Lab 1
Elias Alesand
elial148

Changes

When extracting data from the distance matrix i previously used the rows instead of columns. This resulted in a classifier that was not useful. Resulting changes in the report and code are marked with green.

Assignment 1

Questions 3 and 4.

Confusion matrix for dataset *train*, K=5:

true \ predicted	0	1
0	787	158
1	119	306

Misclassification rate = 277/1370=0.202

Confusion matrix for dataset *train*, K=1

true \ predicted	0	1
0	939	6
1	2	423

Misclassification rate = 8/1370=0.006

Confusion matrix for dataset *test*, K=5

test_classes \ classification	0	1
0	695	242
1	193	240

Misclassification rate = 435/1370=0.318

Confusion matrix for dataset *test*, K=1

test_classes \ classification	0	1
0	639	298
1	178	255

Misclassification rate = 476/1370=0.347

I see that for the train data set the classifier performs better, much better in fact for K=1. This makes sense since this is the set that trains the classifier. For the test dataset the classifier performs a bit worse for K=1.

Question 5.
The kknn classifier gives the following results:

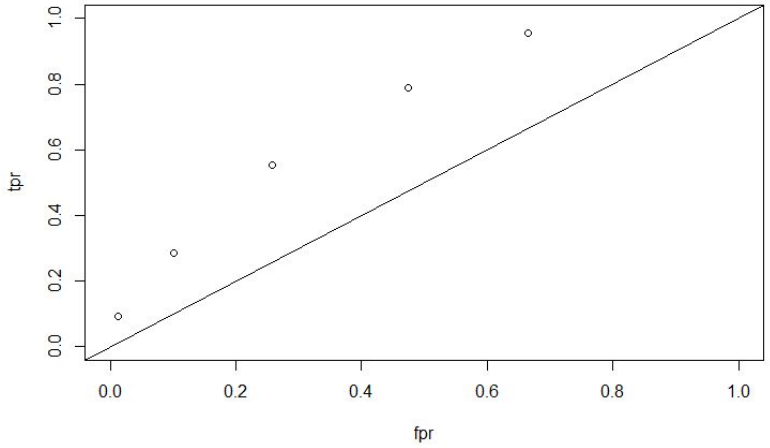
Confusion matrix for *test* with kknn(), K=5:

	predicted	
test_classes	0	1
0	640	297
1	177	256

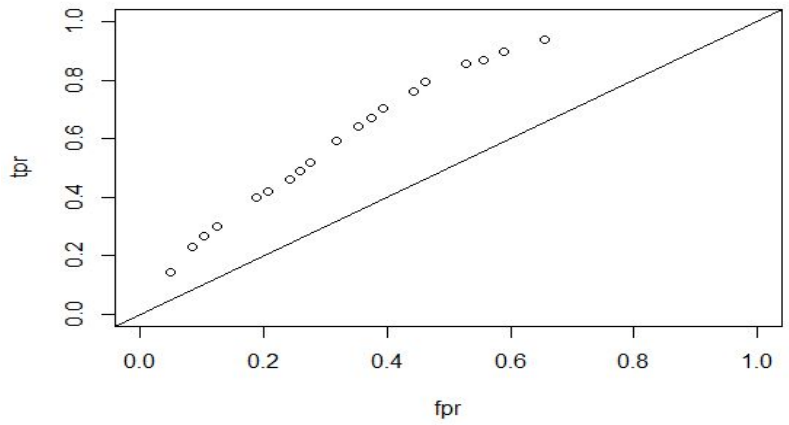
Misclassification rate = 474/1370=0.346

The kknn() classifier classifies spam more correctly but it is slightly worse at classifying non-spam correctly.

Question 6.
Roc curve for my classifier, K=5



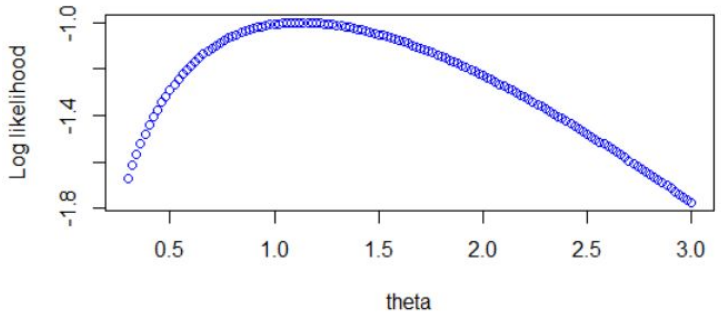
Roc curve for kknn() classifier, K=5



From the ROC-curves i see that the kknn() classifier is above the diagonal line which means that it performs quite good. The ROC curve for my classifier is also above the diagonal line which is good. The two classifiers looks to have very similar performance.

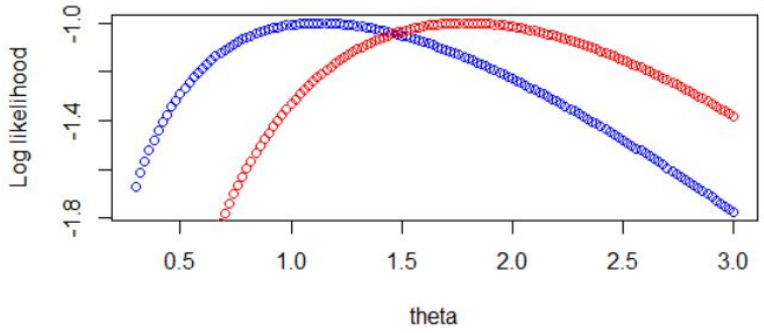
Assignment 2

1. The life length x is distributed exponentially.
2. Plot of the curve showing log likelihoods dependency on theta



The maximum value of theta looks to a bit above 1, using the which.max function i get a value for the optimal theta=1.12

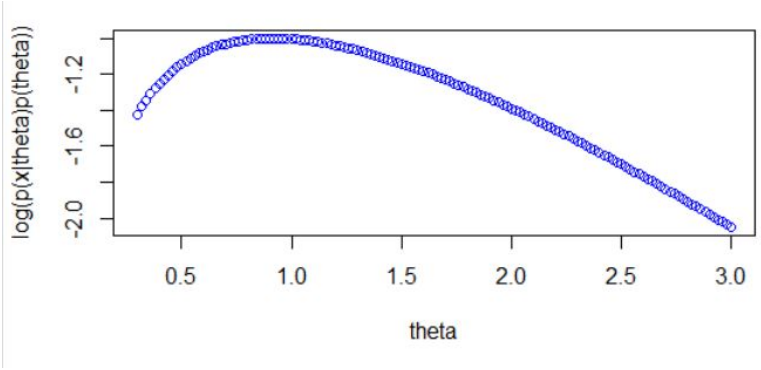
3. Now i look at the first 6 observations. I normalize the curves to see them on the same



plot.

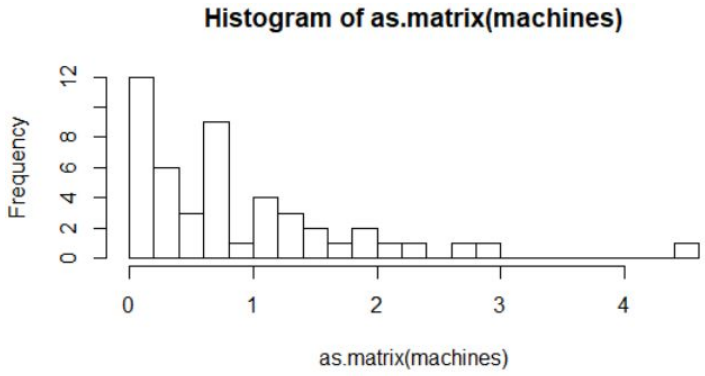
Blue curve is for the whole data set and red curve is for 6 observations. Optimal theta for these specific 6 observations is 1.78. When using fewer data points the maximum likelihood solution seems to be less reliable since the found optimal theta shifts so much.

4. Plot of the probability for the bayesian model.

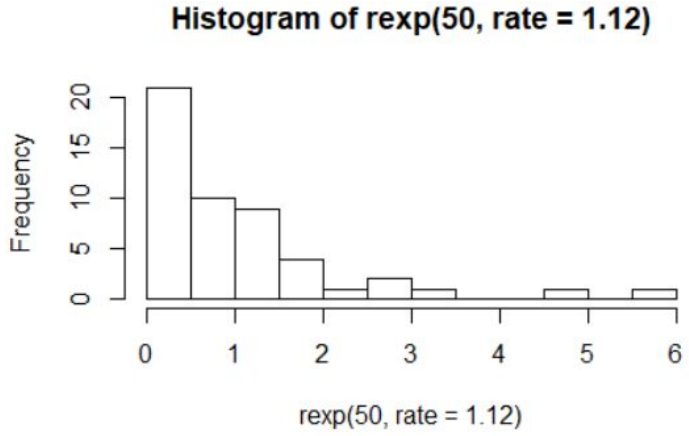


Optimal theta is 0.92. It is fairly close the the previously calculated optimal theta using the log likelihood measure. The measure that is calculated is bayes theorem excluding the marginal probability in denominator since it just scales the values and won't affect the optimal value of theta.

5. The histogram for the given data set looks like this:



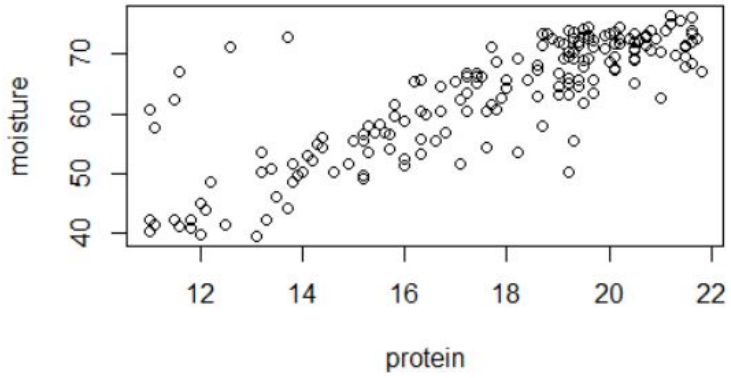
The histogram for a generated data set with theta=1.12 as in question 2 looks like this (using seed 12345):



Assignment 4

Changes

For assignment 4 there are two changes: The probabilistic model described in 4.2 was not correct and i forgot to consider case lambda = 0 in 4.7.

1. 

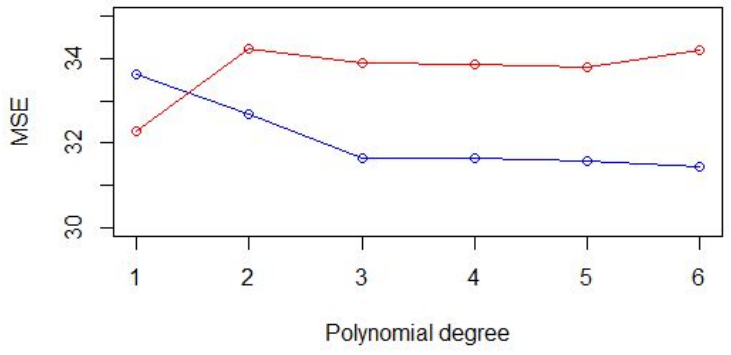
   I think that a linear model would describe the relationship fairly well but there is a lot of noise. Also the rate that the moisture increases as protein increases looks to slow down with protein>20 so maybe with more data i could more conclusively decide if a linear model is valid.

2. The model Mi can be described with a normal distribution where the parameters are a polynomial function and some standard deviation:

   $$Y \sim N(\beta_0 + \beta_1 x + ... + \beta_n x^n, \sigma^2)$$

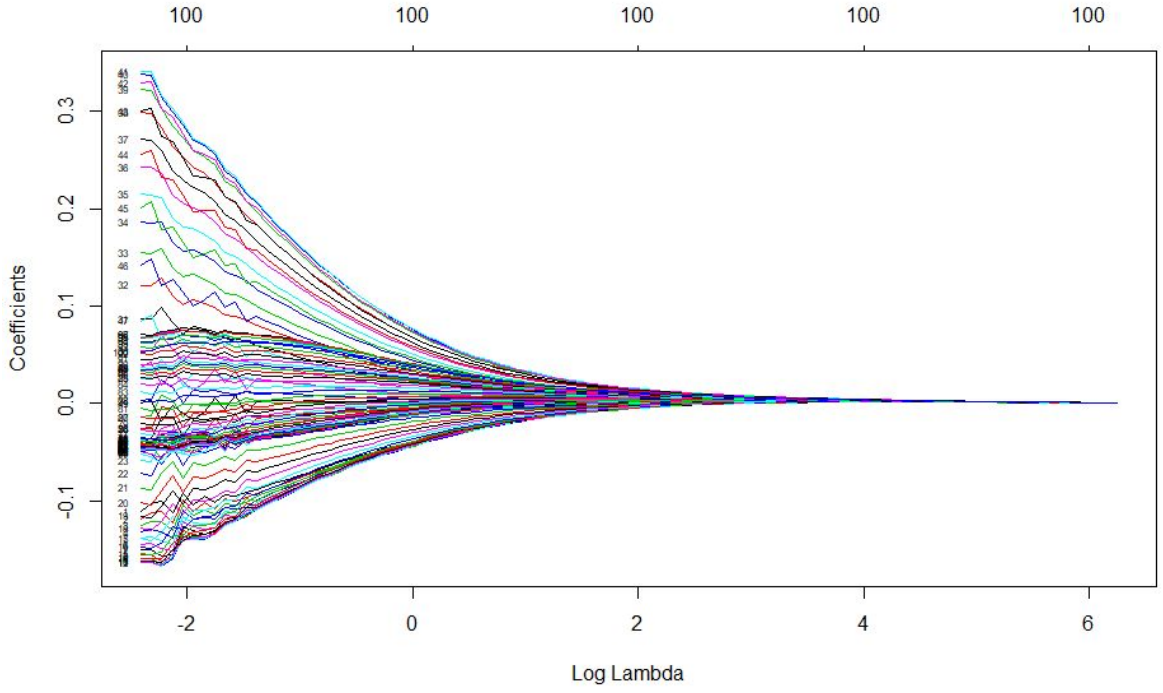
   Mean square error (MSE) will measure the mean error for the model. Using this i can compare different models and conclude which one is the best according to this measure. It is an intuitive way of comparing models based on distances.

3. Here i show a plot of how MSE changes depending on the degree of the polynomial for both train data (blue) and validation data (red).

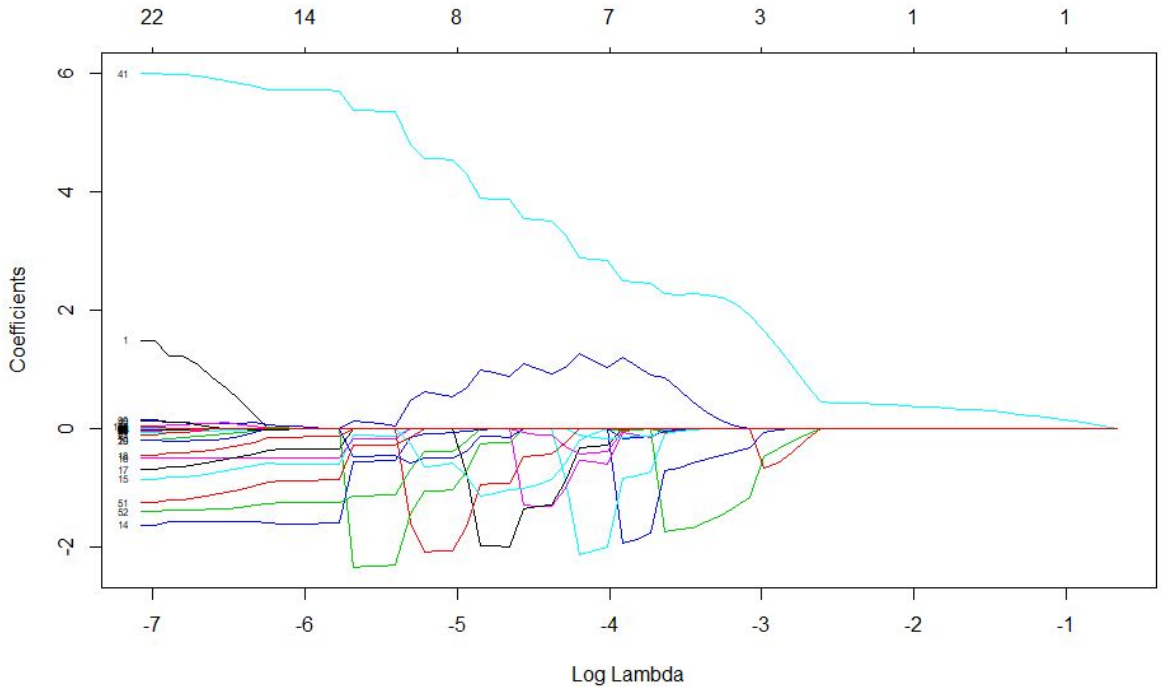


The blue line shows MSE for the train set and the red line shows MSE for the validation set. I see that the MSE for the train set decreases as the degree of the polynomial increases. This makes sense since the model will fit the trained set more the higher the degree is, eventually overfitting to the train set. The validation MSE shows that a polynomial of degree 1 (linear model) fits the data set the best. High degree polynomials will create high variance due to overfitting.

4. When performing variable selection where *Fat* is the response and channels 1-100 are predictors by using stepAIC we see that 63 variables are selected.
5. Fitting a ridge regression model with the same predictors and response gives me the following plot:

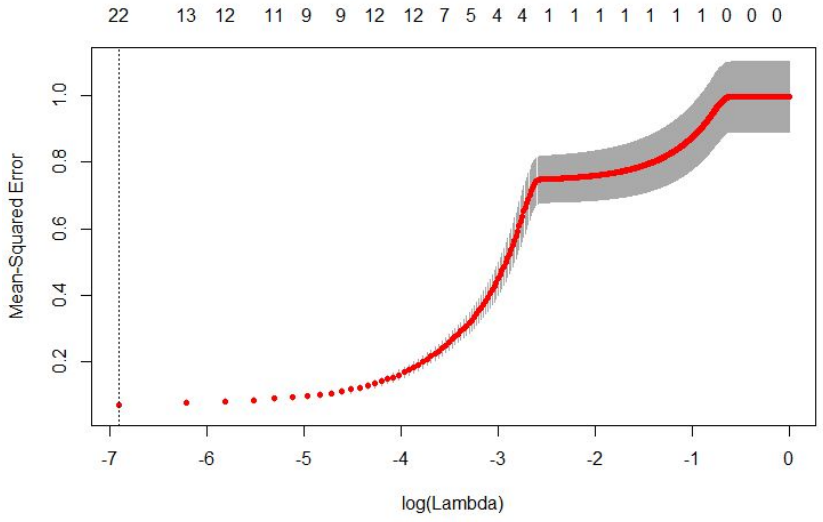


6. Similarly, i fit a LASSO model and get the following plot:



In the LASSO model i see that some coefficients go to 0 very quickly there are a lot of fluctuations where as in the ridge regression all coefficients slowly tend towards 0. This means that for the ridge regression many, if not all, of the coefficients will have a value until they all reach 0 while the LASSO regression will have a smaller of subset that is not equal to 0 at any given lambda.

7. When performing cross-validation to find the optimal LASSO model i find the optimal lambda to be 0 and since that means there is no penalty, every channel will be active.



The mean-squared error gets lower as lambda approaches 0.

8. The optimal LASSO model chooses all 100 variables while the AIC function chose 63 variables.

Code appendix

Assignment 1:

```
library("kknn", lib.loc="~/R/win-library/3.4")
library(readxl)
spambase = read_excel("spambase.xlsx")
n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))

#__use this for kknn_classifier
train=spambase[id,]
test=spambase[-id,]

#__use this for knearest()
test = as.matrix(test)
train = as.matrix(train)

#__Extracting true classes from training and test set
training_classes=train[,ncol(train)]
test_classes=test[,ncol(test)]

#_______Finding nearest neighbors using kknn()
kknn_classifier = function(k){
  Prob = numeric(nrow(test))
  kknn_package = kknn(Spam~.,train,test,k=k)
  fit = fitted(kknn_package)
  return (fit)
}

#______Finding nearest neighbors using cosine similarity
knearest=function(data,k,newdata) {
#________Calculating distance vector_________
  n1=dim(data)[1]
  n2=dim(newdata)[1]
  p=dim(data)[2]
  Prob=numeric(n2)
  X=as.matrix(data[,-p])
  Xn=as.matrix(newdata[,-p])
  X=X/matrix(sqrt(rowSums(X^2)), nrow=n1, ncol=p-1)
  Xn=Xn/matrix(sqrt(rowSums(Xn^2)), nrow=n2, ncol=p-1)

  c = X%*%t(Xn)
  d = 1-c

#________Finding nearest neighbors________
  for (i in 1:n2 ){
    #Extracts a column from the distance matrix
    col = d[,i]
    #Orders the distance column and saves the lowest k values's indexes
```

```
    nearest_indexes = order(col)[1:k]
    #Using indexes to find their classes
    #Using indexes to find their classes
    nearest_classes = training_classes[nearest_indexes]
    #Sets probability to the nearest neighbors class values divided by number of
neighbors examined
    Prob[i]=sum(nearest_classes)/k
  }
  return(Prob)
}
#Classifies cases given probabilities and a cutoff
#where values above cutoff is classified as spam
classify=function(p,cutoff){
  classification = vector(mode="numeric")
  for (i in 1:length(p)){
    if (p[i]>cutoff){
      classification[i]=1
    }
    else{
      classification[i]=0
    }
  }
  return (classification)
}
#Returns True positive rate and False positive rate by comparing two vectors of
classified observations
ROC=function(Y, Yfit, p){
  m=length(p)
  TPR=numeric(m)
  FPR=numeric(m)
  for(i in 1:m){
    t=table(Y,Yfit>p[i])
    TPR[i]=t[2,2]/sum(t[2,])
    FPR[i]=t[1,2]/sum(t[1,])
  }
  return (list(TPR=TPR,FPR=FPR))
}
#Code that is used to solve the questions
predict=knearest(train,5,test)
predict=kknn_classifier(5)
classification=classify(predict,0.5)
table(test_classes,classification)
roclist=ROC(test_classes,predict,seq(from=0.05,to=0.95,by=0.05))
tpr=roclist[["TPR"]]
fpr=roclist[["FPR"]]
plot(fpr,tpr,xlim = c(0,1),ylim = c(0,1))
abline(a=0,b=1)
```

Assignment 2:

```
library(readxl)
machines = read_excel("machines.xlsx")

#Calculates the log likelihood of dataset x and variable theta
#for a exponential distribution
log_likelihood = function(x,theta){
  p = prod(theta*exp(-1*theta*x))
  return (log(p))
}
#Calculates the likelihood for the given dataset
t=seq(from=0.3,to=3,by=0.02)
ll=numeric(0)
for(i in 1:length(t)){
  ll[i]=log_likelihood(machines,t[i])
}

#Same as above for first 6 observations
data_subset=machines[1:6,]
ll2=numeric(0)
for(i in 1:length(t)){
  ll2[i]=log_likelihood(data_subset,t[i])
}

#Calculating likelihood using the bayesian model
bayesian_model = function(x,theta){
  p = prod(theta*exp(-1*theta*x))
  prior=10*exp(-10*theta)
  l = log(p*prior)
  return (l)
}
bayes=numeric(0)
for(i in 1:length(t)){
  bayes[i]=bayesian_model(machines,t[i])
}

#Plotting log likelihood for the whole data set as a function of theta
norm_ll=ll/(abs(max(ll)))
plot(t,norm_ll,xlab = "theta",ylab = "Log likelihood",col="blue")
t[which.max(ll)]
#Plotting log likelihood for the six first observations as a function of theta
norm_ll2=ll2/abs(max(ll2))
points(t,norm_ll2,col="red")
t[which.max(ll2)]
#Plotting likelihood based on the bayesian model as a function of theta
norm_bayes=bayes/abs(max(bayes))
plot(t,norm_bayes,xlab = "theta",ylab = "log(p(x|theta)p(theta))")
t[which.max(bayes)]
```

```r
#Creates histogram for the given data set
hist(as.matrix(machines),15)
#Creates histogram for a generated data set
set.seed(12345)
hist(rexp(50,rate=1.12),15)
```

Assignment 4:

```
library(MASS)
library(readxl)
library(glmnet)
d = read_excel("tecator.xlsx")
data = as.matrix(d)
moisture=data[,ncol(data)]
protein=data[,ncol(data)-1]
#_______Protein vs Moisture plot
plot(protein,moisture)
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=d[id,]
validation=d[-id,]

##_______MSE CALCULATIONS_________________
#Train
mse_train = numeric(0)
for (i in 1:6){
  model = lm(Moisture~poly(Protein,i),train)
  Protein=train$Protein
  mse=mean((train$Moisture-predict(model,data.frame(Protein)))^2)
  mse_train[i]=mse
}
#Validation
mse_validation = numeric(0)
for (i in 1:6){
  model = lm(Moisture~poly(Protein,i),train)
  Protein=validation$Protein
  mse=mean((validation$Moisture-predict(model,data.frame(Protein)))^2)
  mse_validation[i]=mse
}
plot(mse_train,ylim = c(30,35),col="blue",ylab="MSE",xlab="Polynomial degree")
lines(mse_train,col="blue")
points(mse_validation,col="red")
lines(mse_validation,col="red")
##_______AIC_____________________________
fit=lm(Fat~., data.frame(data[,2:102]))
step=stepAIC(fit,direction = "both")
summary(step)
##_______Ridge regression________________
covariates=scale(data[,2:101])
response=scale(data[,102])

model0=glmnet(covariates,response,alpha=0,family="gaussian")
plot(model0,label=TRUE,xvar="lambda")
##_______LASSO___________________________
model1=glmnet(covariates,response,alpha=1,family="gaussian")
plot(model1,label=TRUE,xvar="lambda")
##_______CROSS VALIDATION________________
model2=cv.glmnet(covariates,response,alpha=1,family="gaussian",lambda =
```

```
seq(from=0,to=1,by=0.001))
model2$lambda.min
plot(model2)
coef(model2,s="lambda.min")
```