lab1block1P

Prudhvi Peddmallu

23 April 2019

Assignment 1. Spam classification with nearest neighbors

kknn(nearest neighbour),logisticregression,Feature selection by cross-validation in a linear model

The data file spambase.xlsx contains information about the frequency of various words, characters etc for a total of 2740 e-mails. Furthermore, these e-mails have been manually classified as spams (spam = 1) or regular e-mails (spam = 0). 1. Import the data into R and divide it into training and test sets (50%/50%) by using the following code: 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 (functions glm(), predict()) 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 (use table()) and the misclassification rates for training and test data. Analyse the obtained results. 3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1 if p(Y = 1/X) > 0.9$,otherwise$\hat{Y} = 0$ and report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have? 4. Use standard classifier kknn() with K=30 from package kknn, report the the misclassification rates for the training and test data and compare the results with step 2. 5. Repeat step 4 for K=1 and compare the results with step 4. What effect does the decrease of K lead to and why? ## libraries of kknn & glm

```
library(readxl)
library(kknn)
library(ggplot2)
library(glmnet)
library(MASS)
```

xlsx file

1.1-divideing the data

```
library(readxl)
spambase <-read_xlsx('spambase.xlsx')
### diveding the data
n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spambase[id,]
test=spambase[-id,]
```

1.2&1.3logistic regression

```
model <- glm(Spam~.,family=binomial(link='logit'),data=train)
predection1_test <- predict(model,newdata=test,type='response')
predection5_test <-ifelse(predection1_test > 0.5,1,0)

predection1_train <- predict(model,newdata=train,type='response')
predection5_train <-ifelse(predection1_train > 0.5,1,0)
confusionmatrix5_test<-data.frame("Spam" = test$Spam,
              "predection5_test"= as.numeric(predection5_test))
confusionmatrix5_test = table(confusionmatrix5_test)
confusionmatrix5_train<-data.frame("Spam" = train$Spam,"predection5_train"=
                                     as.numeric(predection5_train))
confusionmatrix5_train = table(confusionmatrix5_train)
diagonal_train<-sum(diag(confusionmatrix5_train))
misclassificationrate5_train <- 1-(sum(diagonal_train)/sum(confusionmatrix5_train))
diagonal_test<-diag(confusionmatrix5_test)
misclassificationrate5_test<-1-(sum(diagonal_test)/sum(confusionmatrix5_test))
```

```
cat("Using glm classifier dividing data at 0.5 ")

cat("Confusion matrix for train data")

confusionmatrix5_train

cat("Confusion matrix for test data")

confusionmatrix5_test

cat("misclassification rate for train data",misclassificationrate5_train)

cat("\nmisclassification rate for test data",misclassificationrate5_test)
```

1.2&1.3-2method

```
spam_data <- read.xlsx("spambase.xlsx", sheetName = "spambase_data")
spam_data$Spam <- as.factor(spam_data$Spam)
tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")
tecator_data <- tecator_data[,2:NCOL(tecator_data)] # removing sample column
set.seed(12345)
n = NROW(spam_data)
id = sample(1:n, floor(n*0.5))
train = spam_data[id,]
test = spam_data[-id,]
best_model <- glm(formula = Spam ~., family = binomial, data = train)
summary(best_model)
# prediction
train$prediction_prob <- predict(best_model, newdata = train, type = "response")
test$prediction_prob <- predict(best_model, newdata = test , type = "response")
train$prediction_class_50 <- ifelse(train$prediction_prob > 0.50, 1, 0)
test$prediction_class_50 <- ifelse(test$prediction_prob > 0.50, 1, 0)
train$prediction_class_90 <- ifelse(train$prediction_prob > 0.90, 1, 0)
test$prediction_class_90 <- ifelse(test$prediction_prob > 0.90, 1, 0)
```

```
# plots
ggplot(train, aes(prediction_prob, color = Spam)) +
geom_density(size = 1) + ggtitle("Training Set's Predicted Score for 50% cutoff") +
scale_color_economist(name = "data", labels = c("negative", "positive")) +
theme_economist()
ggplot(test, aes(prediction_prob, color = Spam)) +
geom_density(size = 1) + ggtitle("Test Set's Predicted Score for 50% cutoff") +
scale_color_economist(name = "data", labels = c("negative", "positive")) +
theme_economist()
#confusion table
conf_train <- table(train$Spam, train$prediction_class_50)
```

```
names(dimnames(conf_train)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train)
conf_test <- table(test$Spam, test$prediction_class_50)
names(dimnames(conf_test)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test)
#confusion table
conf_train1 <- table(train$Spam, train$prediction_class_90)
names(dimnames(conf_train1)) <- c("Actual Train", "Predicted Train")
conf_train1
conf_test1 <- table(test$Spam, test$prediction_class_90)
names(dimnames(conf_test1)) <- c("Actual Test", "Predicted Test")
conf_test1
#Choosing the best cutoff for test
cutoffs <- seq(from = 0.05, to = 0.95, by = 0.05)
accuracy <- NULL
for (i in seq_along(cutoffs)){
prediction <- ifelse(test$prediction_prob >= cutoffs[i], 1, 0) #Predicting for cut-off
accuracy <- c(accuracy,length(which(test$Spam == prediction))/length(prediction)*100)}
cutoff_data <- as.data.frame(cbind(cutoffs, accuracy))
ggplot(data = cutoff_data, aes(x = cutoffs, y = accuracy)) +
geom_line() +
ggtitle("Cutoff vs. Accuracy for Test Dataset")
```

Analysis:-When the algorithm was modeled using classification principle which divided at 0.5, the observed misclassification rate was 0.177 for test and 0.162 for train. As variance is not high, the issue of overfitting is not happening and the misclassification rate is comparitively smaller value which means the model is not underfitting as well. When the algorithm classified data at 0.9 , the misclassification rate increased to 0.312 for test data and 0.306 for training data. So the misclassification rate is increasing using this classification principle. On analysing the misclassification matrix we see an increase in observations which are actually spam but not considered as spam because of increment in the conditioning value. So the efficiency of the model in predicting the spams has decreased.

1.4-kknn-knearest neighbour

```
#train data
knearest_30_1 <- kknn(formula = factor(Spam) ~., train, train,k = 30, kernel = "optimal")
fit_kknn30_1 <- fitted(knearest_30_1)
tablekknn_30_1 <- table(train$Spam,fit_kknn30_1)
misclassificationkknn_30_1 <- 1- (sum(diag(tablekknn_30_1))/sum(tablekknn_30_1))
```

```
#test data
knearest_30_2 <- kknn(formula = factor(Spam) ~., train, test,k = 30, kernel = "optimal")
fit_kknn30_2 <- fitted(knearest_30_2)
tablekknn_30_2 <- table(test$Spam,fit_kknn30_2)
misclassificationkknn_30_2 <- 1- (sum(diag(tablekknn_30_2))/sum(tablekknn_30_2))
```

```
cat("kknn algorithm with k = 30")

cat("Confusion matrix for train data")

tablekknn_30_1

cat("Confusion matrix for test data")

tablekknn_30_2

cat("misclassification rate for train data = ", misclassificationkknn_30_1)

cat("misclassification rate for test data = ", misclassificationkknn_30_2)
```

1.4 kknn-2method

```
#k=30
knn_model30 <- train.kknn(Spam ~., data = train, kmax = 30)
train$knn_prediction_class <- predict(knn_model30, train)
test$knn_prediction_class <- predict(knn_model30, test)
conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)
conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
#k=1
knn_model1 <- train.kknn(Spam ~., data = train, kmax = 1)
train$knn_prediction_class <- predict(knn_model1, train)
test$knn_prediction_class <- predict(knn_model1, test)
conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)
conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
```

The classifier kknn() was used to do the classification model. This function is used to do classification using K nearest neighbour algorithm. Here we take K=30 (considering 30 nearest neighbours). A misclassification rate of 0.329 is obtained for test data and 0.172 is obtained for train data.

On comparing this results with step 2, there is no much change in train error but the test error shows a significant increase where here it is 0.329 whereas in step 2 it was 0.17.

So here we can say that the model in step 2 is better compared to the model using k nearest neighbout algorithm with number of neighbours is 30.

k=1

Misclassification for test data increases from 0.32 to 0.34 whereas training error decreased to zero on selecting k as 1. Here none of the training data is misclassified. So here we can say that when we take k as 1, the model is highly biased to train data and is not a generalised model

So as value of K increases, the model is getting generalised which inturn decreases the test error.

The accuracy of k nearest neighbour algorithm depends on selecting the appropriate features. If not the model will be affected by the noisy data.

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

1. Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions). Your function should depend on: . X: matrix containing X measurements. . Y: vector containing Y measurements . Nfolds: number of folds in the cross-validation. You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted, and the seed 12345 should be used for that purpose.
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

3.163.2-k-fold cross-validation

```
feature_selection <-  function(X,Y,Nfolds){


  Intercept <- rep(1,length(Y))

  Number_Of_Features <- c()
  cv_rate <- c()
  best_model <- 0
  cv_rate_bestmodel <- 0

  total_features <- ncol(X)

  for(k in 1: total_features){
    M = combn(total_features , k)

    for(j in 1:ncol(M)){
      Z = X[ ,M[ , j]]

      #Adding intercept
      Z = as.matrix(cbind(Intercept , Z))

      data <- cbind(Y ,  Z)
```

```
      #Randomly shuffle the data
      set.seed(12345)

      data_s<-data[sample(nrow(data)), ]

      #Create N equally size folds
      folds <- cut(seq(1,nrow(data_s)),breaks=Nfolds,labels=FALSE)
      #folds <- rep_len(1:Nfolds,nrow(data_s))

      cv = 0
      #Nfolds cross validation to find the best model

      for(i in 1:Nfolds){

        testIndexes <- which(folds==i,arr.ind=TRUE)

        test_fold <- data_s[testIndexes, ]
        training_fold <- data_s[-testIndexes, ]

        X_train = as.matrix(training_fold[,-1])
        Y_train = as.matrix(training_fold[,1])

        beta_hat <- solve(t(X_train)%*%X_train)%*%t(X_train)%*% Y_train
        y_hat <- as.vector(t(beta_hat) %*% t(as.matrix(test_fold[,-1])))

        residual_vector = as.vector(test_fold[,1]) - y_hat
        rss <- sum(residual_vector^2)/(length(y_hat))

        cv = cv+rss

      }

      cv = cv/Nfolds
      cv_rate <- c(cv, cv_rate)

      if(min(cv_rate)==cv){
        best_model = colnames(Z[,-1])
        cv_rate_bestmodel = cv
      }

      Number_Of_Features <- c((ncol(Z)-1), Number_Of_Features)
    }
  }

  output <- list("Minimum CV rate" = cv_rate_bestmodel,
                 "Features selected" = as.vector(best_model))
  plot(x=Number_Of_Features,cv_rate,
       xlab="Number of features",
          ylab="Cross validation rate",
                main = "MSE vs Cross validation rate")
  return(output)
}
data(swiss)
```

```
Y = swiss$Fertility
X = swiss[,-c(1)]
Nfolds = 5

feature_selection(X,Y,Nfolds)
```

By using package k fold of cv

```
#By using package k fold of cv
#this is to perform one model
model<-glm(swiss~Fertility,data=swiss)
mse_cv<-cv.glm(data=swiss,model)#this gives the crossvalidation values
mse_cv<-cv.glm(data = swiss,model)$delta[1]#gives MSE value for the model
#k folds like 10folds and 5 folds we need to use K = 5 or 10
library(boot)
Msevalue= Null()
for (i in 1:10) {
  model<-glm(swiss~ploy(Fertility),i)#ploy gives fertility power,i will take power values
  Msevalue[i]=cv.glm(data = swiss,model,k= 10)$delta[1]

}
Msevalue
```

3.1&3.2-2method-k-fold cross validation

```
subset_function <- function(X,Y,N){
# X = swiss[,2:6]
# Y = swiss[,1:1]
# N = 5
df <- cbind(X,Y)
temp <- NULL
final <- NULL
for(i in 1:NCOL(X)){
combs <- as.data.frame(gtools::combinations(NCOL(X), r=i, v=colnames(X), repeats.allowed=FALSE))
combs <- tidyr::unite(combs, "formula", sep = ",")
temp <- rbind(combs, temp)
}
set.seed(12345)
df2 <- df[sample(nrow(df)),]
df2$k_fold <- sample(N, size = nrow(df), replace = TRUE)
result <- NULL
for (j in 1:NROW(temp))
{
for(i in 1:N){
train = df2[df2$k_fold != i,]
test = df2[df2$k_fold == i,]
vec <- temp[j,]
train_trimmed = lapply(strsplit(as.character(vec), ","), function(x) train[x])[[1]]
test_trimmed = lapply(strsplit(as.character(vec), ","), function(x) test[x])[[1]]
y_train = train[,c("Y"), drop = FALSE]
```

```
y_test = test[,c("Y"), drop = FALSE]
train_trimmed = as.matrix(train_trimmed)
test_trimmed = as.matrix(test_trimmed)
y_test = as.matrix(y_test)
y_train = as.matrix(y_train)
t_train = as.matrix(t(train_trimmed))
t_test = as.matrix(t(test_trimmed))
betas = solve(t_train %*% train_trimmed) %*% (t_train %*% y_train)
train_trimmed = as.data.frame(train_trimmed)
test_trimmed = as.data.frame(test_trimmed)
train_trimmed$type = "train"
test_trimmed$type = "test"
final <- rbind(train_trimmed, test_trimmed)
y_hat_val = as.matrix(final[,1:(ncol(final)-1)]) %*% betas
mse = (Y - y_hat_val)^ 2
data <- cbind(i, vec, mse, type = final$type)
result <- rbind(data, result)
}}
result <- as.data.frame(result)
colnames(result) <- c("kfold", "variables", "mse", "type")
result$mse <- as.numeric(result$mse)
result$no_variables <- nchar(as.character(result$variables)) - nchar(gsub('\\,', "", result$variables))
variable_performance <- result_test %>% group_by(kfold, no_variables) %>%
summarise(MSE = mean(mse, na.rm = TRUE))
myplot <- ggplot(data = variable_performance, aes(x = no_variables, y = MSE, color=kfold)) +
geom_line() + ggtitle("Plot of MSE(test) vs. Number of variables")
myplot2 <- ggplot(data = result_test, aes(x = variables, y = mse, color=kfold)) +
geom_bar(stat="identity") + ggtitle("Plot of MSE(test) vs. Features by folds") + coord_flip()
best_variables <- result_test %>% group_by(variables) %>%
summarise(MSE = mean(mse, na.rm = TRUE)) %>% arrange(MSE) %>%
select(variables) %>% slice(1) %>% as.vector()
return(list(myplot, myplot2, best_variables))
}
subset_function(X = swiss[,2:6], Y = swiss[,1:1], N = 5)
```

From the plot, it is clear that as number of features increases the variance in cross validation rate is decreasing and also the cross validation rate reduces to smaller values which means the error rates are decreasing and model efficiency is increasing.

We used the swiss data which gives information on fertility rates in switzerland.

The features selected for the best model are :

Agriculture, Education, Catholic and Infant mortality along with the intercept. This feature selection can be justified as

1. Agriculture: The income from agriculre can affect fertility rate and higer the income, the socio economic conditions becomes better which in turn supports higher fertility and vice versa
2. Education: As more people are educated, the knowledge on factors affecting fertility impact fertility
3. Catholic: Catholic religion also affects fertility positively due to the preaching given by the religion
4. Infant mortality: Infant mortality rate directly affects fertility. As mortality rate increases fertility decreases

Also from feature selection, the feature examination is not selected and it can be justified as a person writing or clearing an examination does not affect fertility.

Assignment 4. Linear regression and regularization

The Excel file tecator.xlsx contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry

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?
2. Consider model ???????????????? in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power ???????? (i.e M1 is a linear model, M2 is a quadratic model and so on). Report a probabilistic model that describes ????????????????. Why is it appropriate to use MSE criterion when fitting this model to a training data?
3. Divide the data into training and validation sets(50%/50%) and fit models M????????????????,????????=1.6. For each model, record the training and the validation MSE and present a plot showing how training and validation MSE depend on i (write some R code to make this plot). Which model is best according to the plot? How do the MSE values change and why? Interpret this picture in terms of bias-variance tradeoff. Use the entire data set in the following computations:
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.
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 ???????? and report how the coefficients change with ????????.
6. Repeat step 5 but fit LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?
7. Use cross-validation to find the optimal LASSO model (make sure that case ????????=0 is also considered by the procedure) , report the optimal ???????? 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 with ????????.
8. Compare the results from steps 4 and 7.

4.1-linear model

```
#Read data
tecator = read_xlsx("tecator.xlsx")
tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")
tecator_data <- tecator_data[,2:NCOL(tecator_data)] # removing sample column
```

```
#linearmodel
p<-ggplot(tecator, aes(x= Protein, y=Moisture))+
                      geom_point()+
                  xlab("Protein")+ylab("Moisture")+
                  ggtitle("Moisture vs Protein")
p
```

The plot looks linear by seeing the distribution of scatter points. A linear relationship is observed where as the protein increases, moisture is also increasing.

4.2-proof-The probabilistic model

The data can be fit using a polynomial function below:

$y(x, w) = \sum_{j=0}^{6} w_j x^j$ where x is the protein variable and $w = (w_0.....w_6)$ are the coefficients

Here, it is appropriate to use MSE criterion while fitting the model to training data because, the value of the coefficients in the above equation can be determined by minimising the error function and mean squared error function is one of the most simple and easily calculated error function.

Assuming the observed output values as $t = (t_1.....t_6)^T$

We can minimize the error function and select values for coefficients that have the minimum value using :

$E(w) = \frac{1}{2} \sum_{n=1}^{6} y((x_n, w) - t_n)^2$

The probabilistic model

The goal here is to make predictions for the Moisture values based on a set of protein values as inputs. The probability distribution of the target variables can be expressed as below

$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$

The above distribution explains that the output values has a gaussian distribution with mean $y(x, w)$ and variance β^{-1}

To find the value of the coefficients can be derived using maximum likelihood. The likelihood function can be defined as below

$p(t|x, w, \beta) = \prod_{n=1}^{6} \mathcal{N}(t_n|y(x_n, w), \beta^{-1}$

maximizing the logarithm of above likelihood function gives us the coefficients

$lnp(t|x, w, \beta) = -\beta/2 \sum_{j=1}^{6} (y(x_n, w) - t_n)^2$

Maximising the above function gives the polynomial coeffieients to be used with various polynomial degrees of protein to obtain the result.

4.3-Each model validation and MSE-mycode &groupcode

```
library(gally)
#mycode
protein<-tecator_data %>%#Dividingthe data50/50
select(Moisture,Protein) %>%
mutate(
Protein2 = Protein^2,
Protein3 = Protein^3,
Protein4 = Protein^4,
Protein5 = Protein^5,
Protein6 = Protein^6
)
#divide the data
n=dim(protein)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
trainp=protein[id,]
testp=protein[-id,]
#Train & Test-MSE for
#protein model 0
```

```r
protein_model_0<-lm(Moisture~ Protein , data = trainp)
proteintrain_0MSE <- mean(protein_model_0$residuals^2)
protein_model0predict <- predict(protein_model_0,testp)
protein_test_0MSE<-sum(((protein_model0predict - testp$Moisture)^2))/nrow(testp)

#protein model 1
protein_model_1<-lm(Moisture~ Protein + Protein2, data = trainp)
proteintrain_1MSE <- mean(protein_model_1$residuals^2)
protein_model1predict <- predict(protein_model_1,testp)
protein_test_1MSE<- sum(((protein_model1predict - testp$Moisture)^2))/nrow(testp)

#protein model 2
protein_model_2<-lm(Moisture~ Protein + Protein2 + Protein3, data = trainp)
proteintrain_2MSE <- mean(protein_model_2$residuals^2)
protein_model2predict <- predict(protein_model_2,testp)
protein_test_2MSE<- sum(((protein_model1predict - testp$Moisture)^2))/nrow(testp)

#protein model 3
protein_model_3<-lm(Moisture~ Protein + Protein2 + Protein3 + Protein4, data = trainp)
proteintrain_3MSE <- mean(protein_model_3$residuals^2)
protein_model3predict <- predict(protein_model_3,testp)
protein_test_3MSE<- sum(((protein_model3predict - testp$Moisture)^2))/nrow(testp)

#protein model 4
protein_model_4<-lm(Moisture~ Protein + Protein2 + Protein3 + Protein4 + Protein5,data = trainp)
proteintrain_4MSE <- mean(protein_model_4$residuals^2)
protein_model4predict <- predict(protein_model_4,testp)
protein_test_4MSE<- sum(((protein_model4predict - testp$Moisture)^2))/nrow(testp)

#protein model 5
protein_model_5<-lm(Moisture~ Protein+Protein2 +Protein3+Protein4 +Protein5+Protein6,data = trainp)
proteintrain_5MSE <- mean(protein_model_5$residuals^2)
protein_model5predict <- predict(protein_model_5,testp)
protein_test_5MSE<- sum(((protein_model5predict - testp$Moisture)^2))/nrow(testp)
#plotMSE & model complexity
a <- c(protein_test_0MSE,protein_test_1MSE,protein_test_2MSE,protein_test_3MSE,
       protein_test_4MSE,protein_test_5MSE)
b <- c(proteintrain_0MSE,proteintrain_1MSE,proteintrain_2MSE,proteintrain_3MSE,
       proteintrain_4MSE,proteintrain_5MSE)
protein_plot<-data.frame(a,b)

ggplot(protein_plot) + geom_line(aes(y=a,x=1:6),colour='red') + geom_line(aes(y=b,x=1:6),colour='blue')

#groupcode
tecator$Protein2 <- tecator$Protein^2
tecator$Protein3 <- tecator$Protein^3
tecator$Protein4 <- tecator$Protein^4
tecator$Protein5 <- tecator$Protein^5
tecator$Protein6 <- tecator$Protein^6

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

```
tecator_train=tecator[id,]
tecator_test=tecator[-id,]


formula_list <- c(Moisture~Protein,Moisture~Protein+Protein2,
                  Moisture~Protein+Protein2+Protein3,
                  Moisture~Protein+Protein2+Protein3+Protein4,
                  Moisture~Protein+Protein2+Protein3+Protein4+Protein5,
                  Moisture~Protein+Protein2+Protein3+Protein4+Protein5+Protein6)



fitlist <- lapply(formula_list, function(x){

lm(x, data = tecator_train)
})


MSE_train <- sapply(fitlist , function(x){

  sum((x$fitted.values - tecator_train$Moisture)^2)/nrow(tecator_train)
})


MSE_test <- sapply(fitlist , function(x){

  y_hat_test = predict(x, tecator_test)
  sum((y_hat_test - tecator_test$Moisture)^2)/nrow(tecator_test)
})

MSE_data <- data.frame("order"= 1:length(MSE_train),"train" = MSE_train,
                       "test" = MSE_test)
ggplot(data = MSE_data)+geom_line(aes(x = order, y = MSE_train), colour = "Blue")+
  geom_line(aes(x = order, y=MSE_test), color = "Red")+
  xlab("Order of Polynomial") + ylab("Mear squared error")
```

4.3-2method

```
final_data <- tecator_data
magic_function <- function(df, N)
{
df2 <- df
for(i in 2:N)
{
df2[paste("Protein_",i,"_power", sep="")] <- (df2$Protein)^i
}
df2 <- df2[c("Protein_2_power", "Protein_3_power",
"Protein_4_power", "Protein_5_power",
"Protein_6_power")]
df <- cbind(df,df2)
return(df)
}
```

```
final_data <- magic_function(final_data, 6)
set.seed(12345)
n = NROW(final_data)
id = sample(1:n, floor(n*0.5))
train = final_data[id,]
test = final_data[-id,]
# model building
M_1 <- lm(data = train, Moisture~Protein)
M_2 <- lm(data = train, Moisture~Protein+Protein_2_power)
M_3 <- lm(data = train, Moisture~Protein+Protein_2_power+Protein_3_power)
M_4 <- lm(data = train, Moisture~Protein+Protein_2_power+Protein_3_power+
Protein_4_power)
M_5 <- lm(data = train, Moisture~Protein+Protein_2_power+Protein_3_power+
Protein_4_power+Protein_5_power)
M_6 <- lm(data = train, Moisture~Protein+Protein_2_power+Protein_3_power+
Protein_4_power+Protein_5_power+Protein_6_power)
train$type <- "train"
test$type <- "test"
final_data <- rbind(test, train)
# predicting new values
M_1_predicted <- predict(M_1, newdata = final_data)
M_2_predicted <- predict(M_2, newdata = final_data)
M_3_predicted <- predict(M_3, newdata = final_data)
M_4_predicted <- predict(M_4, newdata = final_data)
M_5_predicted <- predict(M_5, newdata = final_data)
M_6_predicted <- predict(M_6, newdata = final_data)
# calculating the MSE
final_data$M_1_error <- (final_data$Moisture - M_1_predicted)^2
final_data$M_2_error <- (final_data$Moisture - M_2_predicted)^2
final_data$M_3_error <- (final_data$Moisture - M_3_predicted)^2
final_data$M_4_error <- (final_data$Moisture - M_4_predicted)^2
final_data$M_5_error <- (final_data$Moisture - M_5_predicted)^2
final_data$M_6_error <- (final_data$Moisture - M_6_predicted)^2
# Chainning like Chainsaw
final_error_data <- final_data %>% select(type, M_1_error, M_2_error, M_3_error,
M_4_error, M_5_error, M_6_error) %>%
gather(variable, value, -type) %>%
separate(variable, c("model", "power", "error"), "_") %>%
group_by(type, power) %>%
summarise(MSE = mean(value, na.rm=TRUE))
ggplot(final_error_data, aes(x = power, y = MSE, color=type)) + geom_point() +
ggtitle("Mean squared error vs. model complexitiy by dataset type")
```

According to the plot, when the polynomial order is 1 and 2, the bias is high but the variance is less. So from this we can say that the model here is simple and will not be able to capture underlying patterns as described by data. When polynomial order increses further beyond 2, the variance is high but is stabilised at a particular value until the order of polynomial increases upto 5. Beyond 5, both bias and varience increases. So from this we can conclude that, when the order of the polynomial is beyond 5 overfitting happens.

When we take bias - variance tradeoff into consideration , model with order of polynomial from 3 to 5 gives the better model

4.4step AIC-linera model by using AIc

```
library(MASS)
formula_t = as.formula(paste("Fat~", paste(names(tecator[2:101]),
collapse = "+")))
fit <- lm(formula = formula_t,data=tecator)
step <- stepAIC(fit, direction="backward")
selcted_variables = step$anova
summary(step$anova)
mycode by using this i got 63 features
library(MASS)
library(dplyr)
#Filtering  the data
tecator_data1<-tecator_data[,2:102]
#perform variable selection of a linear model
Fat_lm <- lm(Fat~.,data = tecator_data1)
#predictors by using stepAIC
AIC_fat <-stepAIC(Fat_lm,direction = "both" ,trace = FALSE)
# anova will display final variables
AIC_fat$anova
```

```
#2method
min.model1 = lm(Fat ~ 1, data=tecator_data[,-1])
biggest1 <- formula(lm(Fat ~., data=tecator_data[,-1]))
step.model1 <- stepAIC(min.model1, direction ='forward', scope=biggest1, trace = FALSE)
summ(step.model1)
```

38 variables are selected. So if we select the 38 features which are obtained using the above algorithm and model a regression algorithm, the model will be as good in predicting the output as the model which uses all 100 variables. Here on trade of between goodness of fit and simplicity of model, a model with 38 features was giving the best result

4.5-Ridge-regression-alpha=0

```
#mycode
#libraries
library(glmnet)
#Ridge-regression plot
Channel_ridge=scale(tecator_data[,2:101])
Fat_ridge=scale(tecator_data[,102])
Ridge_model=glmnet(as.matrix(Channel_ridge),
Fat_ridge, alpha=0,family="gaussian")
plot(Ridge_model, xvar="lambda", label=TRUE)
#groupcode
reg_data = tecator[,2:102]#tectator is the data
reg_data=scale(reg_data)
covariates=reg_data[,1:100]
response=reg_data[, 101]
model_ridge= glmnet(as.matrix(covariates), response,
                    alpha=0,family="gaussian")
plot(model_ridge, xvar="lambda", label=TRUE)
#2-method
y <- tecator_data %>% select(Fat) %>% data.matrix()
```

```
x <- tecator_data %>% select(-c(Fat)) %>% data.matrix()
lambda <- 10^seq(10, -2, length = 100)
ridge_fit <- glmnet(x, y, alpha = 0, family = "gaussian", lambda = lambda)
plot(ridge_fit, xvar = "lambda", label = TRUE,
main = "Plot showing shrinkage of coefficents with rise in log of lambda")
## Change of coefficent with respect to lambda
result <- NULL
for(i in lambda){
temp <- t(coef(ridge_fit, i)) %>% as.matrix()
temp <- cbind(temp, lambda = i)
result <- rbind(temp, result)
}
result <- result %>% as.data.frame() %>% arrange(lambda)
table_cofe <- head(result, 10) %>% select(Channel1, Channel2, Channel84, Channel62,
Channel53, Channel75, Channel57,Protein,
lambda)
knitr::kable(table_cofe, caption = "Coefficent of Ridge Regression vs. Lambda")
```

Analysis: The idea of lasso and ridge regression is to introduce bias to variables in order to reduce/account for multicollinearity. Introducing bias (lambda) to covarience matrix is done by multiplying the diagonal elements by lambda(often 1+lambda), this inflates the covarience of predictors compared to correleations of predictors. The idea is to test the stability of betas that is how likely are the betas/coefficents of regressions to be stable if we keep introducing bias. We can clearly see that 'Channel1' and 'Channel2' betas go from positive to negative with very little bias introduced while terms like 'Channel75' dont change the beta signs. Thus the practice is exclude the terms whose beta/coefficent dont change drastically much within say first 10 introduction of lambda. We see that many of the terms/coefficent tend to zero at around log(lambda) that is ~5. In the above plot, as the lambda value increases, the coefficients are heavily penalized and the effect of all the coeeficients other than intercept reduces to zero (not absolute sero) when Log Lambda reaches 4. So when lamda reached 4, all the coefficients shrink to abosilute zero.

4.6-Lasso regression model alpha=1

```
#groupcode
model_lasso= glmnet(as.matrix(covariates), response,
alpha=1,family="gaussian")
plot(model_lasso, xvar="lambda", label=TRUE)

#mycode
#Lasso-regression plot
Channel_lasso=scale(tecator_data[,2:101])
Fat_lasso=scale(tecator_data[,102])
Lasso_model=glmnet(as.matrix(Channel_lasso),Fat_lasso, alpha=1,family="gaussian")
plot(Lasso_model, xvar="lambda", label=TRUE)
#2method
lambda <- 10^seq(10, -2, length = 100)
lasso_fit <- glmnet(x, y, alpha = 1, family = "gaussian", lambda = lambda)
plot(lasso_fit, xvar = "lambda", label = TRUE,
main = "Plot showing shrinkage of coefficents with rise in log of lambda")
```

When we compare the plot of LASSO with the plot of Ridge regression, as labda increases the coefficients of less relevent features are set to 0 and are not considered and select the other features. So this method of shrinkage can be used for feature selection which is absent in ridge regression.

Analysis: We quickly see that very little introduction of penalisation/bias is all it takes to make many terms/coefficent to zero. This implies for the full dataset lasso is much better suited for regularisation compared to ridge. At lambda around 1 (log lambda is 0) we get only two or three non zero terms.

4.7-Crossvalidation to find optimal LASSO

```
#group code
model_lasso1=cv.glmnet(as.matrix(covariates), response, alpha=1,
                       family="gaussian",lambda=seq(0,1,0.001))

lambda_min = model_lasso1$lambda.min
plot(model_lasso1)
#my code
#Useing-cross-validation to find the optimal LASSO model
Channel_lasso_cross=scale(tecator_data[,2:101])
Fat_lasso_cross=scale(tecator_data[,102])
Lasso_crossmodel=cv.glmnet(as.matrix(Channel_lasso_cross),
Fat_lasso_cross, alpha=1,family="gaussian")
plot(Lasso_crossmodel)
coef(Lasso_crossmodel, s="lambda.min")
#2method
#find the best lambda from our list via cross-validation
lambda_lasso <- 10^seq(10, -2, length = 100)
lambda_lasso[101] <- 0
lasso_cv <- cv.glmnet(x,y, alpha=1, lambda = lambda_lasso, type.measure="mse")
coef(lasso_cv, lambda = lasso_cv$lambda.min)
lasso_cv$lambda.min
result_lasso <- NULL
for(i in 1:length(lambda_lasso)){
temp <- lasso_cv$cvm[i] %>% as.matrix()
temp <- cbind(CVM_error = temp, lambda = lasso_cv$lambda[i])
result_lasso <- rbind(temp, result_lasso)
}
result_lasso <- result_lasso %>% as.data.frame() %>% arrange(lambda)
colnames(result_lasso) <- c("Cross_Mean_Error", "Lambda")
ggplot(result_lasso, aes(x = log(Lambda), y = Cross_Mean_Error)) + geom_point() +
ggtitle("Cross Validation Error vs. Lambda")
```

Analysis: The minimum value of lambda was 0, implies zero penalisation. The variables selected are: Channel98, Channel99, Channel100, Protein, Moisture, Channel37, Channel38, Channel39, Channel40 along with intercept. We see that Cross validation error is lowest at lambda= 0 and remains low till lambda~1 (log lambda 0) after which the error drastically increases at log(lambda) ~ 2.5, the error maxes out and remains about the same for higher values of lambda. This implies that more bias introduction will lead to worse performance.

4.7comparing step AIC and lasso

In step 4 we are using the function stepAIC which selectes the best model using Akaike information criterion. Here first a model with lowest AIC value is selected and the features or predictors are removed one by one and reached a final model with a set of features which has impact on the model.

In step 7, we use cross validation in LASSO regression. Here we consider a tradeoff between goodness of fit and the error rate and we choose a lambda value which gives us the best model taking the error rate as well

as complexity of model into consideration. So here we can see from the plot that the maximum number of features that the cross validation on LASSO gives us 22 which is lesser than 38 obtained in step 7(Using AIC). In AIC all the variables impacting the model are taken where as in LASSO a trade-off is made where we compromise on few features to minimise the error.

loglikelihood

Assignment 2. Inference about lifetime of machines

The data file machines.xlsx contains information about the lifetime of certain machines, and the company is interested to know more about the underlying process in order to determine the warranty time. The variable is following: . Length: shows lifetime of a machine 1. Import the data to R.

2. Assume the probability model $p(x/\theta) = \theta e^{-\theta x}$ for x=Length in which observations are independent and identically distributed. What is the distribution type of x? Write a function that computes the log-likelihood log $p(x/\theta)$ for a given ???????? and a given data vector x. Plot the curve showing the dependence of log-likelihood on θ where the entire data is used for fitting. What is the maximum likelihood value of θ according to the plot?

3. Repeat step 2 but use only 6 first observations from the data, and put the two log-likelihood curves (from step 2 and 3) in the same plot. What can you say about reliability of the maximum likelihood solution in each case?

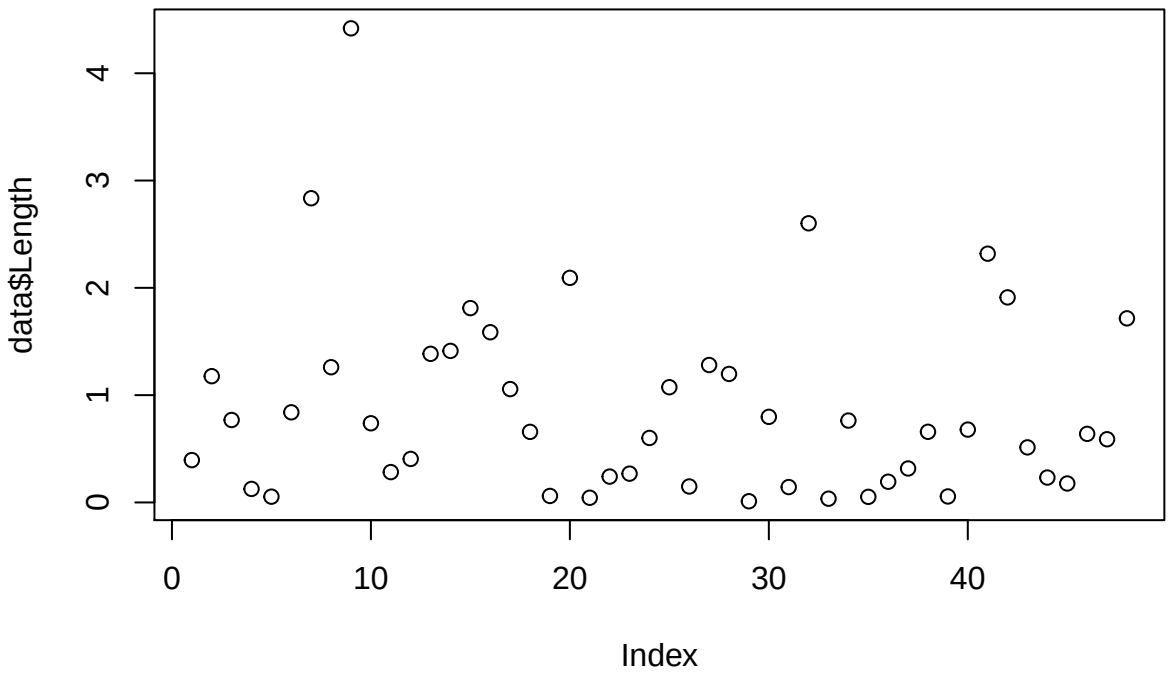
4. Assume now a Bayesian model with $p(x/\theta) = \theta e^{-\theta x}$ and prior = $(p(\theta) = \lambda e^{-\lambda\theta}, \lambda = 10$ Write a function computing $I(\theta) = log(p(x/\theta).p(\theta))$ What kind of measure is actually computed by this function? Plot the curve showing the dependence of $I(\theta)$ on θ???????? computed using the entire data and overlay it with a plot from step 2. Find an optimal θ and compare your result with the previous findings.

5. Use θ ???????? value found in step 2 and generate 50 new observations from $p(x/\theta) = \theta e^{-\theta x}$ (use standard random number generators). Create the histograms of the original and the new data and make conclusions.

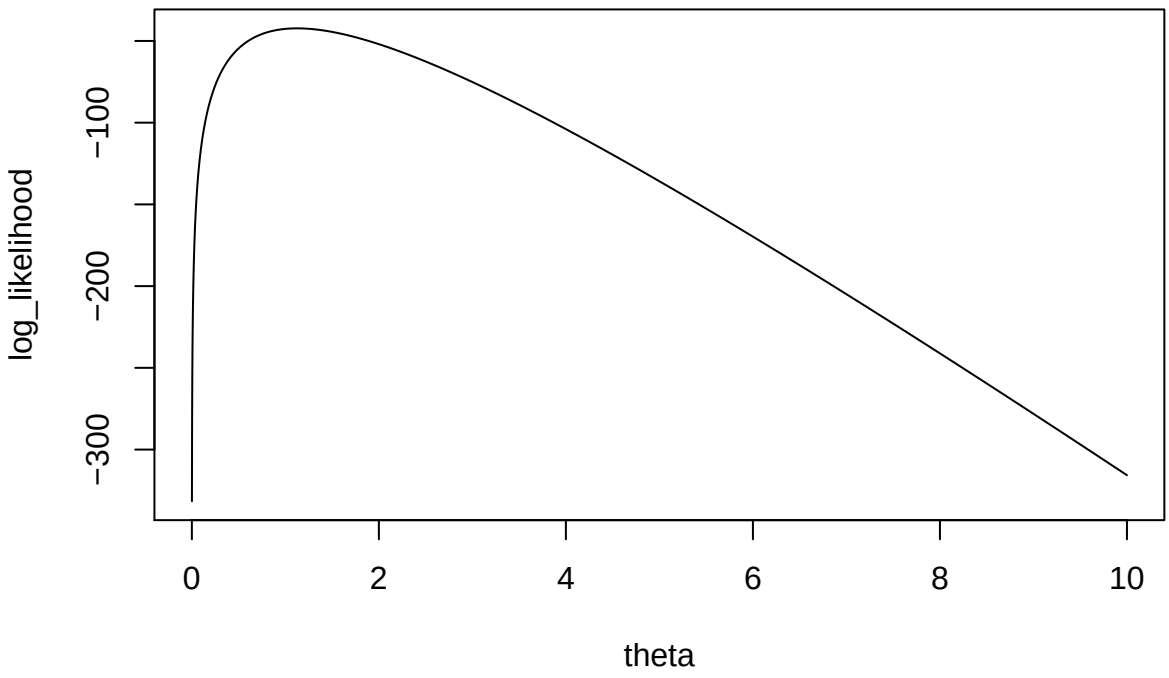
```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.5.3
```

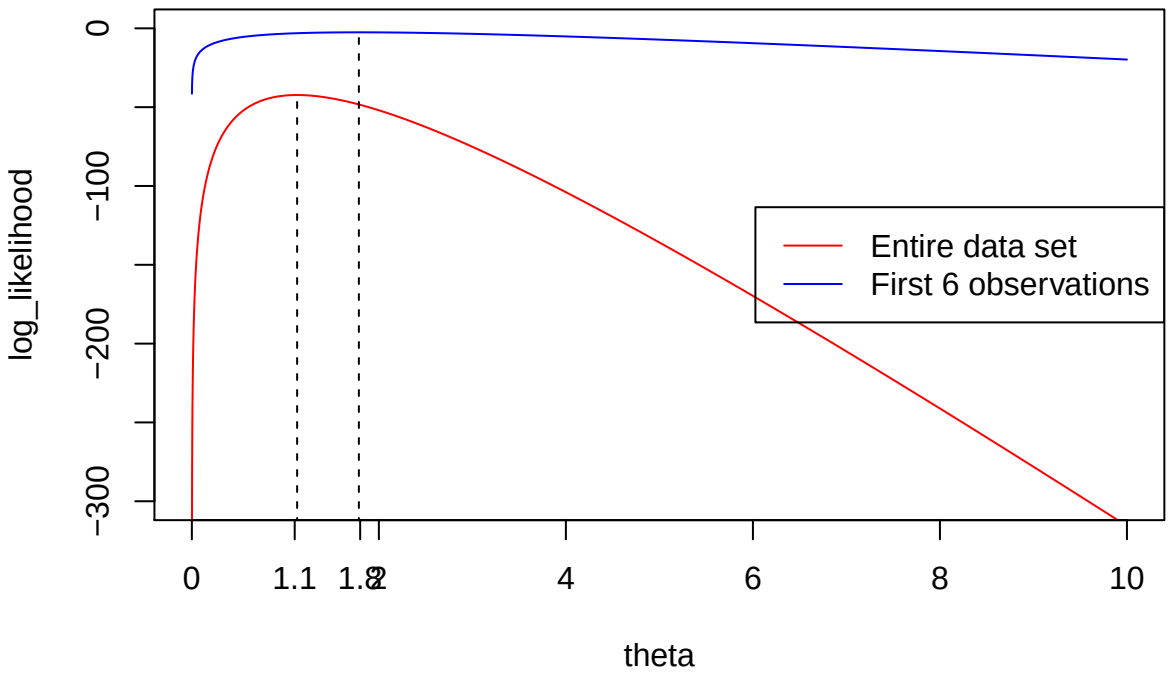
```
data <- read_excel("machines.xlsx")
plot(data$Length)
```



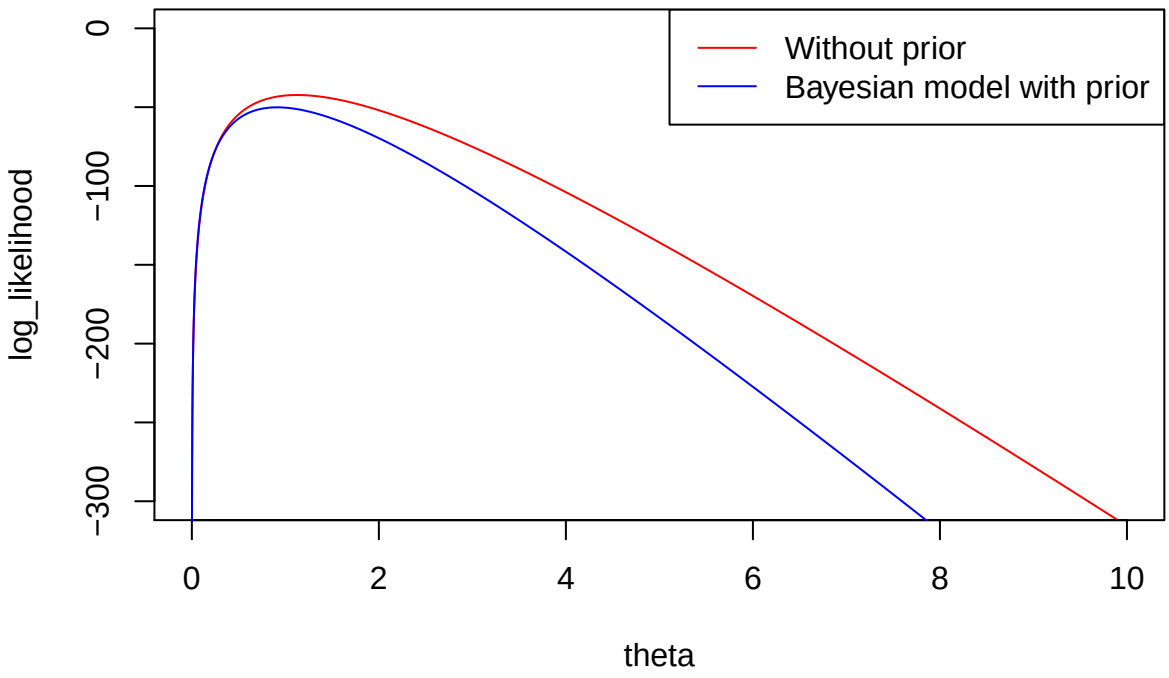

```
length = dim(data)[1]
x = c(data$Length)
theta <- seq(0, 10, 0.001)
log_likelihood <- length*log(theta) - theta*sum(x)
plot(theta, log_likelihood, type = 'l')
```




```
first_6 <- x[1:6]
log_likelihood_2 <- 6*log(theta) - theta*sum(first_6)
plot(theta, log_likelihood, type = 'l', ylim = c(-300,0), col = "red")
lines(theta, log_likelihood_2, type = 'l', col = "blue")
best_theta = theta[which.max(log_likelihood)]
best_theta_2 = theta[which.max(log_likelihood_2)]
lines(c(best_theta, best_theta), c(-1000, max(log_likelihood)), lty=2)
lines(c(best_theta_2, best_theta_2), c(-1000, max(log_likelihood_2)), lty=2)
axis(1, at=round(best_theta_2, 1))
axis(1, at=round(best_theta, 1))
legend("right", legend=c("Entire data set", "First 6 observations"),
col=c("red", "blue"), lty=c(1,1), cex)
```




```
l <- length*log(theta) - theta * sum(x) + log(10) - 10 * theta
plot(theta, log_likelihood, type = 'l', ylim = c(-300,0), col = "red")
lines(theta, l, type = 'l', col="blue")
legend("topright", legend=c("Without prior", "Bayesian model with prior"),
col=c("red", "blue"), lty=c(1,1), cex)
```




```
best_theta2 = theta[which.max(l)]
set.seed(12345)
random <- rexp(50, best_theta)
#hist(random)
```

```
#hist(data$Length)
```