3. ROC Curve

Víctor Villegas, Roger Llorenç, Luis Sierra

2024-03-05

1. Read the SPAM dataset

```
spam <- read.table("spambase/spambase.data",sep=",")

spam.names <- c(read.table("spambase/spambase.names",sep=":",skip=33,nrows=53,as.is=TRUE)[,1],
                "char_freq_#",
                read.table("spambase/spambase.names",sep=":",skip=87,nrows=3,as.is=TRUE)[,1],
                "spam.01")

names(spam) <- spam.names

n<-dim(spam)[1]
p<-dim(spam)[2]-1

spam.01 <- spam[,p+1]
spam.vars <- as.matrix(spam[,1:p])

cat(paste("n = ", n, ', p = ', p,sep=""))
```

```
## n = 4601, p = 57
```

```
cat(paste("Proportion of spam e-mails =", round(mean(spam.01), 2), sep=""))
```

```
## Proportion of spam e-mails =0.39
```

```
glm.spam <- glm(spam.01 ~ spam.vars,family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# summary(glm.spam)
```

2. Train-Test partioning of the data

The manual way to create a train-test split is shown below.

Note the composition of the data, which can be divided as follows:

4601 initial data split into train (2/3) -> 3068 and test (1/3) -> 1533 spam_train (2/3) -> 1209 and nospam_train (2/3) -> 1859 spam_test (1/3) -> 604 and nospam_test (1/3) -> 929

```
set.seed(1234)

spamIndex <- which(spam.01 %in% c(1))
nospamIndex <- which(spam.01 %in% c(0))

spamTrain <- sample(spamIndex, size = ceiling(2/3*length(spamIndex)))
```

```r
nospamTrain <- sample(nospamIndex, size = ceiling(2/3*length(nospamIndex)))
trainIndex <- union(nospamTrain, spamTrain)

trainData <- spam[trainIndex,]
testData <- spam[-trainIndex,]

n <- dim(trainData)[1]
p <- dim(trainData)[2] - 1

Y <- scale(trainData[,p+1], center=FALSE, scale=FALSE)
X <- scale(trainData[,1:p], center=TRUE, scale=TRUE)
```

3. Classification rules

We now consider three methods to obtain classification rules: logistic regression via maximum likelihood, logistic regression via Lasso, and k-nearest neighbors.

Starting with logistic regression via maximum likelihood:

```r
glm.logistic <- glm(spam.01 ~ ., data = as.data.frame(trainData), family = binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
predictions_glm <- predict(glm.logistic, newdata = testData[,1:p], type = "response")

#summary(glm.logistic)
```

Lasso GLM:

```r
library(glmnet)
```

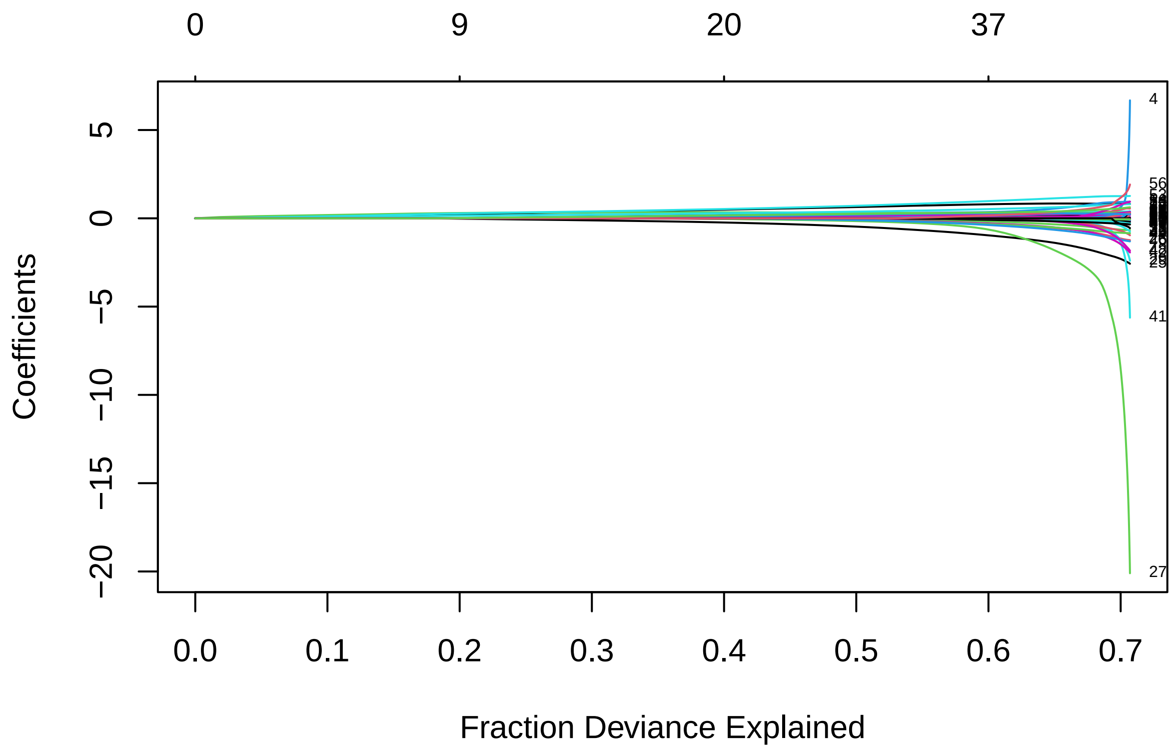
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

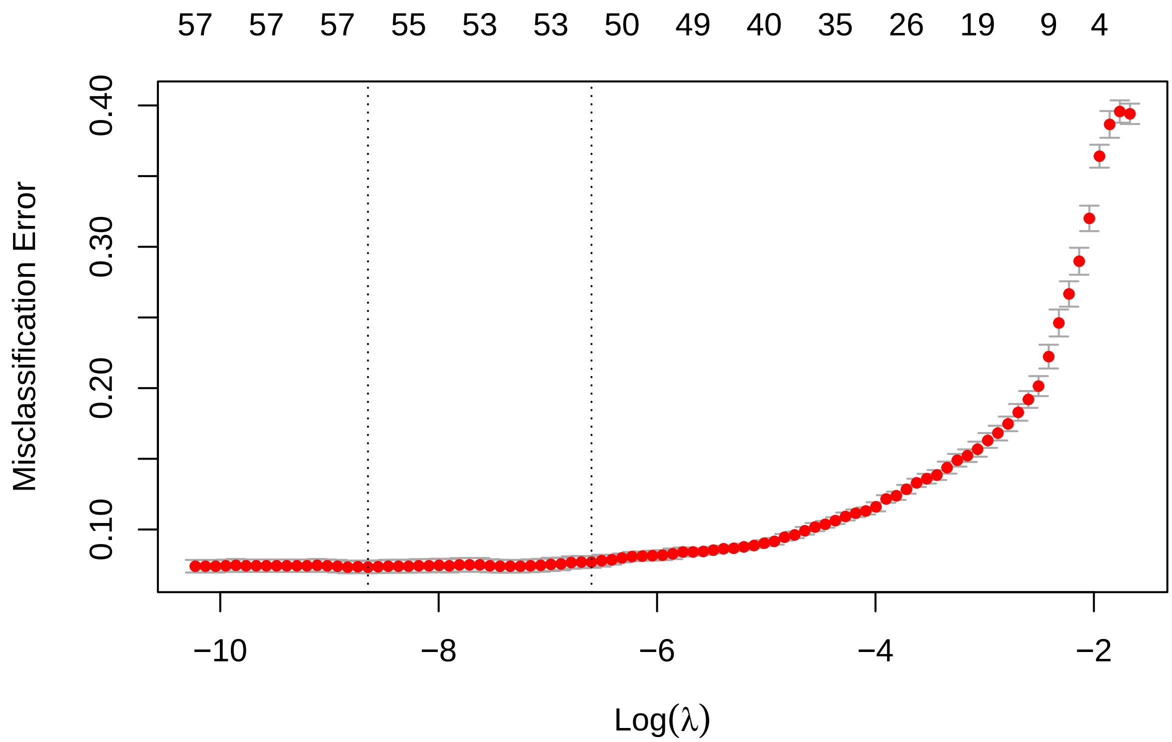
```r
set.seed(1234)

fit_lasso <- glmnet(X, Y, family = "binomial")
cvfit_lasso = cv.glmnet(X, Y, family = "binomial", type.measure = "class")

plot(fit_lasso, xvar = "dev", label = TRUE)
```




```r
plot(cvfit_lasso)
```




```
final_fit_lasso <- glmnet(trainData[,1:p], trainData$spam.01, family = "binomial", lambda = cvfit_lasso$
newx <- model.matrix(testData$spam.01 ~ ., data=testData[,1:p])
predictions_lasso <- predict(final_fit_lasso, newx = newx[,-1], type = "response")
```

k-nearest neighbours:

```
set.seed(1234)

trainData$spam.01 <- as.factor(trainData$spam.01)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knn_fit <- train(spam.01 ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:20),
             trControl  = ctrl,
             data       = trainData)
knn_fit$bestTune #
```

```
##   k
## 1 1
```

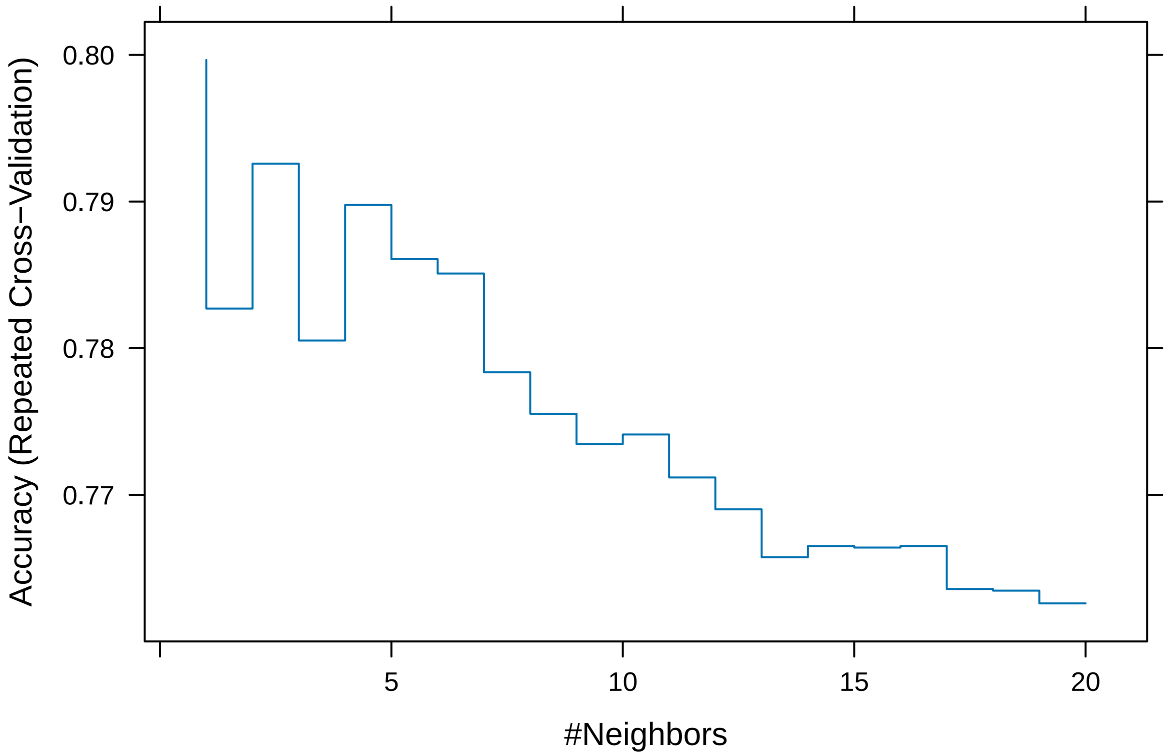
The best performing value for k was $k = 1$.

```
knn_fit
```

```
## k-Nearest Neighbors
##
## 3068 samples
##   57 predictor
```

```
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2761, 2762, 2761, 2761, 2761, 2761, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7996562  0.5797007
##    2  0.7827053  0.5446942
##    3  0.7925855  0.5657786
##    4  0.7805263  0.5399740
##    5  0.7897664  0.5583766
##    6  0.7860715  0.5501987
##    7  0.7850936  0.5479650
##    8  0.7783583  0.5336719
##    9  0.7755314  0.5277080
##   10  0.7734659  0.5233219
##   11  0.7741191  0.5236551
##   12  0.7711886  0.5180970
##   13  0.7690142  0.5137125
##   14  0.7657523  0.5067844
##   15  0.7665148  0.5082876
##   16  0.7664087  0.5087999
##   17  0.7665176  0.5089476
##   18  0.7635836  0.5025951
##   19  0.7634732  0.5025083
##   20  0.7626032  0.5010018
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```r
plot(knn_fit, print.thres = 0.5, type="S")
```




```
predictions_knn <- predict(knn_fit, newdata = testData)
testData$spam.01 <- as.factor(testData$spam.01)
confusionMatrix(predictions_knn, testData$spam.01)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 777 149
##          1 152 455
##
##                Accuracy : 0.8037
##                  95% CI : (0.7829, 0.8233)
##     No Information Rate : 0.606
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5892
##
##  Mcnemar's Test P-Value : 0.9082
##
##             Sensitivity : 0.8364
##             Specificity : 0.7533
##          Pos Pred Value : 0.8391
##          Neg Pred Value : 0.7496
##              Prevalence : 0.6060
##          Detection Rate : 0.5068
```

```
##    Detection Prevalence : 0.6040
##       Balanced Accuracy : 0.7948
##
##        'Positive' Class : 0
##
```

4. ROC Curves

Using the test data we find the following ROC curves:

```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

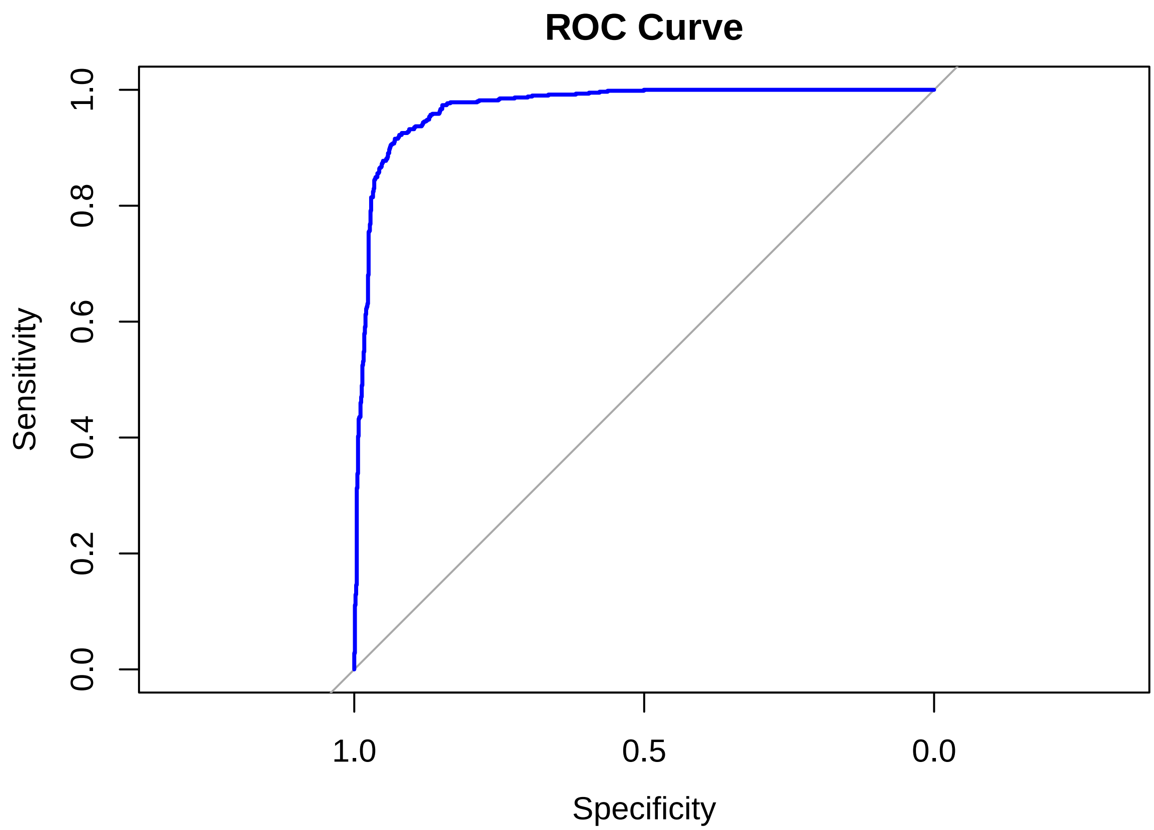
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# glm ROC
roc_curve_glm <- roc(testData[,p+1], predictions_glm)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc_curve_glm, main = "ROC Curve", col = "blue", title="Classic GLM")
```



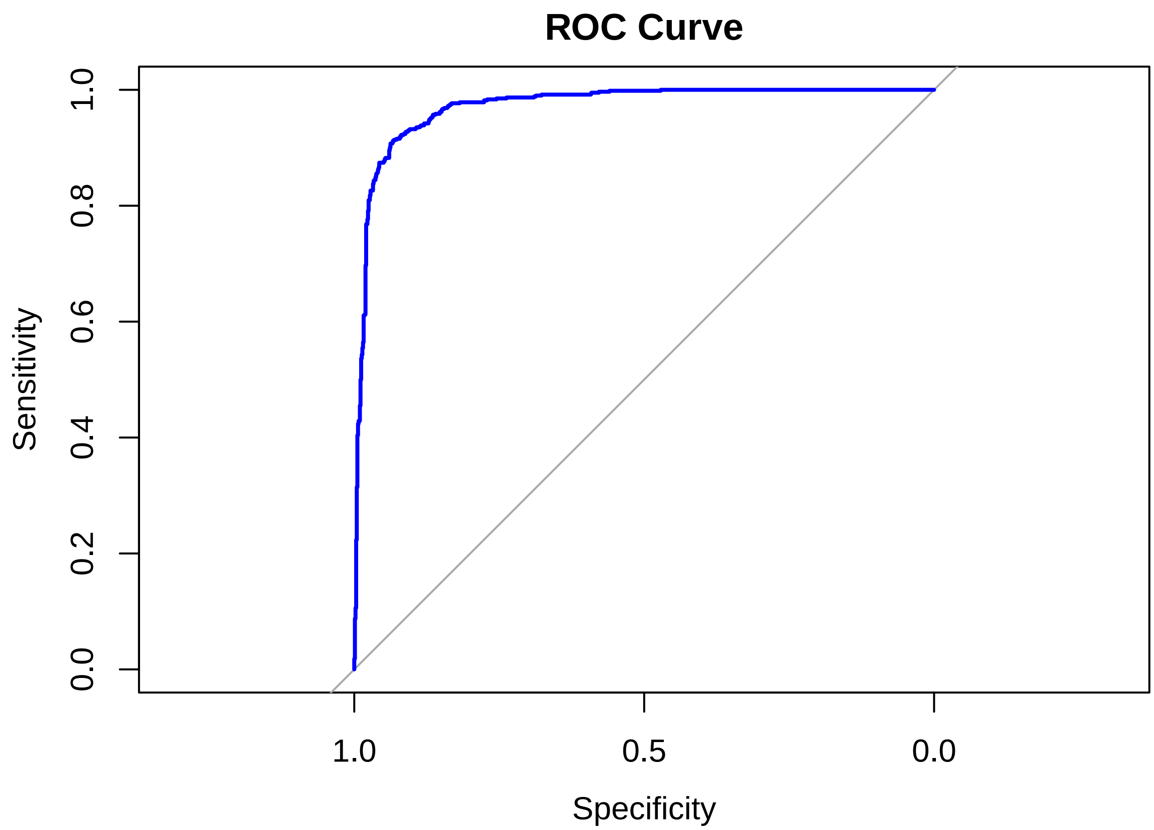

```r
# Lasso ROC
roc_curve_lasso <- roc(testData[,p+1], predictions_lasso)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(testData[, p + 1], predictions_lasso): Deprecated use a
## matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.
```

```
## Setting direction: controls < cases
```

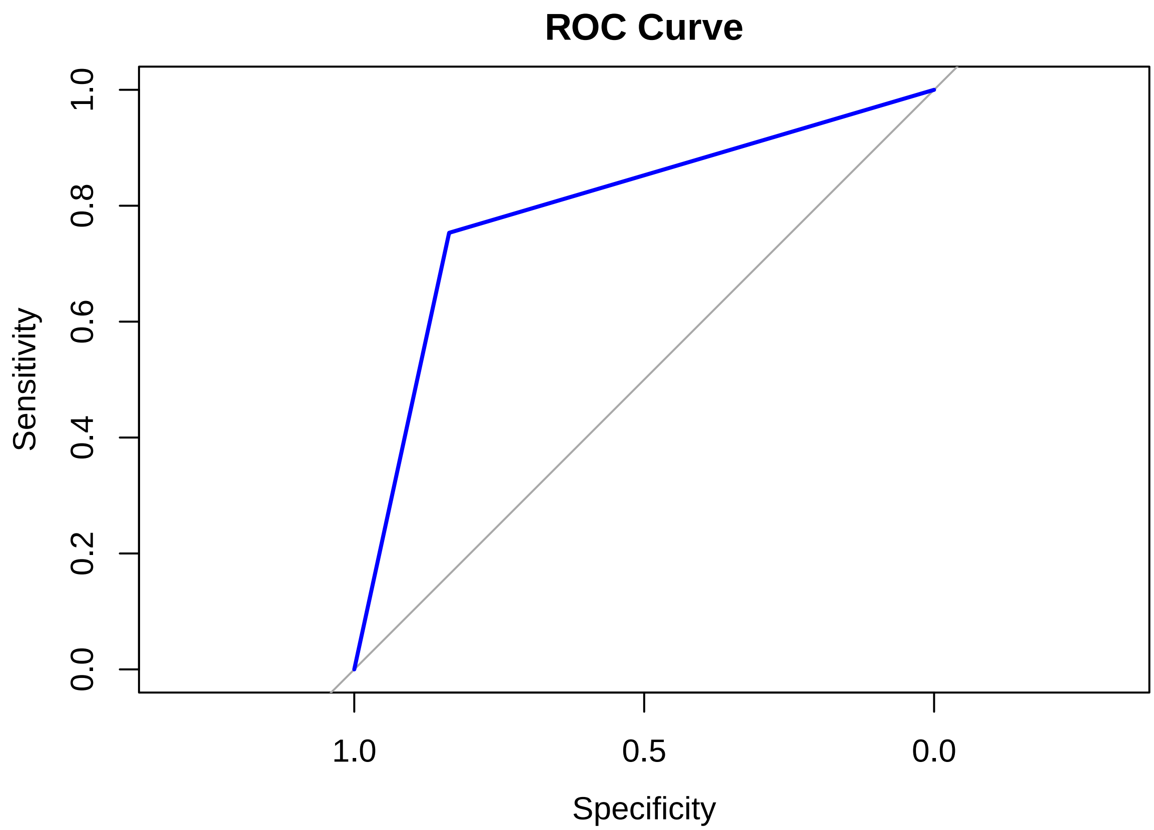
```r
plot(roc_curve_lasso, main = "ROC Curve", col = "blue", title="Lasso GLM")
```




```r
# knn ROC
roc_curve_knn <- roc(testData[,p+1], as.numeric(as.character(predictions_knn)))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(roc_curve_knn, main = "ROC Curve", col = "blue", title="Knn fit")
```



5. Misclassification Rate at $c = \frac{1}{2}$

```
predicted_classes_glm <- ifelse(predictions_glm >= 0.5, 1, 0)
misclassification_rate_glm <- mean(predicted_classes_glm != testData$spam.01)

cat(paste("missclassification rate for GLM using c = 1/2:",misclassification_rate_glm,sep=" "))
```

```
## missclassification rate for GLM using c = 1/2: 0.0802348336594912
```

```
predicted_classes_lasso <- ifelse(predictions_lasso >= 0.5, 1, 0)
misclassification_rate_lasso <- mean(predicted_classes_lasso != testData$spam.01)
cat(paste("missclassification rate for Lasso using c = 1/2:",misclassification_rate_lasso,sep=" "))
```

```
## missclassification rate for Lasso using c = 1/2: 0.076320939334638
```

```
predicted_classes_knn <- ifelse(as.numeric(as.character(predictions_knn)) >= 0.5, 1, 0)
misclassification_rate_knn <- mean(predicted_classes_knn != testData$spam.01)
cat(paste("missclassification rate for 1-nn using c = 1/2:",misclassification_rate_knn,sep=" "))
```

```
## missclassification rate for 1-nn using c = 1/2: 0.19634703196347
```

6. Calculating ℓ_{val}

```
l_val_glm_store <- numeric(dim(testData)[1])

for (j in 1:dim(testData)[1]){
  aux1 = as.numeric(testData$spam.01[j]) * log(predictions_glm[j])
```

```r
  aux2 = (1-as.numeric(testData$spam.01[j])) * log(1-predictions_glm[j])
  l_val_glm_store[j] = aux1 + aux2
  if (is.nan(aux1) | is.nan(aux2) | l_val_glm_store[j] == -Inf ) {
    l_val_glm_store[j] = 0
  }

}
l_val_glm <- mean(l_val_glm_store)

###

l_val_lasso_store = numeric(dim(testData)[1])

for (j in 1:dim(testData)[1]){
  aux1 = as.numeric(testData$spam.01[j]) * log(predictions_lasso[j])
  aux2 = (1-as.numeric(testData$spam.01[j])) * log(1- predictions_lasso[j])
  l_val_lasso_store[j] = aux1 + aux2
  if (is.nan(aux1) | is.nan(aux2) | l_val_lasso_store[j] == -Inf | aux2 == Inf) {
    l_val_lasso_store[j] = 0
  }

}

l_val_lasso <- mean(l_val_lasso_store)

###

preds <- as.numeric(as.character(predictions_knn))

l_val_knn_store <- numeric(dim(testData)[1])

for (j in 1:dim(testData)[1]){
  aux1 = as.numeric(testData$spam.01[j]) * log(preds[j])
  aux2 = (1-as.numeric(testData$spam.01[j])) * log(1-preds[j])
  l_val_knn_store[j] = aux1 + aux2
  if (is.nan(aux1) | is.nan(aux2) | l_val_knn_store[j] == -Inf | l_val_knn_store[j] == Inf) {
    l_val_knn_store[j] = 0
  }

}

l_val_knn <- mean(l_val_knn_store)
```

```r
cat(paste("l_val for classical GLM",l_val_glm,sep=" "))
```

```
## l_val for classical GLM -4.98083188638305
```

```r
cat(paste("l_val for Lasso GLM",l_val_lasso,sep=" "))
```

```
## l_val for Lasso GLM -5.65131876027708
```

```r
cat(paste("l_val for 1-nn",l_val_knn,sep=" "))
```

```
## l_val for 1-nn 0
```