creditCard

Rafael Nogales & Daniel Muñoz

17 de junio de 2016

Predictor de pagos en tarjetas de crédito.

En primer lugar vamos a pasar el archivo xls a csv para ello usamos directamente la herramienta de Excel para exportar el archivo a csv.
Despues ya podemos abrirlo con read.csv
Nota: Podríamos haber utilizado el paquete **gdata** para leer directamente desde Excel pero este metodo no es efectivo con datasets grandes porque la herramienta de lectura de gdata es *muy* lenta para archivos grandes.

```
#Leer datos
creditCardData <- read.csv("~/Desktop/UGR/4-CUARTO/Semestre 2/AprendizajeAutomatico/Credit-Card-Predicto
```

```
#Creamos las particiones de TRAIN y TEST
library("caret")
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
set.seed(2)
train <- createDataPartition(creditCardData$default.payment.next.month,
                                                times = 1, p = 0.8, list = F)
credit.train <- creditCardData[train,]
credit.test <- creditCardData[-train,]
```

Regresion Logistica

```
#Generamos un modelo a partir del train con todas las variables
credit.model <- glm(default.payment.next.month ~ . , data = credit.train)
```

```
prediction <- predict(credit.model, credit.test)
pred.class <- (prediction > 0.5)*1
t.glm<- table(predict=pred.class, truth=credit.test$default.payment.next.month)
t.glm
```

```
##        truth
## predict    0    1
##       0 4636 1091
##       1   81  192
```

```
error.glm <- 1 - sum(diag(t.glm))/sum(t.glm)
error.glm
```

```
## [1] 0.1953333
```

Regresion Logistica rocplot

```
library("ROCR")
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.2.4
```

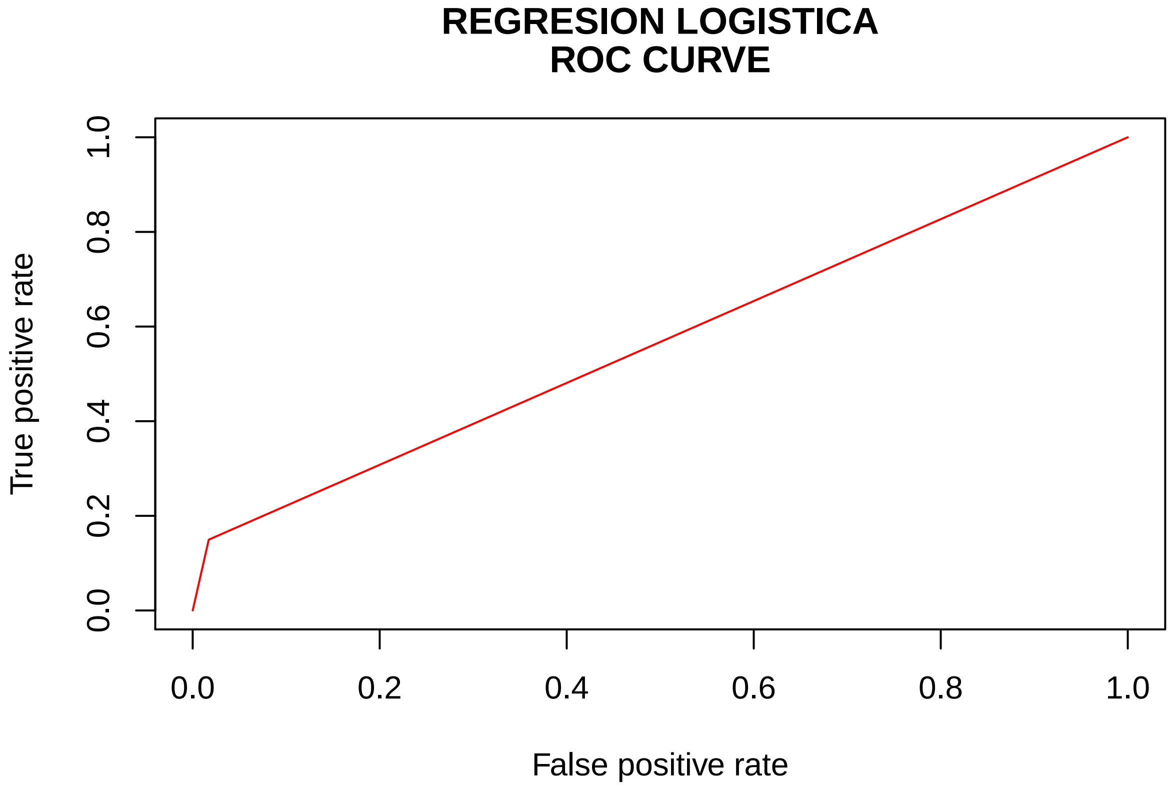
```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
rocplot <- function(pred, truth, ...){
    if(class(pred) == "factor"){
        pred <- as.numeric(pred)
        #Ahora tenemos un vector de 1's y 2's
        #Lo pasamos a -1 y 1
        pred <- 2*pred-3
    }
    predob <- prediction(pred, truth)
    perf <- performance(predob, "tpr", "fpr")

    plot(perf, ...)
}
```

```
log.predict <- pred.class             #Solo es renombrar
rocplot(log.predict, credit.test$default.payment.next.month, col="red", main=c("REGRESION LOGISTICA", "
```



Validacion cruzada Regresion Logistica

```r
cv.error.rl <- function(k=5){
    errores <- vector(length = k)
    for(i in 1:k){
        labels <- creditCardData$default.payment.next.month
        train <- createDataPartition(labels, times = 1, p = 0.8, list = F)

        credit.train <- creditCardData[train,]
        credit.test <- creditCardData[-train,]

        labels.train <- credit.train$default.payment.next.month
        labels.test  <- credit.test$default.payment.next.month

        credit.model <- glm(default.payment.next.month ~ . , data = credit.train)
        summary(credit.model)
        prediction <- predict(credit.model, credit.test)



        pred.class <- (prediction > 0.5)*1
        t.glm<- table(predict=pred.class, truth=credit.test$default.payment.next.month)

        error <- 1 - sum(diag(t.glm))/sum(t.glm)
        errores[i] <- error
    }
    return(mean(errores))
}
```

```
error_glm=cv.error.rl(k=5)
error_glm
```

```
## [1] 0.2038
```

Regresion Lineal

```
#Generamos un modelo a partir del train con todas las variables
credit.model <- lm(default.payment.next.month ~ . , data = credit.train)
```

```
prediction <- predict(credit.model, credit.test)
pred.class <- (prediction > 0.5)*1
t.lm<- table(predict=pred.class, truth=credit.test$default.payment.next.month)
t.lm
```

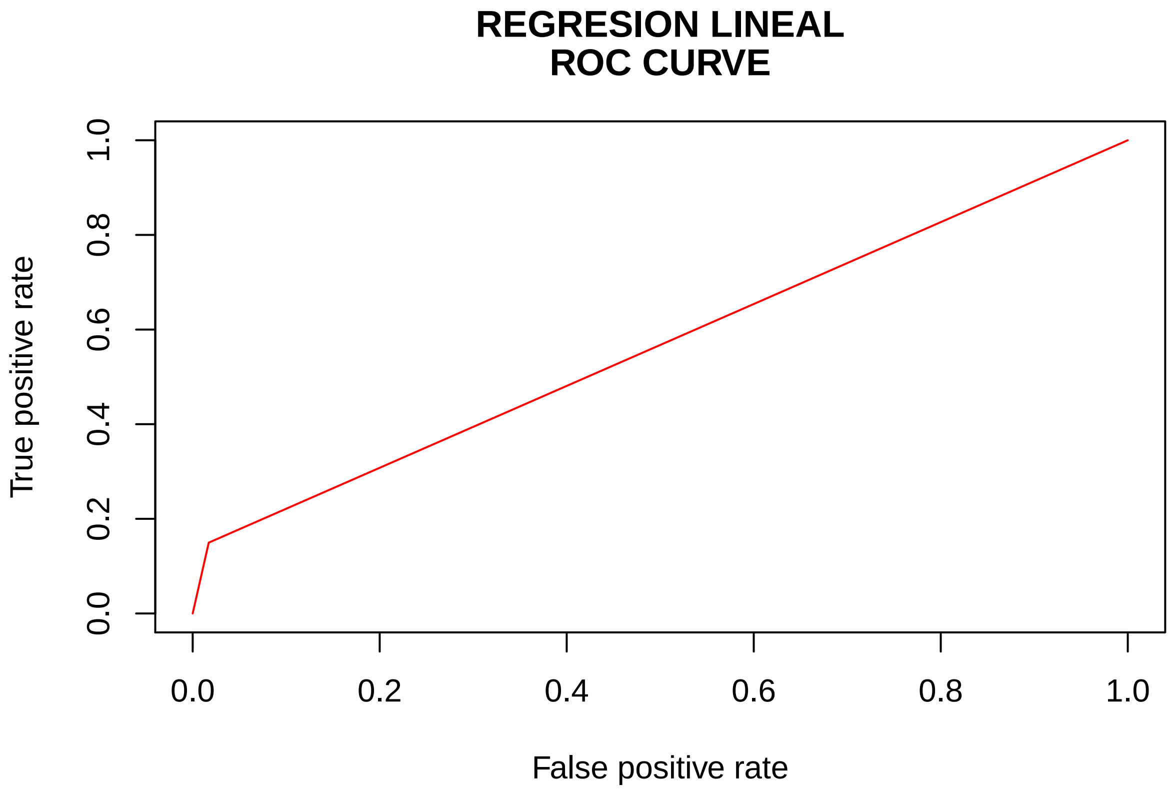
```
##         truth
## predict    0    1
##       0 4636 1091
##       1   81  192
```

```
error.lm <- 1 - sum(diag(t.lm))/sum(t.lm)
error.lm
```

```
## [1] 0.1953333
```

Regresion Lineal rocplot

```
lm.predict <- pred.class             #Solo es renombrar
rocplot(lm.predict, credit.test$default.payment.next.month, col="red", main=c("REGRESION LINEAL", "ROC C
```



Validacion cruzada Regresion Lineal

```
cv.error.lm <- function(data, k=5){
    errores <- vector(length = k)
    for(i in 1:k){
        labels <- creditCardData$default.payment.next.month
        train <- createDataPartition(labels, times = 1, p = 0.8, list = F)

        credit.train <- creditCardData[train,]
        credit.test <- creditCardData[-train,]

        labels.train <- credit.train$default.payment.next.month
        labels.test  <- credit.test$default.payment.next.month

        credit.model <- lm(default.payment.next.month ~ . , data = credit.train)
        summary(credit.model)
        prediction <- predict(credit.model, credit.test)


        pred.class <- (prediction > 0.5)*1
        t.lm<- table(predict=pred.class, truth=credit.test$default.payment.next.month)

        error <- 1 - sum(diag(t.lm))/sum(t.lm)
        errores[i] <- error
    }
    return(mean(errores))
}
```

```
error_lm=cv.error.lm(k=5)
error_lm
```

```
## [1] 0.1998333
```

SVM

```
library("e1071")
tune_cost.credit.svm <- tune(svm, default.payment.next.month ~ .,
                             data=credit.train, kernel="linear",
                             ranges=list(cost=c(0.001, 0.1, 100)),
                             scale = FALSE)
```

regresion lineal con weight-decay

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.4

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

```
modelo_ridge_cv=cv.glmnet(as.matrix(credit.train),credit.train$default.payment.next.month,alpha=0)
mejor_lambda=modelo_ridge_cv$lambda.min
modelo=glmnet(as.matrix(credit.train),credit.train$default.payment.next.month,alpha=0)
pesos=predict(modelo,type = "coefficients",s=mejor_lambda)
pesos=pesos[1:nrow(pesos)]
pesos
```

```
##  [1]  3.539088e-02 -1.685655e-08 -1.284587e-08 -1.477630e-03 -1.572165e-03
##  [6] -2.716504e-03  1.500642e-04  9.192683e-03  2.250067e-03  1.405712e-03
## [11]  9.309094e-04  5.494397e-04  5.299083e-05 -3.000015e-08 -1.058361e-08
## [16] -5.967652e-09 -3.551251e-09  6.314833e-09  3.078338e-09 -5.697875e-08
## [21] -3.459285e-08 -1.508222e-08 -3.578486e-08 -3.181865e-08 -1.197789e-08
## [26]  8.892417e-01
```

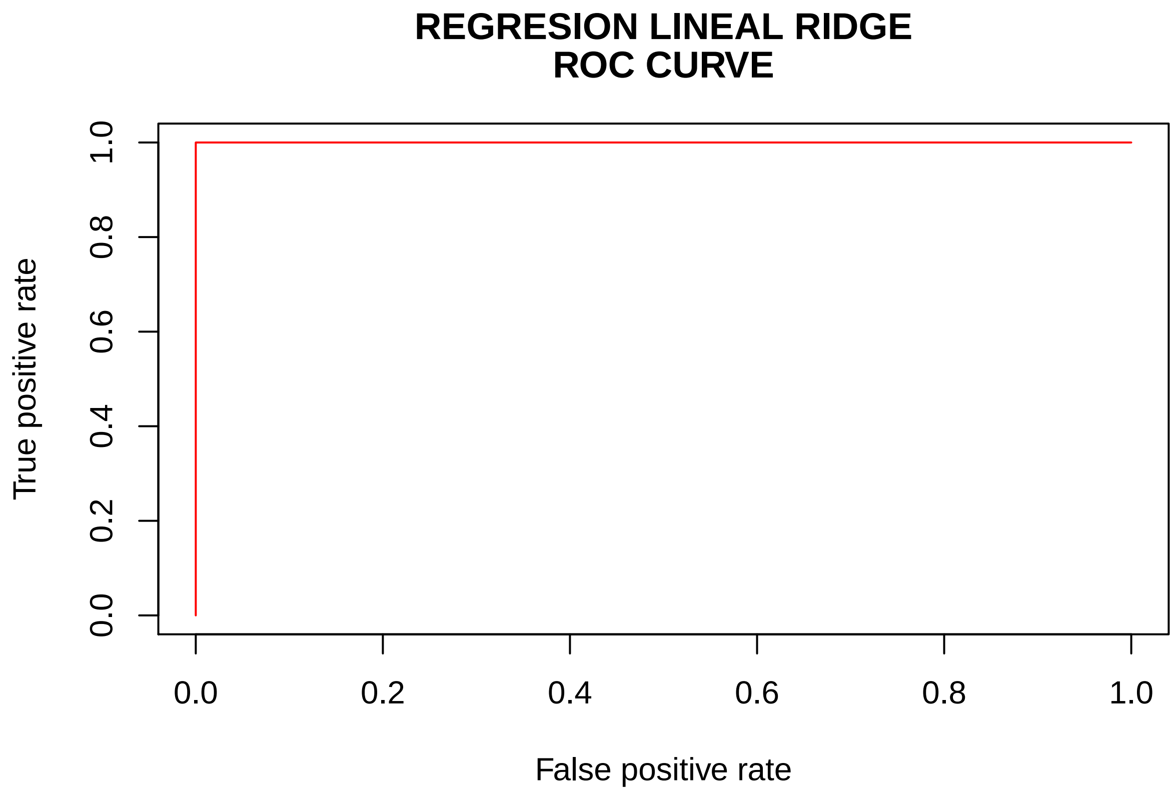
```
prediction=predict(modelo,s=mejor_lambda,newx = as.matrix(credit.test))
pred.class <- (prediction > 0.5)*1
t_ridge<- table(predict=pred.class, truth=credit.test$default.payment.next.month)
t_ridge
```

```
##         truth
## predict    0    1
##       0 4717    0
##       1    0 1283
```

```
error.ridge<- 1 - sum(diag(t_ridge))/sum(t_ridge)
error.ridge
```

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

```
rid.predict <- pred.class             #Solo es renombrar
rocplot(rid.predict, credit.test$default.payment.next.month, col="red", main=c("REGRESION LINEAL RIDGE"
```



Validacion cruzada Regresion Lineal con weight-decay (Ridge)

```
cv.error.rid <- function(data, k=5){
    errores <- vector(length = k)
    for(i in 1:k){
        labels <- creditCardData$default.payment.next.month
        train <- createDataPartition(labels, times = 1, p = 0.8, list = F)

        credit.train <- creditCardData[train,]
        credit.test <- creditCardData[-train,]

        labels.train <- credit.train$default.payment.next.month
        labels.test  <- credit.test$default.payment.next.month

        modelo_ridge_cv <- cv.glmnet(as.matrix(credit.train),
                                     credit.train$default.payment.next.month, alpha=0)
        mejor_lambda <- modelo_ridge_cv$lambda.min
        modelo <- glmnet(as.matrix(credit.train),credit.train$default.payment.next.month,alpha=0)
        prediction <- predict(modelo,s=mejor_lambda,newx = as.matrix(credit.test))
        pred.class <- (prediction > 0.5)*1


        t.rid<- table(predict=pred.class, truth=credit.test$default.payment.next.month)

        error <- 1 - sum(diag(t.rid))/sum(t.rid)
        errores[i] <- error
    }

    return(mean(errores))
}
```

```
error_rid = cv.error.rid(k=5)
error_rid
```

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

Random forest

```
library("randomForest")
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library("gbm")
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1
```

```
library("ipred")
model_rf <- randomForest(credit.train$default.payment.next.month ~.,data = credit.train, ntree=50)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

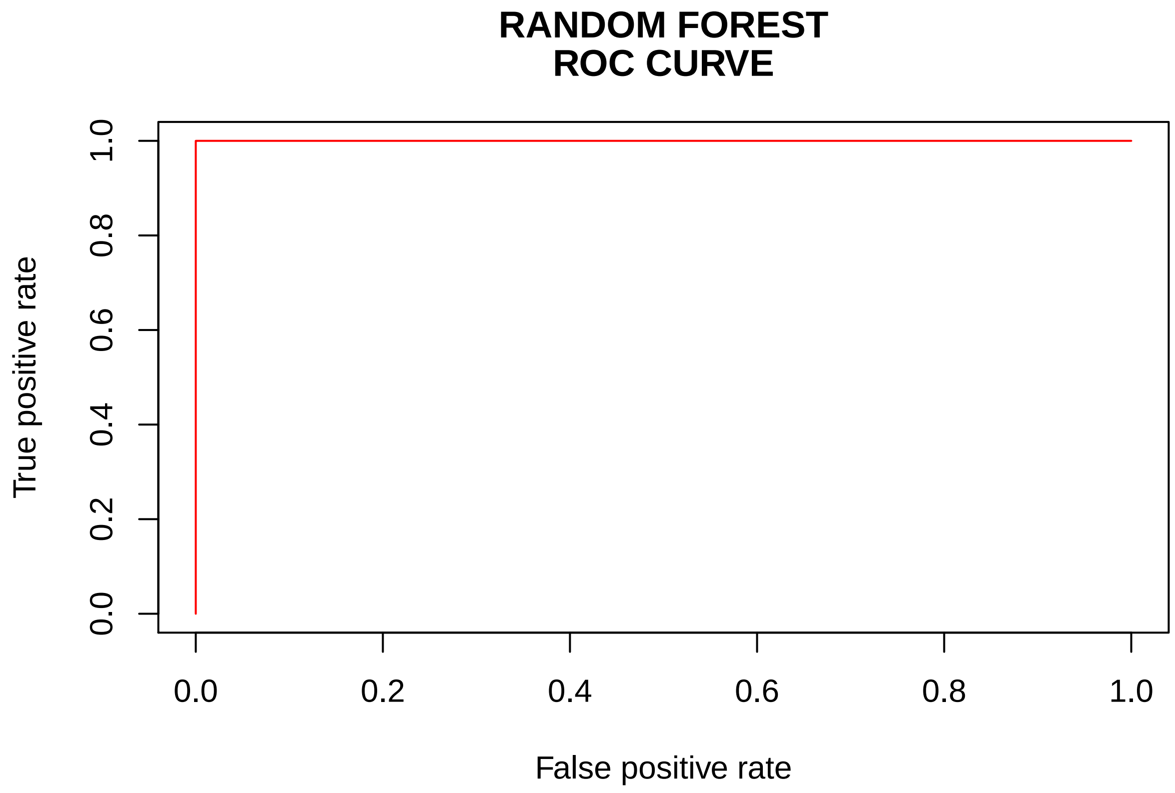
```
predicion <- predict(model_rf, newdata=credit.test)
pred.class <- (prediction > 0.5)*1
t_random.forest <- table(predict=pred.class, truth=credit.test$default.payment.next.month)
t_random.forest
```

```
##          truth
## predict     0    1
##         0 4717    0
##         1    0 1283
```

```
error.random.forest<- 1 - sum(diag(t_random.forest))/sum(t_random.forest)
error.random.forest
```

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

```
rid.predict <- pred.class              #Solo es renombrar
rocplot(rid.predict, credit.test$default.payment.next.month, col="red", main=c("RANDOM FOREST", "ROC CU
```



Validacion cruzada Regresion Lineal con weight-decay (Ridge)

```
cv.error.rf <- function(k=5, model){
    errores <- vector(length = k)
    for(i in 1:k){
        labels <- creditCardData$default.payment.next.month
        train <- createDataPartition(labels, times = 1, p = 0.8, list = F)

        credit.train <- creditCardData[train,]
        credit.test <- creditCardData[-train,]

        labels.train <- credit.train$default.payment.next.month
        labels.test  <- credit.test$default.payment.next.month

        prediction <- predict(model,newdata = credit.test)
        pred.class <- (prediction > 0.5)*1

        t<- table(predict=pred.class, truth=credit.test$default.payment.next.month)

        error <- 1 - sum(diag(t))/sum(t)
        errores[i] <- error
    }

    return(mean(errores))
}
```

```
error_rf = cv.error.rf(k=5, model= model_rf)
error_rf
```

```
## [1] 0.04263333
```

KNN

```
library("ipred")
library("class")
library("e1071")
set.seed(1)
mejor_k=tune.knn(x=credit.train,y=as.logical(credit.train$default.payment.next.month),
                 k=1:10,tunecontrol=tune.control(sampling = "cross"), cross=10)
mejor_k
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   k
##  10
```

```
##
## - best performance: 0.2394167
```

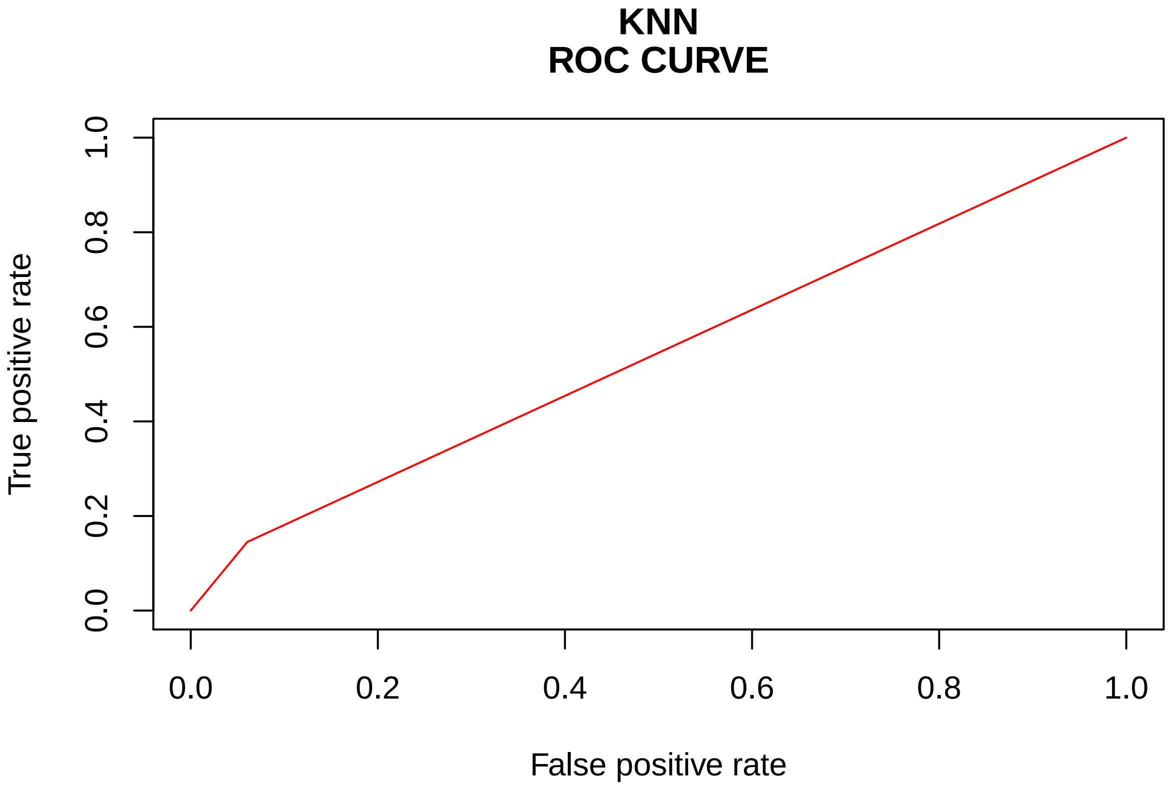
```
k <- mejor_k$best.parameters
knn.pred <- knn(credit.train,credit.test,credit.train$default.payment.next.month,k=k[1,1])
knn.pred <- as.numeric(knn.pred)-1
t_knn<- table(predict = knn.pred, truth=credit.test$default.payment.next.month)
t_knn
```

```
##        truth
## predict    0    1
##       0 4432 1097
##       1  285  186
```

```
error.knn<- 1 - sum(diag(t_knn))/sum(t_knn)
error.knn
```

```
## [1] 0.2303333
```

```
rocplot(knn.pred, credit.test$default.payment.next.month, col="red", main=c("KNN", "ROC CURVE"))
```



Validacion cruzada KNN

```
cv.error.knn <- function(k=5, mejor_k){
    errores <- vector(length = k)
    for(i in 1:k){
        labels <- creditCardData$default.payment.next.month
        train <- createDataPartition(labels, times = 1, p = 0.8, list = F)

        credit.train <- creditCardData[train,]
        credit.test <- creditCardData[-train,]

        labels.train <- credit.train$default.payment.next.month
        labels.test  <- credit.test$default.payment.next.month

        prediction <- knn(credit.train,credit.test,
                          credit.train$default.payment.next.month,k=mejor_k)
        prediction <- as.numeric(prediction)-1

        t<- table(predict=prediction, truth=credit.test$default.payment.next.month)

        error <- 1 - sum(diag(t))/sum(t)
        errores[i] <- error
    }

    return(mean(errores))
}
```

```
error_knn = cv.error.knn(k=5, mejor_k = 9)
error_knn
```

```
## [1] 0.2383333
```