Lab 1

Bjorn Hansen, Erik Anders, Ahmed Alhasan

12/18/2019

Assignment 1.Spam classification with nearest neighbors

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     803  142
##   Spam         81  344
##
## $`Missclassification Rate`
## [1] 0.1627737
```

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     791  146
##   Spam         97  336
##
## $`Missclassification Rate`
## [1] 0.1773723
```

Analysis:

- The missclassification rate for the training set is a little less because of overfitting, the model already have seen the train data and it should not have been tested with it.
- False-Positive Rate, $\frac{FP}{FP+TN}$ (classified as spam where it actually not) is about 15.58% (146/937) and True-Positive Rate $\frac{TP}{TP+FN} = 77.60\%$ (336/433), which could be considered as bad rates in the case of a spam filter because we might filter out important emails as spam.

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     941    4
##   Spam        335   90
##
## $`Missclassification Rate`
## [1] 0.2474453
```

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     926   11
##   Spam        367   66
```

```
##
## $`Missclassification Rate`
## [1] 0.2759124
```

- Missclassification rate has increased when 0.8 threshold been used, this dramatically decreased the False-Postive and increased the False-Negative, meaning the model now is less predictive and more foregiving towards spam but much less normal emails get filtered as spam.

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     807  138
##   Spam         98  327
##
## $`Missclassification Rate`
## [1] 0.1722628
```

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     672  265
##   Spam        187  246
##
## $`Missclassification Rate`
## [1] 0.329927
```

- With 32.9% miss_rate on the test dataset the k-nearest neighbor perform much worse than the 17.7% miss_rate from the logistic regression.
- There is understandable big gap in the miss_rate between the train and test datasets because the model was trained on the train data and have seen it.
- False-Positive Rate and True-Positive Rate seems to be much worse than the ones obtained from logistic regression.

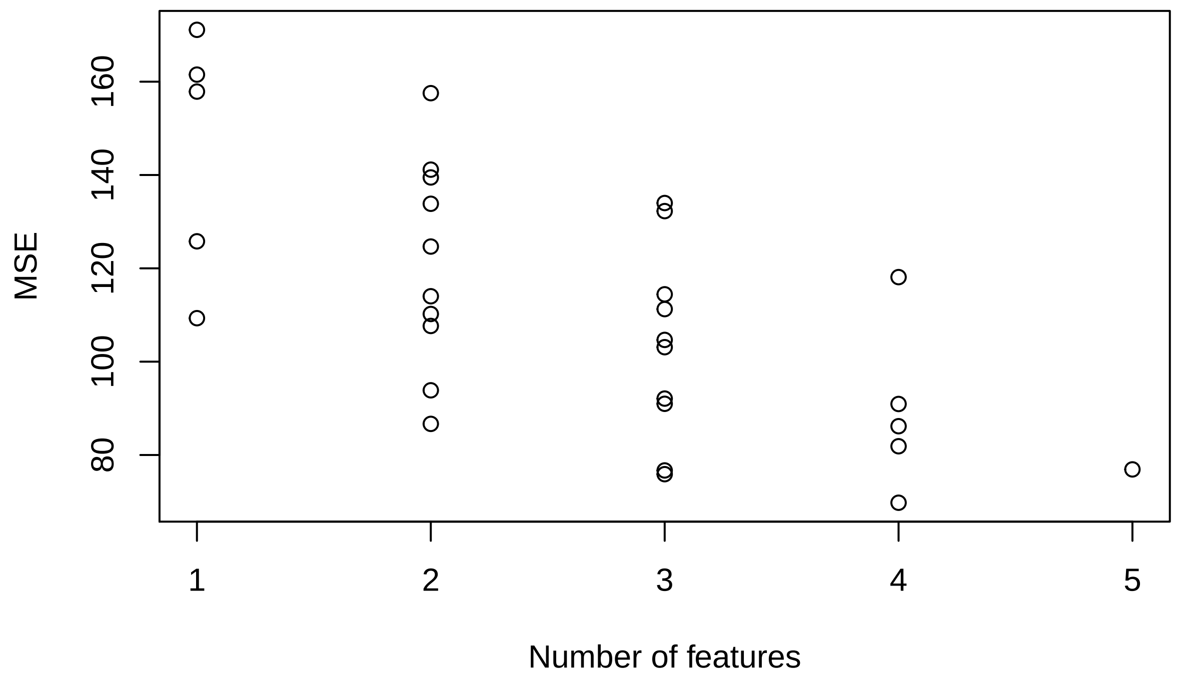
```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     945    0
##   Spam          0  425
##
## $`Missclassification Rate`
## [1] 0
```

```
## $`Confusion Matrix`
##          Predicted
## Actual    No Spam Spam
##   No Spam     640  297
##   Spam        177  256
```

```
##
## $`Missclassification Rate`
## [1] 0.3459854
```

- Reducing k to 1 on the training dataset will always make 0% missclassification rate because nearest neighbor will be the point itself, and the model will predict the exact points that it was trained on.
- Using k=1 on the testing dataset has increased the miss_rate to about 36%, this is because of extreme overfitting where every point from the trainging dataset have very small area that doesnt allow much predictiblity for the unseen testing dataset.

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

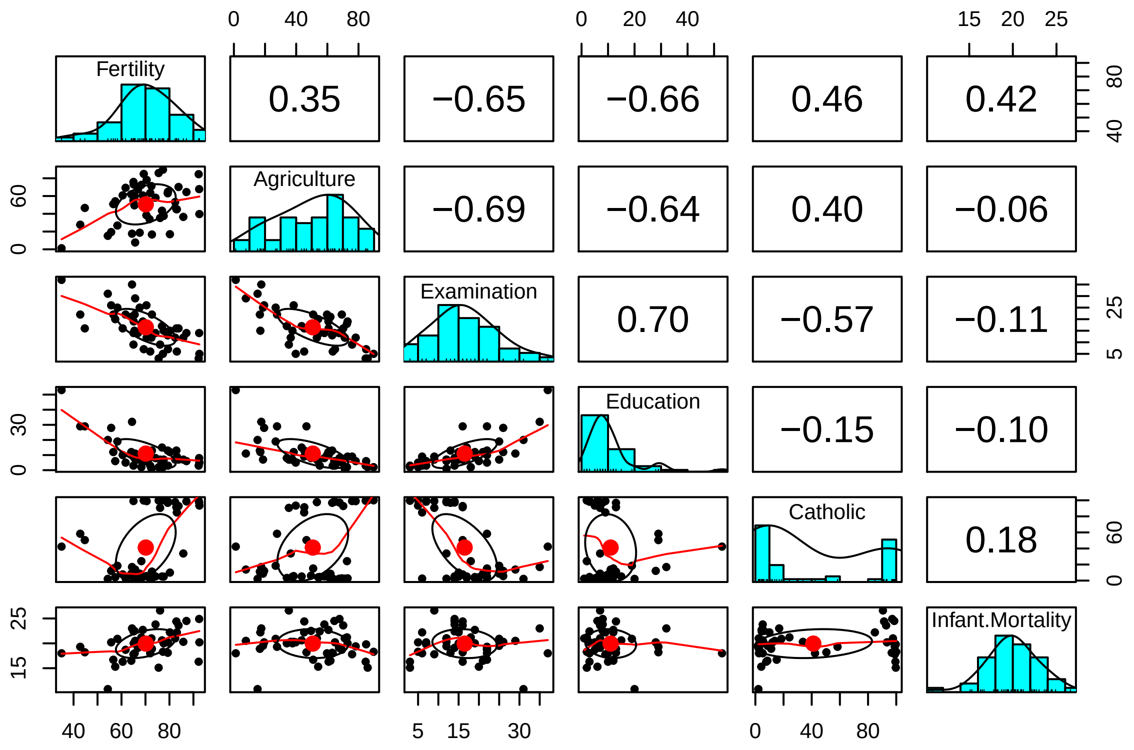


```
## $CV
## [1] 69.77112
##
## $Features
## [1] 1 0 1 1 1
```

Analysis:

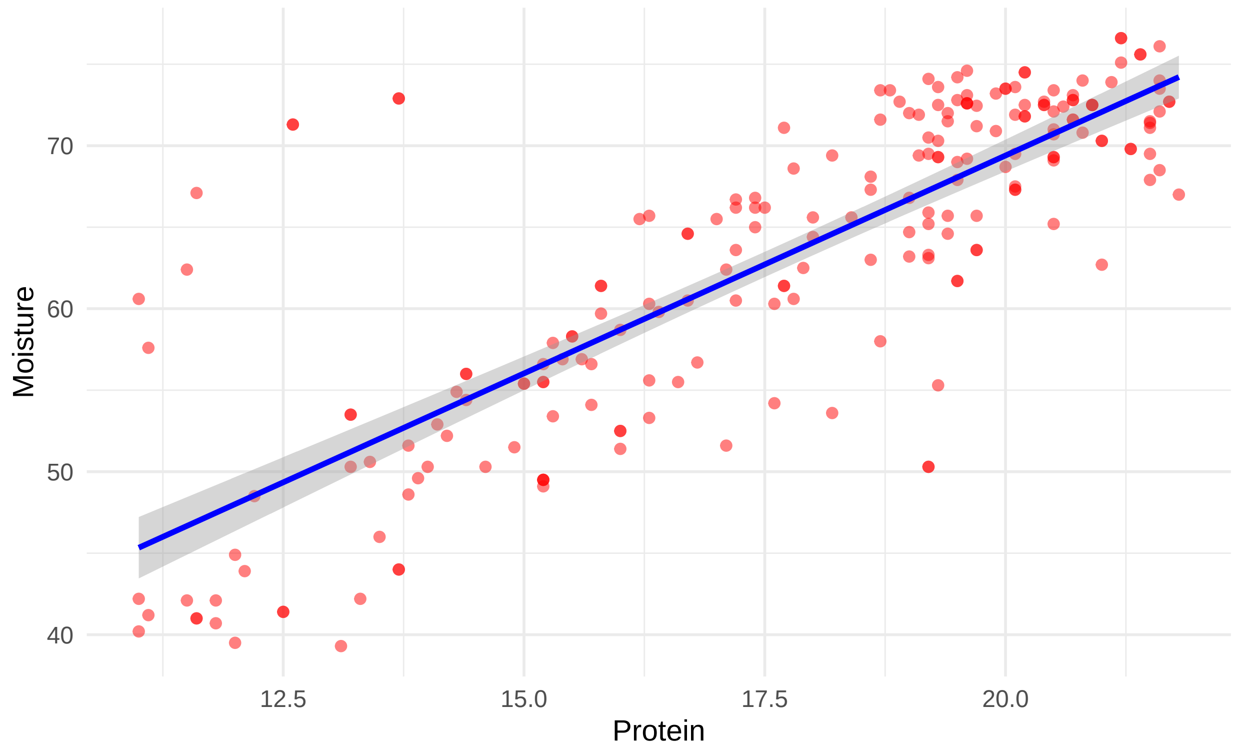
- From the plot the cross-validation shows that accuracy of prediction is increasing with the number of features and the error at its minimum when 4 features are selected (Agriculture, Education, Catholic

& Infant.Mortality), using all 5 features is a little worse but closely accurate, probably using more than 4 features start to show overfitting.



- However, from the single correlation between each of the features with Fertility it is not clear why the cross-validation have selected these 4 features (Agriculture, Education, Catholic & Infant.Mortality) and dropped (Examination) as the best selection with least MSE, since Examination is more correlated (negative correlation) with Fertility than Agriculture or Infant.Mortality.

Assignment 4. Linear regression and regularization



Analysis:

- Yes, there is a clear linear relationship between Moisture and Protein.

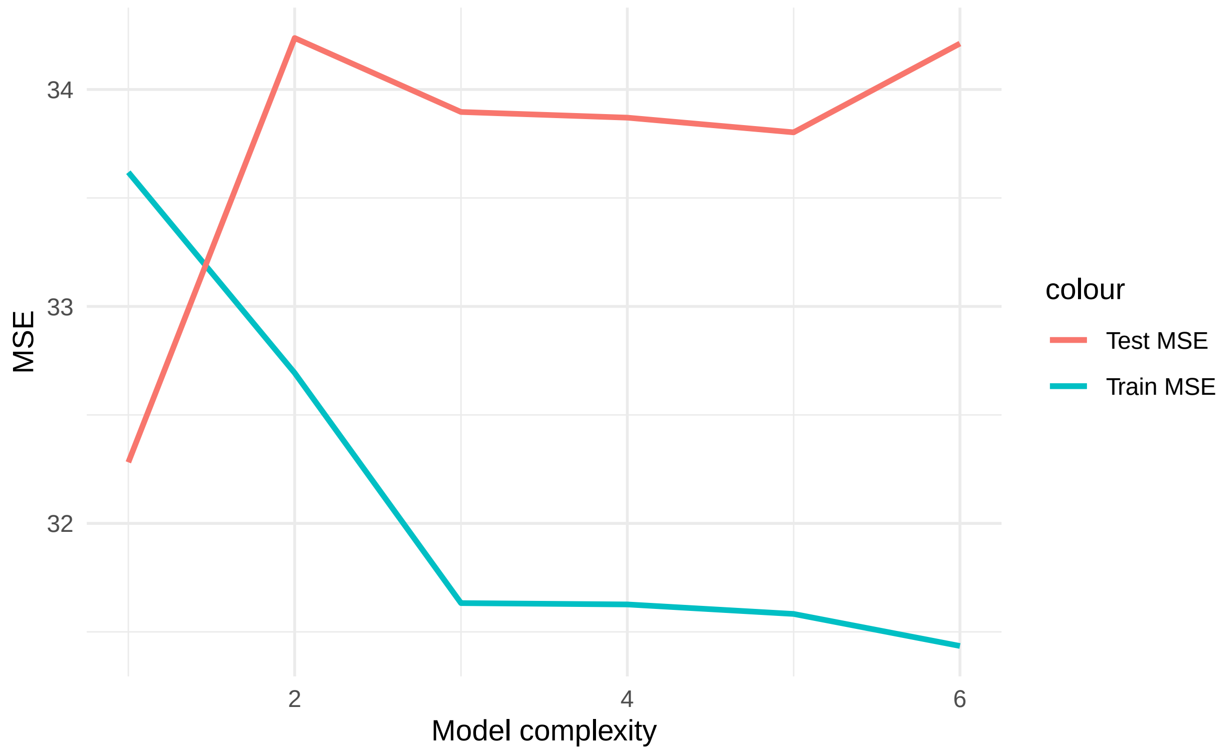
- Probabilistic model :

$$M_i = \beta_0 + \sum_{k=1}^{i} \beta_k x^k + \varepsilon_i$$

$$\varepsilon_i \sim N(0, \sigma^2)$$

- Since the error is normally distributed, MSE is the Maximum Likelihood Estimator for a normal distribution.

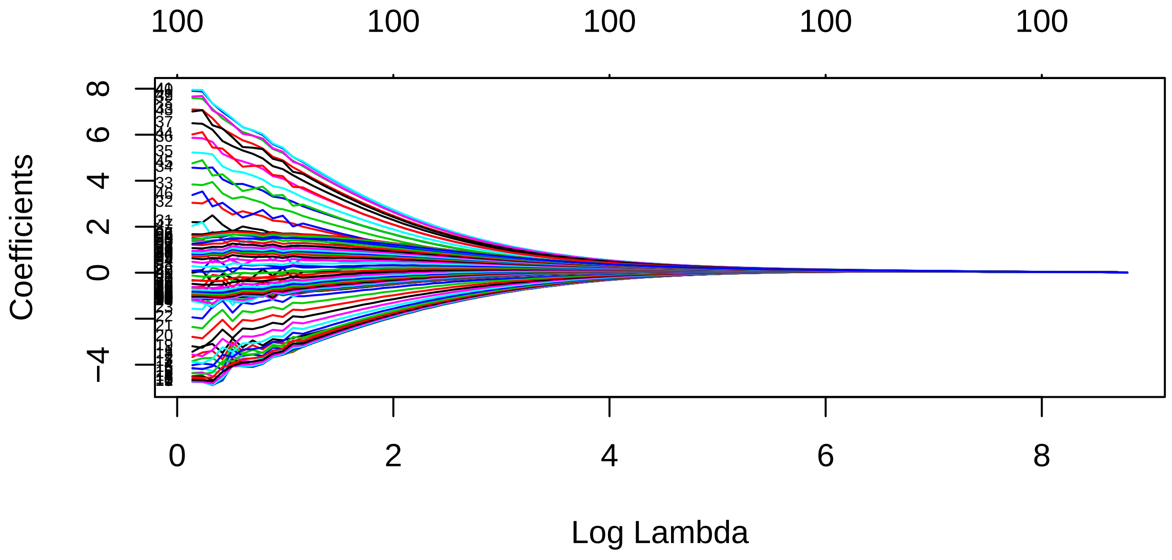
```
##   train_MSE test_MSE
## 1     33.62    32.28
## 2     32.69    34.24
## 3     31.63    33.90
## 4     31.63    33.87
## 5     31.58    33.80
## 6     31.44    34.21
```



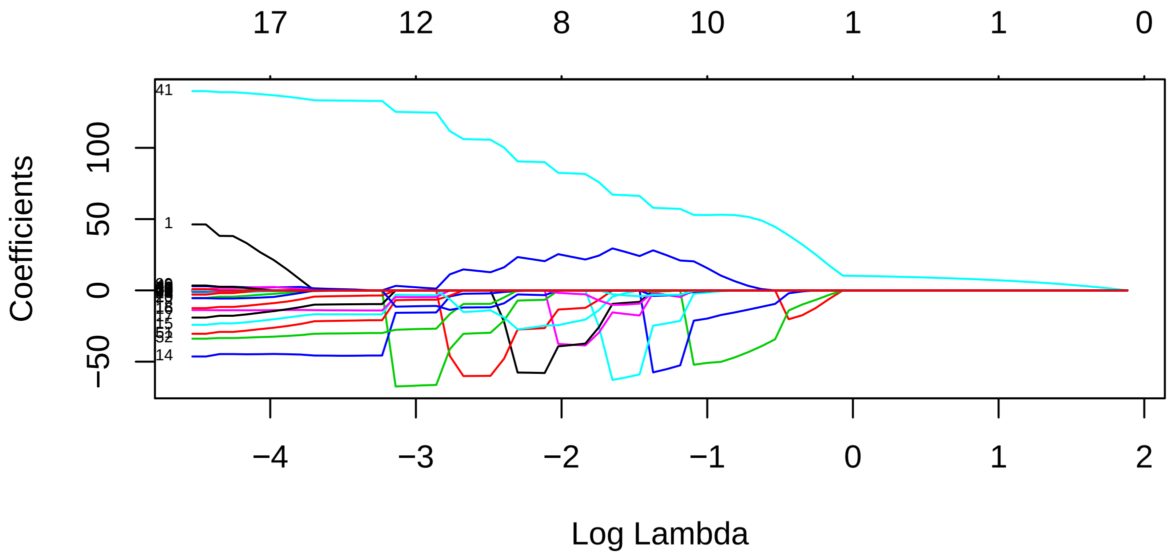
- The model with i = 1 "linear with x" is the best model because it have the least test error and the most simple.
- MSE values always decrease when increasing the complexity of the model and testing it with the training data set because overfitting and using train data as test data.
- Because $MSE = Var(\hat{y}) + [Bias(\hat{y})]^2 + Var(\epsilon)$ and because that variance is inherently a nonnegative quantity, and squared bias is also nonnegative. Hence, we see that the expected test MSE can never lie below $Var(\epsilon)$, the irreducible error. Therefore, bias & variance are competetive meaning if it bias minimized variance will increase and vice versa.

```
##                     [,1]
## Selected Channels   64
```

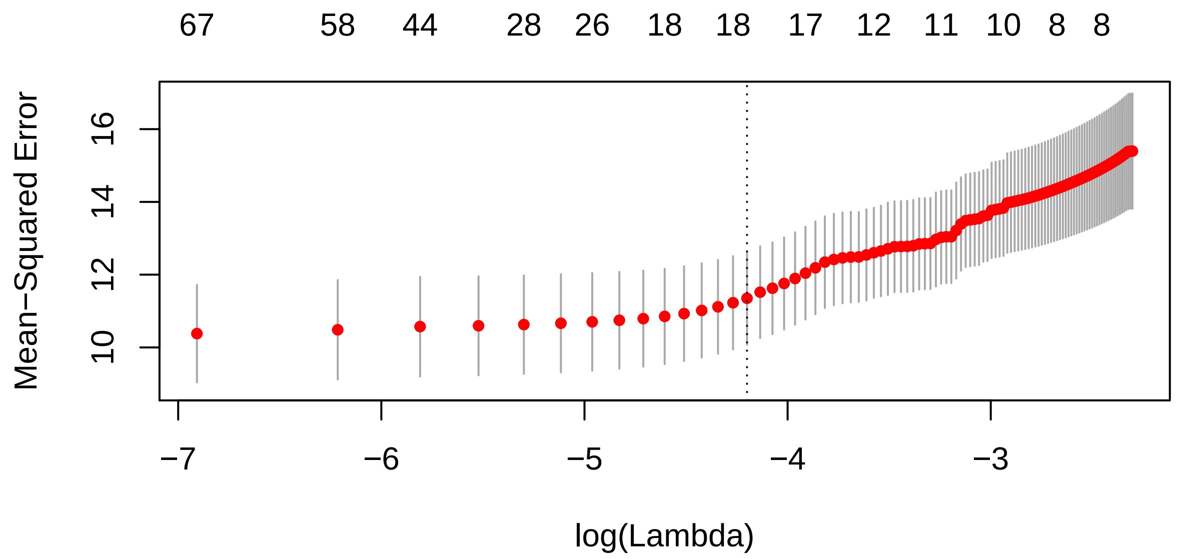
- When all data is used 64 channels where selected as the best model to predict Fat. stepAIC,
- A backward step wise selection (default) starts with the most complicated model (a model with all the 100 variables) and drops the least useful predictors
- The model with the minimum SSE is selected (the one with 64 channels in this case), however this might not be the ideal model because how stepAIC works it could drop a variable in the begining that could be part of the ideal model.



- Lambda adds a penalty to the coefficients "excpet β_0", so the higher lambda is, the least significant variables will get close to 0.



- With Lasso, the least correlated variables will get eliminated completely (set to 0), until lambda reaches a value where all coefficients are eliminated except for the intercept. In this case, from the 100 variables (channels) we started with, channel# 41 was the last to get eliminated.



```
##                   [,1]
## Minimum Lambda     0

##               [,1]
## 1se Lambda 0.015
```

- 0 is the minimum lambda that gives the minimum mean cross-validated error(for this particular C.V.). Which is normal linear regression without penalty.
- All the 100 variables have been selected with the minimum lambda, and as log of lambda grows larger the MSE also increases, however by taking the lambda within one standard error of the minimum, only 18 variables will be selected, which is much less than what was selected by stepAIC in step 4.

APPENDIX

```
spambase = read.csv("Data/spambase.csv", header = TRUE)

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

logistic <- function(data, p) {

  model <- glm(as.factor(Spam) ~.,family=binomial, data=train)
```

```r
  # type "response" is used to give the probabilities instead of log-odds
  predicted  <- predict(model, data, type = 'response')
  classified <- ifelse(predicted > p,1,0)

  con_matrix <- table(Predicted = classified, Actual = data$Spam)

  miss_rate  <- 1- sum(diag(con_matrix))/sum(con_matrix)
  #miss_rate <- mean(classified != data$Spam)

  result = list("Confusion Matrix" = con_matrix, "Missclassification Rate" = miss_rate)
  result
}

logistic(train, 0.5)
logistic(test, 0.5)
logistic(train, 0.8)
logistic(test, 0.8)


library(kknn)
knn <- function(data, k){

  classified <- kknn(formula = as.factor(Spam) ~ .,
                     train = train,
                     test = data,
                     k = k)

  con_matrix <- table(Predicted = classified$fitted.values, Actual = data$Spam)

  miss_rate <- 1- sum(diag(con_matrix))/sum(con_matrix)

  result = list("Confusion Matrix" = con_matrix,
                "Missclassification Rate" = miss_rate)
  return(result)
}

knn(train, 30)
knn(test, 30)
knn(train, 1)
knn(test, 1)



### Assignment 3

data("swiss")
library(ggplot2)

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

myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  set.seed(12345)
  ind=sample(n,n)       # Shuffle
  X1=X[ind,]            # New X after shuffle
  Y1=Y[ind]             # New Y after shuffle
  sF=floor(n/Nfolds)    # Fold size
  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)                # Selected Features
            if (sum(model)==0) next()               # Skips the first iteration where the model have no
            SSE=0
            selected_feature <- which(model == 1)
            for (k in 1:Nfolds){
              breaks <- seq(1,n,sF)
              folds <- ind[breaks[k]:breaks[k+1]-1]

              Xtrain <- X1[-folds,selected_feature]
              Xpred  <- X1[folds,selected_feature]
              Ytrain <- Y1[-folds]
              Ypred  <- Y1[folds]

              Yp <- mylin(Xtrain,Ytrain,Xpred)
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model

          }

  plot(Nfeat, MSE, xlab = "Number of features", ylab = "MSE")

  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}

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

```r
library(psych)
pairs.panels(swiss)



##### Assignment 4
library(ggplot2)
library(glmnet)
library(MASS)

tecator <- read.csv("Data/tecator.csv", header = TRUE)

ggplot(tecator, aes(x = Protein, y = Moisture)) +
  geom_point(alpha = 0.5, color = 'red') +
  geom_smooth(method = "lm", color = "blue")

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

M <- function(data,power) {
  MSE <- matrix(0,power,1)

  for (i in 1:power) {
    model <- lm(Moisture ~ poly(Protein,i), data = train)
    pred  <- predict(model, data)
    MSE[i,] <- mean((data$Moisture - pred)^2)
    }

  return(MSE)
}
train_MSE <- M(train,6)
test_MSE  <- M(test,6)


df = data.frame(train_MSE,test_MSE)

round(df,2)

ggplot(df) +
  geom_line(aes(x = 1:6, y = train_MSE, color = "train_MSE"), size = 1) +
  geom_line(aes(x = 1:6, y = test_MSE, color = "test_MSE"), size = 1) +
  ylab("MSE") +
  xlab("Model complexity")

#Using stepAIC on all data
channels_all_data   <- tecator[,2:101]
Fat_model_all_data  <- lm(tecator$Fat ~ ., channels_all_data)
best_fit_all_data   <- stepAIC(Fat_model_all_data, trace=FALSE)

as.matrix(c("Selected Channels"= length(best_fit_all_data$coefficients)))
```

```r
#Using stepAIC on training data only
channels_train_data   <- train[,2:101]
Fat_model_train_data  <- lm(train$Fat ~ ., channels_train_data)
best_fit_train_data   <- stepAIC(Fat_model_train_data, trace=FALSE)

as.matrix(c("Selected Channels"= length(best_fit_train_data$coefficients)))

#Testing stepAIC model on the test data
AIC_model  <- lm(formula(best_fit_train_data), channels_train_data)
pred  <- predict(AIC_model, test)
MSE   <- as.matrix(c("MSE" = mean((test$Fat - pred)^2)))
round(MSE,2)


ridge <- glmnet(x = as.matrix(tecator[,2:101]),
                y = as.matrix(tecator$Fat),
                alpha = 0,
                family = "gaussian")

plot(ridge, xvar="lambda", label=TRUE)

lasso <- glmnet(x = as.matrix(tecator[,2:101]),
                y = as.matrix(tecator$Fat),
                alpha = 1,
                family = "gaussian")

plot(lasso, xvar="lambda", label=TRUE)

set.seed(12345))
lasso_cv         <- cv.glmnet(x = as.matrix(tecator[,2:101]),
                        y = as.matrix(tecator$Fat),
                        alpha=1,
                        family="gaussian",
                        lambda = 0:100 * 0.001)

plot(lasso_cv)
as.matrix(c("Optimum Lambda" = lasso_cv$lambda.min))
as.matrix(c("Optimum Lambda" = lasso_cv$lambda.1se))
```