data$Fertility, data=data,K=10, foldType="consecutive")
# f2=cvFit(fit2, y=data$Fertility, data=data,K=10, foldType="consecutive")
# f3=cvFit(fit3, y=data$Fertility, data=data,K=10, foldType="consecutive")
# res=cvSelect(f1,f2,f3)
# plot(res)




#assignment 4.

library(readxl)
tecator <- read_excel("C:/Users/DELL PC/Desktop/Machine learning/tecator.xlsx")
View(tecator)

pro = lm(Moisture ~ Protein, tecator)

# plotting the linear model
ggplot(tecator) + geom_point(aes(x = Protein, y = Moisture), color = "black")
# Data fits in the linear model except for a few data points (min & max residuals are high).
plot(tecator$Protein, tecator$Moisture)

n1 = dim(tecator)[1]
suppressWarnings(RNGkind(sample.kind = "Rounding"))
set.seed(12345)
id_1 = sample(1:n1, floor(n1*0.5))
train_tecator = tecator[id_1,]
test_tecator  = tecator[-id_1,]
models = seq(1,6)
train_MSE = numeric(length = length(models))
pred_m_MSE = numeric(length = length(models))
#m = list()
```

```r
#resid_poly = matrix(0, nrow = 107, ncol = 6)
for (i in 1:6) {
  form = as.formula(Moisture ~ poly(Protein, i))
  m = lm(Moisture ~ poly(Protein, i), train_tecator)
  train_MSE[i] = mean(m$residuals^2)
  #train_MSE = as.data.frame(train_MSE)
  #names(train_MSE) = c("Models", "MSE")
  pred_m = predict(m, newdata = test_tecator)
  pred_m_MSE[i] = mean((test_tecator$Moisture-pred_m)^2)
  #plot(pred_m_MSE)
  MSE_data = data.frame(train_MSE, pred_m_MSE)
}

  ggplot(data = MSE_data) +
    geom_line(aes(x = models, y = train_MSE, color = "Training MSE")) +
    geom_line(aes(x = models, y = pred_m_MSE, color = "Testing MSE")) +
    ylab("Mean Squared Error") + xlab("Model Complexity") + ggtitle("MSE of Models")+
    theme(plot.title = element_text(hjust = 0.5))
  #pred_m[[i]] = list(predict(m[[i]], train_tecator))
  #esid_polynomial[,i] = matrix(train_tecator$Moisture - pred_mat[,i])
  #resid_poly[i] = list(as.train_tecator$Moisture - as.numeric(unlist(pred_m[[i]])))


unpred_m = unlist(pred_m)
View(unpred_m)
pred_mat = matrix(unpred_m, ncol = 6)
View(pred_mat)
#problem 1.4.3
 #  x_feat = cbind.data.frame(tecator$Moisture, tecator$Protein, tecator$Protein^2, tecator$Protein^3,
 #                 tecator$Protein^4, tecator$Protein^5, tecator$Protein^6)


 # feat_list = names(x_feat)
 # model = lapply(feat_list, function(x) {
 #   lm(substitute(Moisture ~ i, list(i = as.name(x))), data = x_feat)})
 # lapply(model, summary)
 #


 # x_feat1 = cbind.data.frame(tecator$Moisture, tecator$Protein)
 # names(x_feat1) = c("moisture", "protein")
 #

  #m[i] = list(lm(moisture ~ protein^i, x_feat1))

#Problem 1.4.4
library(MASS)
 dat = tecator[,-c(1,(103:104))]
fit = lm(Fat ~., data = dat)
step = stepAIC(fit, direction = "both")
step$anova
summary(step)
length(step$coefficients)-1
```

```r
#64 variables were selected

#Problem 1.4.5

train_dat = as.matrix(dat[,2:101])
test_dat = as.matrix(dat[,102])
model_1 = cv.glmnet(as.matrix(train_dat), test_dat, alpha=0, family = "gaussian")
plot(model_1)




model_ridge = glmnet(as.matrix(train_dat), test_dat, alpha=0, family = "gaussian")
plot(model_ridge, xvar = "lambda", label = TRUE)

#Problem 1.4.6

model_LASSO = glmnet(as.matrix(train_dat), test_dat, alpha=1, family = "gaussian")
plot(model_LASSO, xvar = "lambda", label = TRUE)


#problem 1.4.7


train_dat = as.matrix(dat[,2:101])
test_dat = as.matrix(dat[,102])
model_1 = cv.glmnet(as.matrix(train_dat), test_dat, alpha=1,family = "gaussian")
plot(model_1)
length(model_1$nzero)
#choosing the best model by cross validation
coefficients_LASSO = coef(model_1, s = "lambda.min")
number_coef = length(which(coefficients_LASSO != 0))
model_1$lambda.min
min(model_1$cvm)

#MSE is the least when lambda = 0.01072971
#The total number of variables selected are 23.
# CV score increases with increase in value of lambda.
```