# Lab 1 Report

*Zuxiang Li*

*11/19/2019*

## Assignment 1. Spam classification with nearest neighbors

**1. Import the data into R and divide it into training and test sets (50%/50%) by using the following code:**

```
library(openxlsx)
data<-read.xlsx("material/spambase.xlsx")
n <- dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
test <- data[-id,]
```

**2. Use logistic regression to classify the training and test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.5$, otherwise $\hat{Y} = 0$ and report the confusion matrices and the misclassification rates for train and test data. Analyze the obtained results.**

```
## [1] "Rate= 0.5 train"
##      Target
## Model   0   1
##     0 804  93
##     1 127 346
## [1] "Misclassfication Rate= 0.160583941605839"
```

```
## [1] "Rate= 0.5 test"
##      Target
## Model   0   1
##     0 808  92
##     1 143 327
## [1] "Misclassfication Rate= 0.171532846715328"
```

When the rate equal to 0.5, we got the confusion matrix of performing model with training and testing data set. The results of matrices and misclassification rate are similar.

**3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.8$, otherwise $\hat{Y} = 0$**

```
## [1] "Rate= 0.8 train"
##      Target
## Model   0   1
##     0 921 333
##     1  10 106
## [1] "Misclassfication Rate= 0.25036496350365"
```

```
## [1] "Rate= 0.8 test"
##      Target
## Model   0   1
##     0 931 314
##     1  20 105
## [1] "Misclassfication Rate= 0.243795620437956"
```

We increase the ratio to 0.8, the misclassification rate increases too.

**4. Use standard kknn() with K = 30 from package *kknn*, report the misclassification rates for the training and test data and compare the results with step 2.**

```
##      Target
## Model   0   1
##     0 779  77
##     1 152 362
```

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

```
##      Target
## Model   0   1
##     0 702 180
##     1 249 239
```

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

Compare to the results in step2, we got a worse result using kknn, the misclassification rate is 16% and 31% for training and test data.

**5. Repeat step 4 for K=1 and compare results with step 4. What effects does the decrease of K lead to and why?**

```
##      Target
## Model   0   1
##     0 931   0
##     1   0 439
```

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

```
##      Target
## Model   0   1
##     0 644 185
##     1 307 234
```

```
## [1] 0.3591241
```

We got an overfitted kknn model using k=1. For training data, the misclassification rate is 0%. As for test data, the rate is higher up to 35%. When k is a small value, it will be interfered with by errors and noises easily.

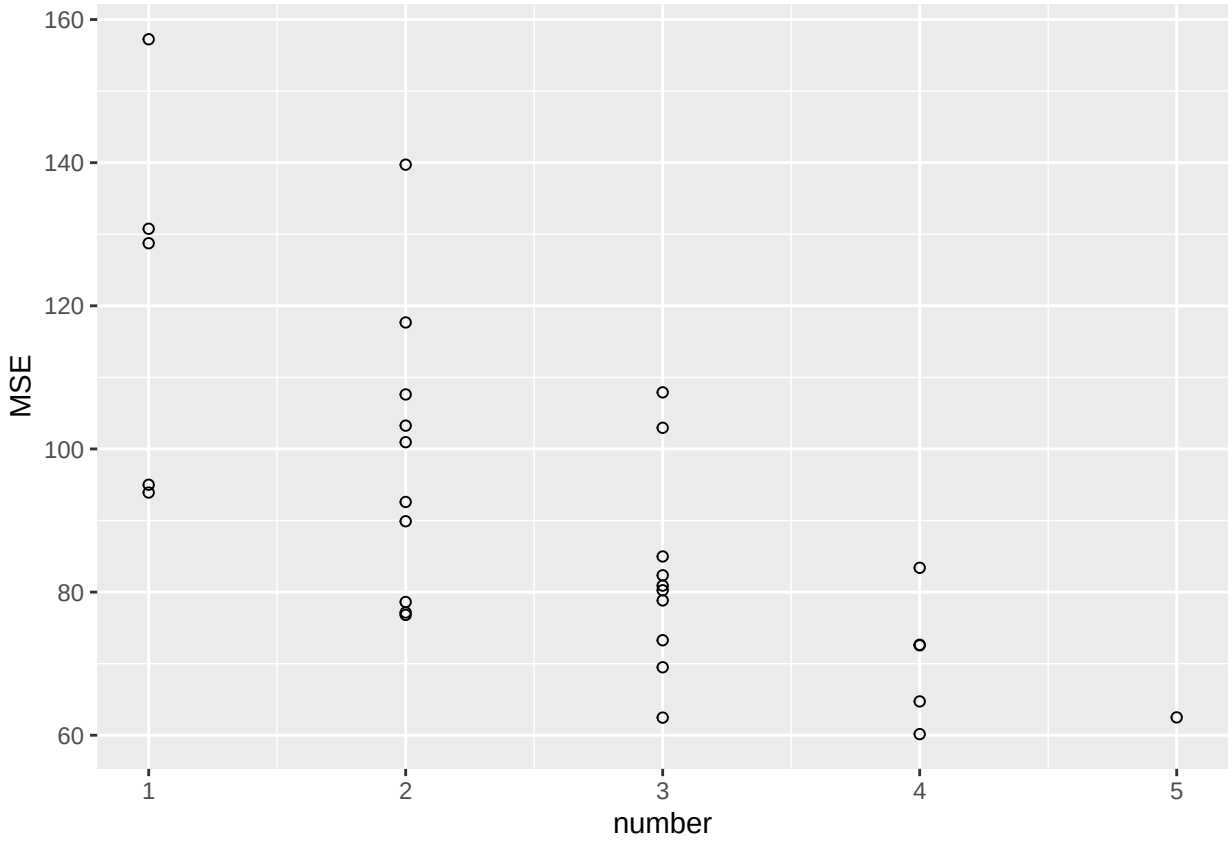
# Assignment 3. Feature selection by cross-validation in a linear model.

**2. Test your function on data set swiss available in the standard R repository:**

- Fertility should be Y
- All other variables should be X
- Nfolds should be 5

Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

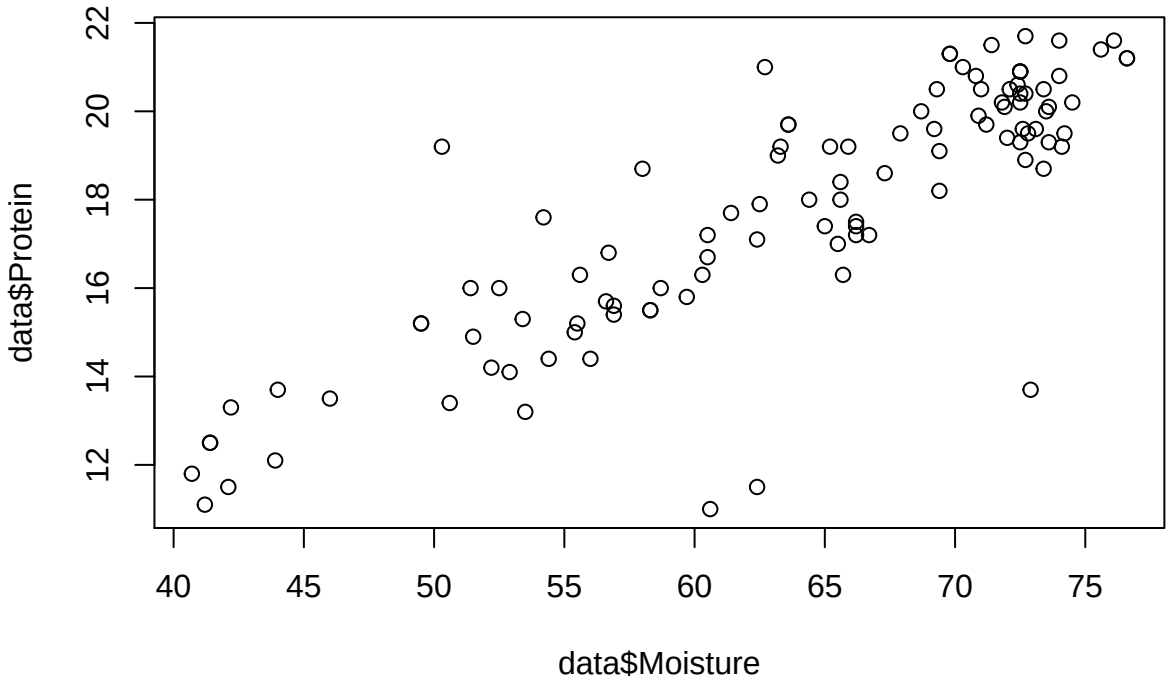
```
## $CV
## [1] 60.15763
##
## $Features
## [1] 1 0 1 1 1
##
## $plot
```




From the plot we can observe that with the number of features increases, in general, the values of MSE decreases. For the best subset, its' MSE minimized when number of features is 4. When number of features is small, model can be inaccurate, but if with too many parameters, the model will be difficult to use and interpret since it's overfitted,

# Assignment 4. Linear regression and regularization

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




Yes, the data can be approximated by a linear model.

**2. Consider model $M_i$ in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power i (i.e $M_1$ is a linear model, $M_2$ is a quadratic model and so on). Report a probabilistic model that describes $M_i$. Why is it appropriate to use MSE criterion when fitting this model to a training data?**

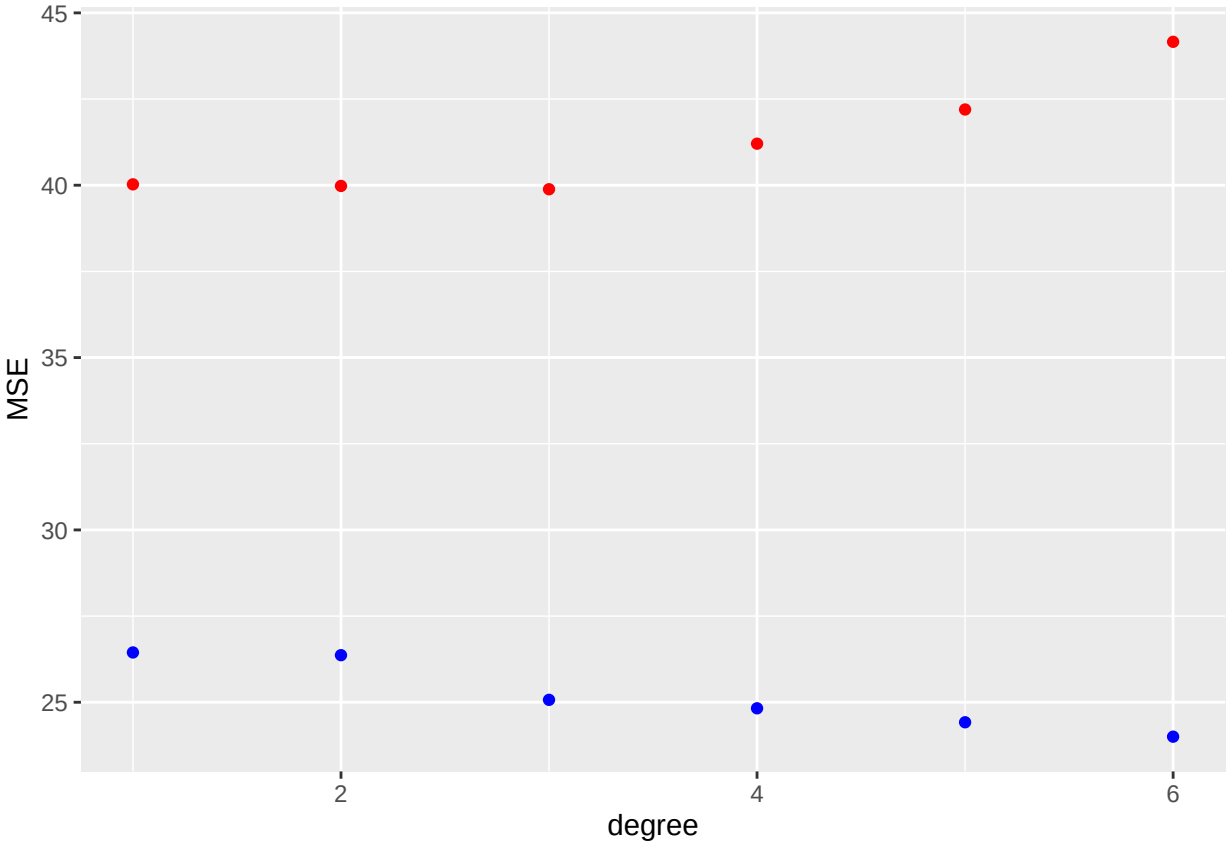
$$M_i = \sum_{j=0}^{i} \beta_j x^j + \varepsilon$$

$$i = 1, \cdots, 6$$

$$\varepsilon \sim N(\mu, \sigma^2)$$

With increasing model complexity (higher n polynomial degrees), the model tends to be overfitted, capturing noise from the training data. An overfitted model would produce inferior results (higher variance) when predicting for the test data. The MSE criterion therefore can tell how well fit (or under/overfit) the model is to the data.

**3. Divide the data (50/50) and fit models $M_i, i = 1, \cdots, 6$. For each model, record the training and validation MSE and present a plot showing how training and validation MSE depend on $i$. Which model is best according to this plot? How do MSE values change and why? Interpret this picture in bias-variance tradeoff.**




The best model is when i=3. According to the plot, when i increase, the MSE of training data begins to decrease, since the model captured the noise and becoming overfitted. The model complexity goes higher will cause low bias and high variance. If complexity goes lower will result in high bias and low variance. So we need to find a moderate value to balance the bias and variance.

**4. Perform variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC. Comment on how many variables were selected.**

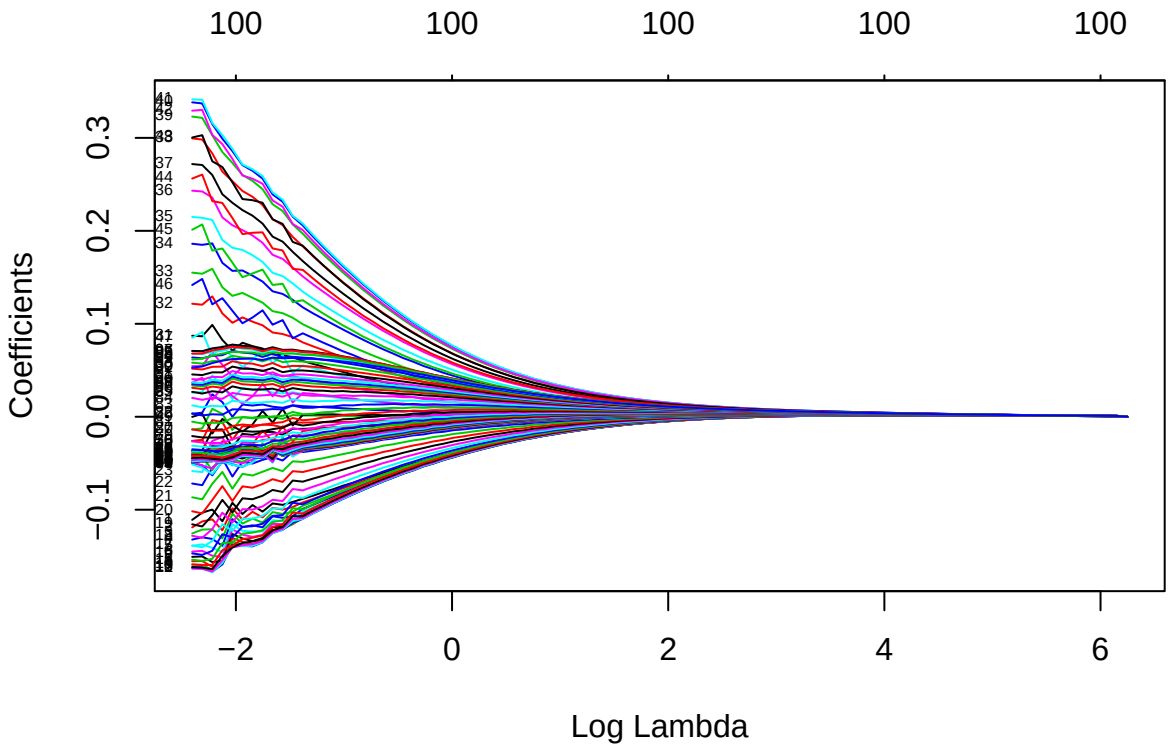
```
## [1] 63
```

Exclude the intercept we got 63 variables.

**5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor $\lambda$ and report how the coefficients change with $\lambda$**

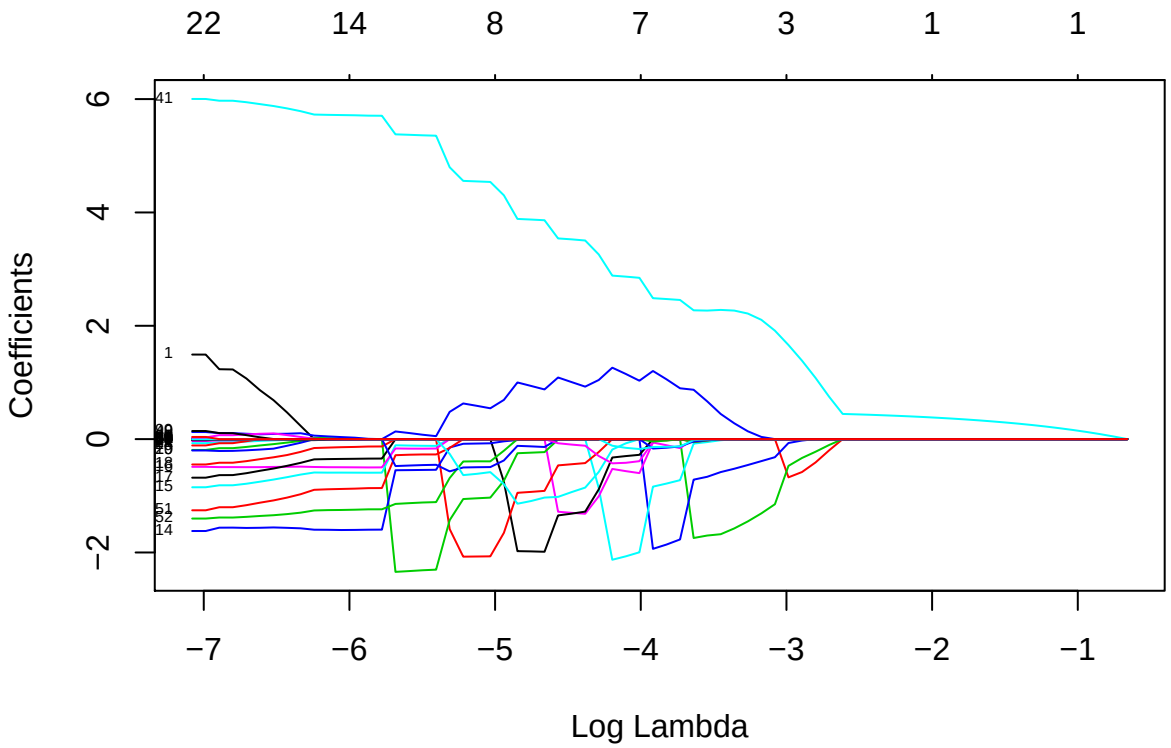
```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0
```



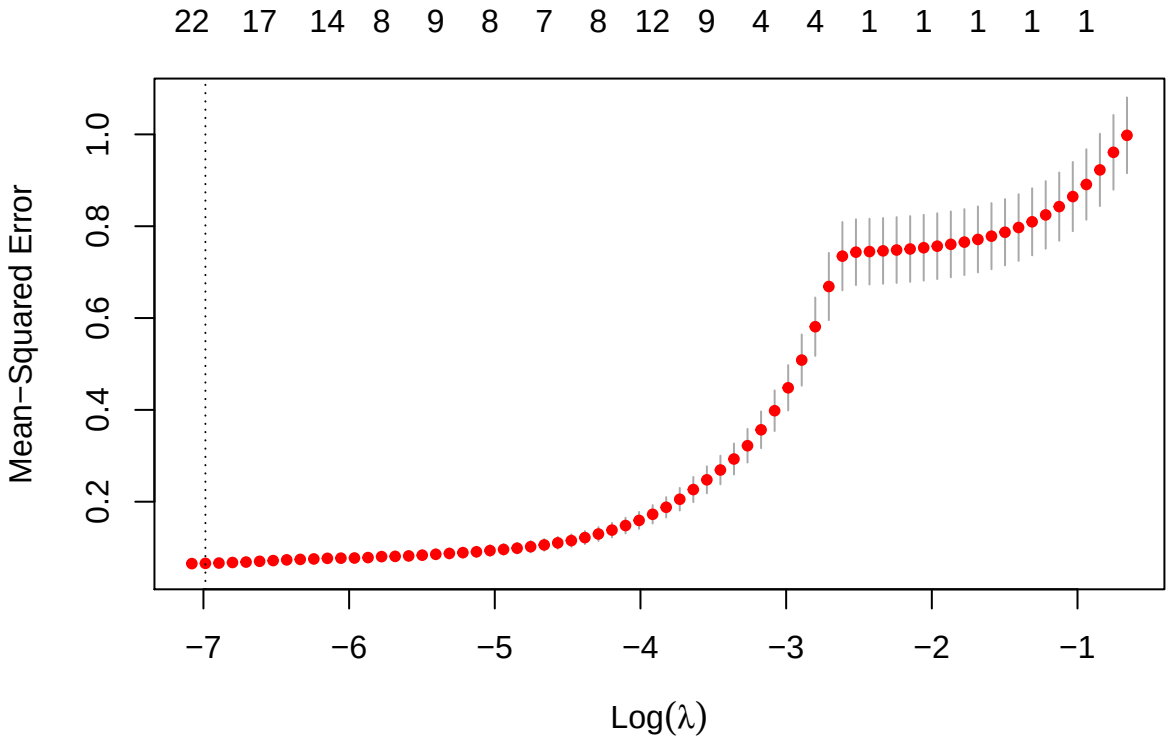

For ridge regression, we can observe from the plot that as lambda increases, all coefficients tend to be zero.

**6. Repeat step 5 but fit with LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?**




For LASSO, the coefficients converge faster.

**7. Use cross-validation to find optimal LASSO model (make sure that case $\lambda = 0$ is also considered by the procedure), report the optimal $\lambda$ and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes $\lambda$**




```
## [1] "Lambda value for best performance:"
```

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

```
##
## Call:  cv.glmnet(x = covariates, y = response, lambda = new_lambda,      alpha = 1, family = "gaussi
##
## Measure: Mean-Squared Error
##
##        Lambda Measure       SE Nonzero
## min 0.0000000 0.05826 0.007849     100
## 1se 0.0009243 0.06574 0.008853      20
```

CV score increases with $\lambda$. When $\lambda = 0$ we got the lowest MSE value with no variable dropped.

**8. Compare the results from steps 4 and 7.**

From (4), after running stepAIC, the model kept 63 variables, 36 variables were dropped. In (4), after cross-validation the LASSO model it shows that when $\lambda = 0$ we got the best performance. No variable were dropped.

# Appendix

## Assignment 1

### 1.1

```
library(openxlsx)
data<-read.xlsx("material/spambase.xlsx")
n <- dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
test <- data[-id,]
```

### 1.2

```
cft<-function(rate,test_or_train){
  mod<-glm(Spam~.,data=train,family = "binomial")
  y_pre<-predict(mod,test_or_train,type="response")
  y_pre<-as.integer(y_pre>rate)
  cft<-table(Model=y_pre,Target=test_or_train$Spam)
  print(paste("Rate=",rate,substitute(test_or_train)))
  print(cft)
  tp <- cft[2, 2]
  tn <- cft[1, 1]
  fp <- cft[2, 1]
  fn <- cft[1, 2]
  accuracy <- 1-((tp + tn)/(tp + tn + fp + fn))
  print(paste("Misclassfication Rate=",accuracy))
}
cft(0.5,train)
cft(0.5,test)
```

### 1.3

```
cft(0.8,train)
cft(0.8,test)
```

### 1.4

```
library(kknn)
r <- train.kknn(Spam~., data = train, ks = 30)
fit<-predict(r,train)
fit<-as.integer(fit>0.5)
cft<-table(Model=fit,Target=train$Spam)
cft
tp <- cft[2, 2]
tn <- cft[1, 1]
fp <- cft[2, 1]
fn <- cft[1, 2]
print(accuracy <- 1-(tp + tn)/(tp + tn + fp + fn))
```

```
fit<-predict(r,test)
fit<-as.integer(fit>0.5)
cft<-table(Model=fit,Target=test$Spam)
cft
tp <- cft[2, 2]
tn <- cft[1, 1]
fp <- cft[2, 1]
fn <- cft[1, 2]
print(accuracy <- 1-(tp + tn)/(tp + tn + fp + fn))
```

### 1.5

```
library(kknn)
r <- train.kknn(Spam~., data = train, ks = 1)
fit<-predict(r,train)
fit<-as.integer(fit>0.5)

cft<-table(Model=fit,Target=train$Spam)
cft
tp <- cft[2, 2]
tn <- cft[1, 1]
fp <- cft[2, 1]
fn <- cft[1, 2]
print(accuracy <- 1-(tp + tn)/(tp + tn + fp + fn))

fit<-predict(r,test)
fit<-as.integer(fit>0.5)
cft<-table(Model=fit,Target=test$Spam)
cft
tp <- cft[2, 2]
tn <- cft[1, 1]
fp <- cft[2, 1]
fn <- cft[1, 2]
print(accuracy <- 1-(tp + tn)/(tp + tn + fp + fn))
```

## Assignment 3

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

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){
            #MISSING: compute which indices should belong to current fold
            current_feat=which(model==1)
            #MISSING: implement cross-validation for model with features in "model" and iteration i.
            #train_X=X1[((k-1)*9+1):(k*9),]
            #train_X_Pred=train_X[,-current_feat]
            #train_Y=Y1[((k-1)*9+1):(k*9)]
            #return(mylin(train_X,train_Y,train_X_Pred))
            begin_pos=(k-1)*sF+1
            if(k==Nfolds){
              end_pos=length(Y1)
            }else{
              end_pos=k*sF
            }

            test_X=X1[begin_pos:end_pos,current_feat]

            train_X=X1[-begin_pos:-end_pos,current_feat]

            train_Y=Y1[-begin_pos:-end_pos]

            Ypred=mylin(train_X,train_Y,test_X)

            #MISSING: Get the predicted values for fold 'k', Ypred, and the original values for folf
            Yp=Y1[begin_pos:end_pos]
            SSE=SSE+sum((Ypred-Yp)^2)
          }
          curr=curr+1
          MSE[curr]=SSE/n
          Nfeat[curr]=sum(model)
          Features[[curr]]=model
        }
```

```
  #MISSING: plot MSE against number of features
  library(ggplot2)
  #df<-data.frame(,y=MSE)
  df<-data.frame(number=c(),MSE=c())
  for(i in 1:length(Features)){
    tmp=data.frame(number=sum(Features[[i]]),MSE=MSE[i])
    df=rbind(df,tmp)
  }
  plot1<-ggplot(df,aes(x=number,y=MSE))+geom_point(shape=21)
  #return(plot1)
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]],plot=plot1))
}

myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)
```

### Assignment 4

#### 4.1

```
library(openxlsx)
library(ggplot2)

data<-read.xlsx("material/tecator.xlsx")
n=dim(data)[1]
set.seed(12345)
id=sample(1:n,floor(n*0.5))
train=data[id,]
test=data[-id,]

data<-data.frame(Moisture=train$Moisture,Protein=train$Protein)
plot(data$Moisture,data$Protein)
```

#### 4.3

```
my_model<-function(data,i){
  form <- Moisture ~ poly(Protein, i)
  model<-lm(form,data)
  return(model)
}
train_MSE<-c()
for(i in 1:6){
  model<-my_model(data,i)
  y_pred<-predict(model,train)
  MSE<-mean((y_pred-train$Moisture)^2)
  train_MSE<-c(train_MSE,MSE)
}
test_MSE<-c()
for(i in 1:6){
  model<-my_model(data,i)
  y_pred<-predict(model,test)
```

```
  MSE<-mean((y_pred-test$Moisture)^2)
  test_MSE<-c(test_MSE,MSE)
}
df1<-data.frame(degree=c(1:6),MSE=train_MSE)
df2<-data.frame(degree=c(1:6),MSE=test_MSE)

#ggplot()+geom_point(df1,aes(x=degree,y=MSE),color="blue")
plot1<-ggplot(df1,aes(x=degree,y=MSE))+geom_point(color="blue")
plot1<-plot1+geom_point(df2,mapping=aes(x=degree,y=MSE),color="red")
plot1
```

### 4.4

```
library(MASS)
data<-read.xlsx("material/tecator.xlsx")
n_data<-data.frame(Fat=data$Fat)
n_data<-cbind(n_data,data[,2:101])
fit<-lm(Fat~.,data=n_data)
step<-stepAIC(fit,direction = "both",trace = 0)
length(coef(step))-1
```

### 4.5

```
library(glmnet)
covariates=n_data[,-1]
response=n_data[,1]
covariates=scale(covariates)
response=scale(response)
model0=glmnet(covariates, response, alpha=0,family="gaussian")
plot(model0, xvar="lambda", label=TRUE)
```

### 4.6

```
model1=glmnet(covariates, response, alpha=1,family="gaussian")
plot(model1, xvar="lambda", label=TRUE)
```

### 4.7

```
new_lambda<-c(model1$lambda,0)
mod_lasso=cv.glmnet(covariates, response, alpha=1,family="gaussian",lambda = new_lambda)
plot(mod_lasso)
print("Lambda value for best performance:")
mod_lasso$lambda.min

mod_lasso
```