Ejercicios Clasificación

Antonio Manuel Milán Jiménez

29 de noviembre de 2018

Diagnóstico de cáncer de mama con k-NN

Para este conjunto de datos en el que tendremos que determiar si un tumor es benigno o maligno, utilizaremos un modelo k-NN con diferentes elecciones de "k".

Primero cargamos los datos:

```
wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
```

Preprocesamiento

Eliminamos la característica "id":

```
wbcd <- wbcd[,-1]
```

Reconvertimos la variables de diagnosis a un "factor":

```
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"), labels = c("Benign", "Malignant"))
```

Podemos saber la proporción de los datos en cuanto a casos beningnos y malignos. Lo ideal sería una proporción de 50/50.

```
round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
```

```
##
##    Benign Malignant
##      62.7      37.3
```

A continuación vamos a normalizar los datos:

```
wbcd_n <- as.data.frame(lapply(wbcd[,2:31], scale, center = TRUE, scale = TRUE))
```

Preparación de conjuntos de entrenamiento y test

```
shuffle_ds <- sample(dim(wbcd_n)[1])
eightypct <- (dim(wbcd_n)[1] * 80) %/% 100
wbcd_train <- wbcd_n[shuffle_ds[1:eightypct], ]
wbcd_test <- wbcd_n[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], ]

wbcd_train_labels <- wbcd[shuffle_ds[1:eightypct], 1]
wbcd_test_labels <- wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], 1]
```

Obtención del modelo y resultado

Vamos ya a crear el modelo. Crearemos diversos modelos utilizando diferentes "k" para descubrir cómo varía el acierto del modelo en función de esto.

```
library(class)
require(caret)
```

```
## Loading required package: caret
```

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

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

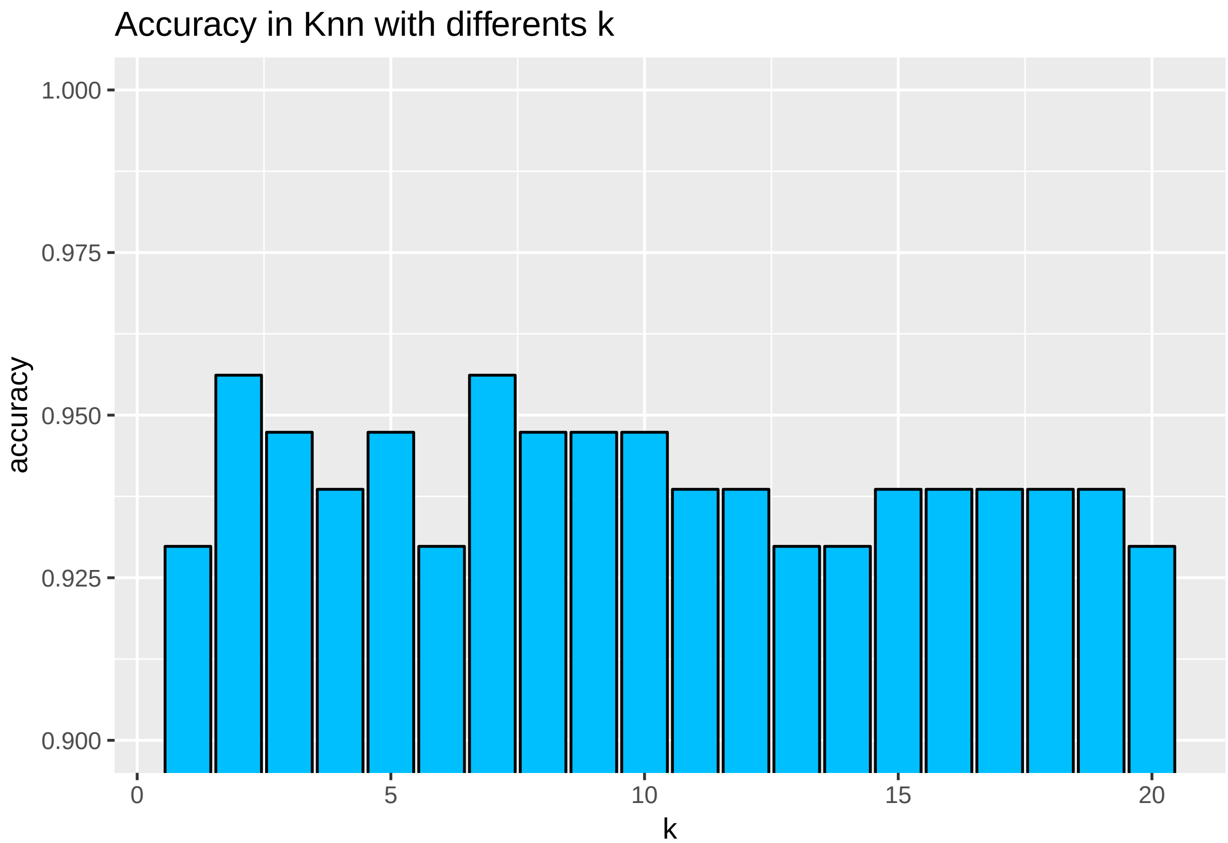
```
getKnn <- function(miK=1){

  wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels, k=miK)

  postResample(pred = wbcd_test_pred, obs = wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], 1])
}
result <- lapply(1:20,getKnn)
result<-unlist(result)[1:20*2-1]

df <- data.frame(k=1:20,accuracy=result)

ggplot(df, aes(x=k,y=accuracy)) + geom_histogram(stat="identity",color="black",fill="deepskyblue")+coor
```



Con este gráfico descubrimos que se comporta mejor el modelo para valores de "k" más bajos (3,5,6,8,9) llegando al 95.61% de acierto.

Regresión logística con datos de "Stock Market"

Vamos a trabajar con el dataset "The Stock Market" con el que, utilizando diferentes variables relacionadas con información de la Bolsa, se trata de predecir si la Bolsa "sube" o "baja".

```
library(ISLR)
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
##  Median :2003   Median : 0.039000   Median : 0.039000
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
##       Lag3                Lag4                Lag5
##  Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200
##  1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000
##  Median : 0.038500   Median : 0.038500   Median : 0.03850
##  Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561
##  3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700
##  Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300
##      Volume           Today           Direction
##  Min.   :0.3561   Min.   :-4.922000   Down:602
##  1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648
##  Median :1.4229   Median : 0.038500
##  Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   :3.1525   Max.   : 5.733000
```

Estudiando las correlaciones de las variables con la variable de salida encontramos, por ejemplo, que la variable "Today" es interesante.

```
cor(as.numeric(Smarket$Direction),Smarket$Today)
```

```
## [1] 0.7305629
```

Vamos a utilizar el paquete "caret" para realizar la regresión logística.

```
require(caret)
```

Se va a comparar los resultados que obtenemos al utilizar únicamente un subconjunto del dataset cómo "train" para entrenar el modelo y al utilizar todo el conjunto de datos para construir el modelo. Dado que el dataset no está predividido en "train" y "test", una forma posible de hacerlo es en función del año del dato. De esta forma:

```
train <- (Smarket$Year < 2005)
test <- (Smarket$Year == 2005)
```

Otro punto interesante es que se va a realizar "cross-validation" con 10 particiones. Para trabajar con el paquete "caret" lo hacemos de la siguiente forma:

```
train_control = trainControl(method="cv",number=10)
```

Ya sí pasamos a entrenar los modelos utilizando todas las variables:

```
glmFit <- train(Smarket[train,-9], y = Smarket[train,9], method = "glm", preProcess = c("center", "scal
glmFit
```

```
## Generalized Linear Model
##
## 998 samples
##   8 predictors
##   2 classes: 'Down', 'Up'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 898, 898, 899, 898, 898, 899, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9959899  0.9919768
```

```
glm.pred <- predict.train(glmFit,newdata=Smarket[test,])
Direction.2005 <- Smarket$Direction[test]
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down  110   0
##     Up      1 141
```

```
mean(glm.pred==Direction.2005)
```

```
## [1] 0.9960317
```

Se obtiene en ambos casos más del 99% de acierto lo que se traduce en que se ha conseguido un buen modelo, no ha habido sobreajuste al utilizar cross-validation y que se ha hecho una buena división para el conjunto de test pues presenta un resultado muy similar.

Trabajando ahora con todo el conjuto de datos para crear el modelo:

```
glmFit <- train(Smarket[,-9], y = Smarket[,9], method = "glm", preProcess = c("center", "scale"), tuneL
glmFit
```

```
## Generalized Linear Model
##
## 1250 samples
##    8 predictors
##    2 classes: 'Down', 'Up'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1126, 1124, 1125, 1124, 1125, 1125, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9943934  0.9887774
```

Se obtiene un resultado muy similar, tan sólo un 0.3~0.4% inferior; por lo que podría interesarnos utilizar cómo modelo el anterior que sólo se entreno con los datos de "train", pues es ligeramente superior su resultado y algo más eficiente.

QDA y LDA con dataset "Stock Market"

Vamos a realizar ahora una comparación entre estos dos algoritmos utilizando únicamente las variables "Lag".

```
library(MASS)
library(ISLR)
lda.fit <- lda(Direction~Lag1+Lag2+Lag3+Lag4+Lag5,data=Smarket, subset=Year<2005)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5, data = Smarket,
##     subset = Year < 2005)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2         Lag3         Lag4          Lag5
## Down  0.04279022  0.03389409 -0.009806517 -0.010598778  0.0043665988
## Up   -0.03954635 -0.03132544  0.005834320  0.003110454 -0.0006508876
##
## Coefficients of linear discriminants:
##              LD1
## Lag1 -0.63046918
## Lag2 -0.50221745
## Lag3  0.10142974
## Lag4  0.09725317
## Lag5 -0.03685767
```

Ahora realizamos la predicción:

```
Smarket.2005 <- subset(Smarket,Year==2005)
lda.pred <- predict(lda.fit,Smarket.2005)
```

Y finalmente obtenemos los resultados:

```
table(lda.pred$class,Smarket.2005$Direction)
```

```
##
##        Down  Up
##   Down   37  30
##   Up     74 111
```

```
resultLDA <- mean(lda.pred$class==Smarket.2005$Direction)
resultLDA
```

```
## [1] 0.5873016
```

Se obtiene un acierto del 58.7%

Probamos ahora el mismo experimento para el algoritmo QDA:

```
qda.fit <- qda(Direction~Lag1+Lag2+Lag3+Lag4+Lag5, data=Smarket, subset=Year<2005)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5, data = Smarket,
```

```
##     subset = Year < 2005)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2         Lag3         Lag4          Lag5
## Down  0.04279022  0.03389409 -0.009806517 -0.010598778  0.0043665988
## Up   -0.03954635 -0.03132544  0.005834320  0.003110454 -0.0006508876
```

Realizamos la predicción y obtenemos finalmente los resultados:

```
qda.pred <- predict(qda.fit,Smarket.2005)
table(qda.pred$class,Smarket.2005$Direction)
```

```
##
##        Down  Up
##   Down   37  35
##   Up     74 106
```

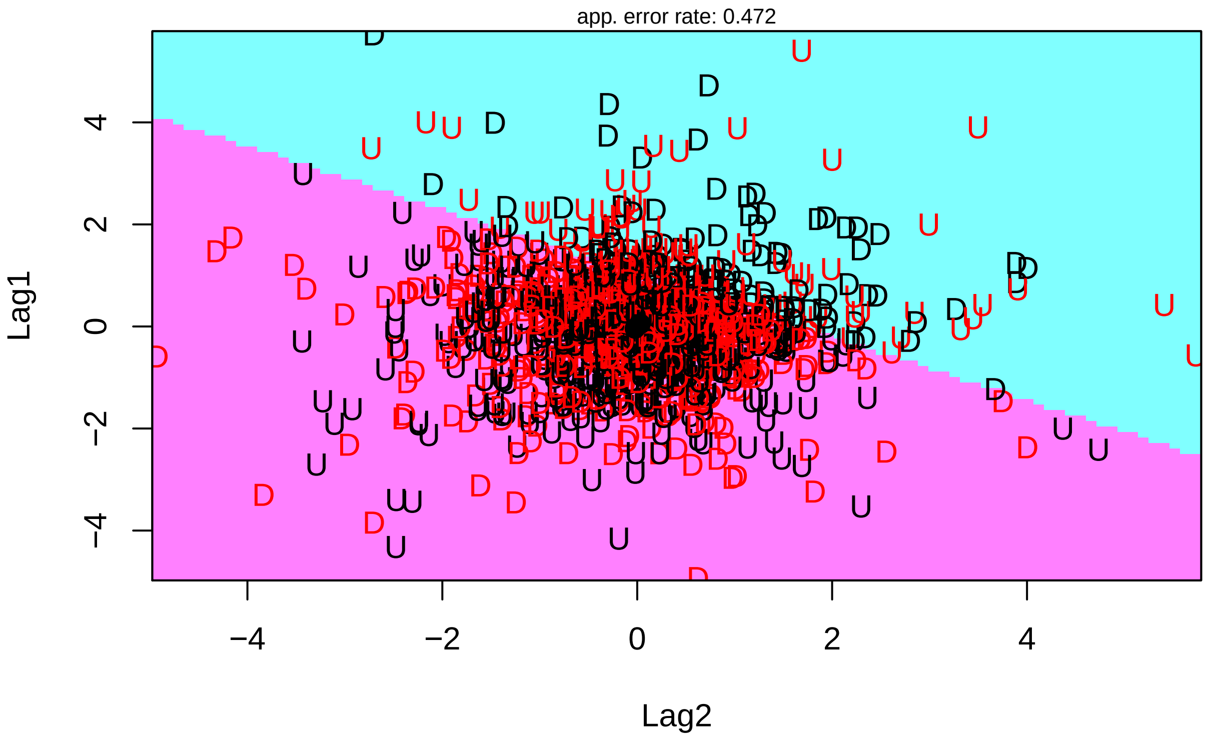
```
resultQDA <- mean(qda.pred$class==Smarket.2005$Direction)
resultQDA
```

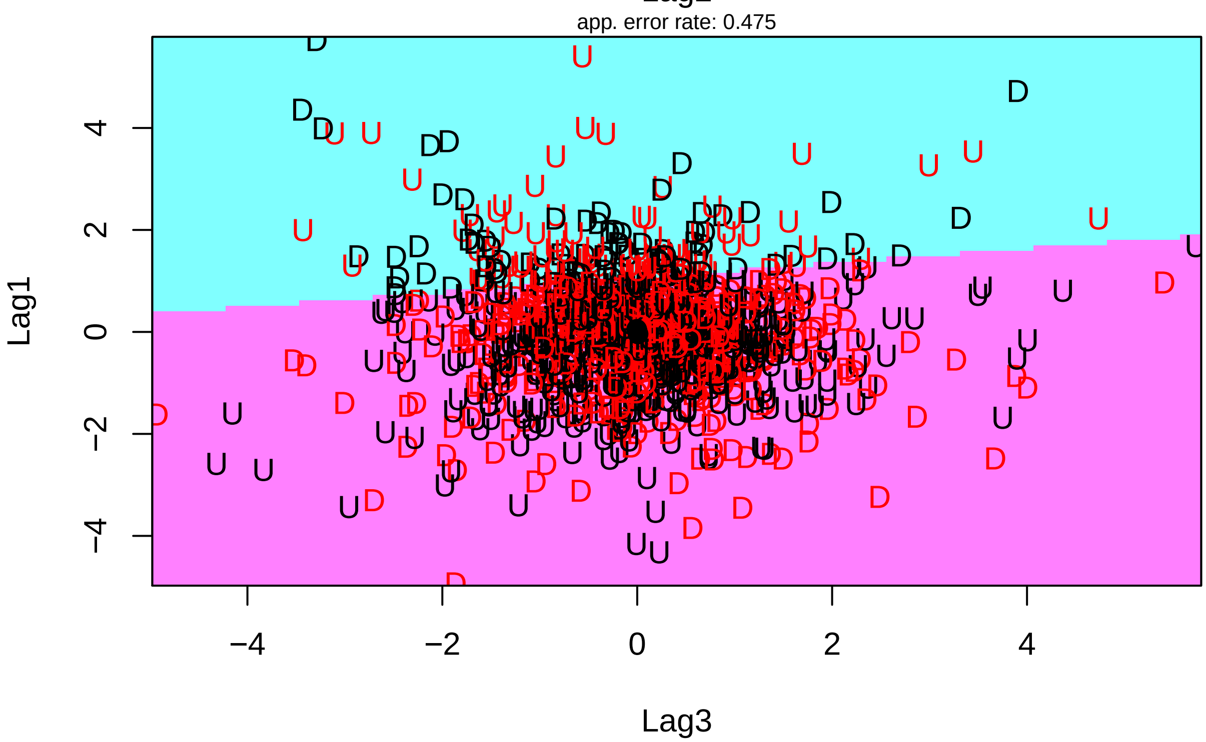
```
## [1] 0.5674603
```

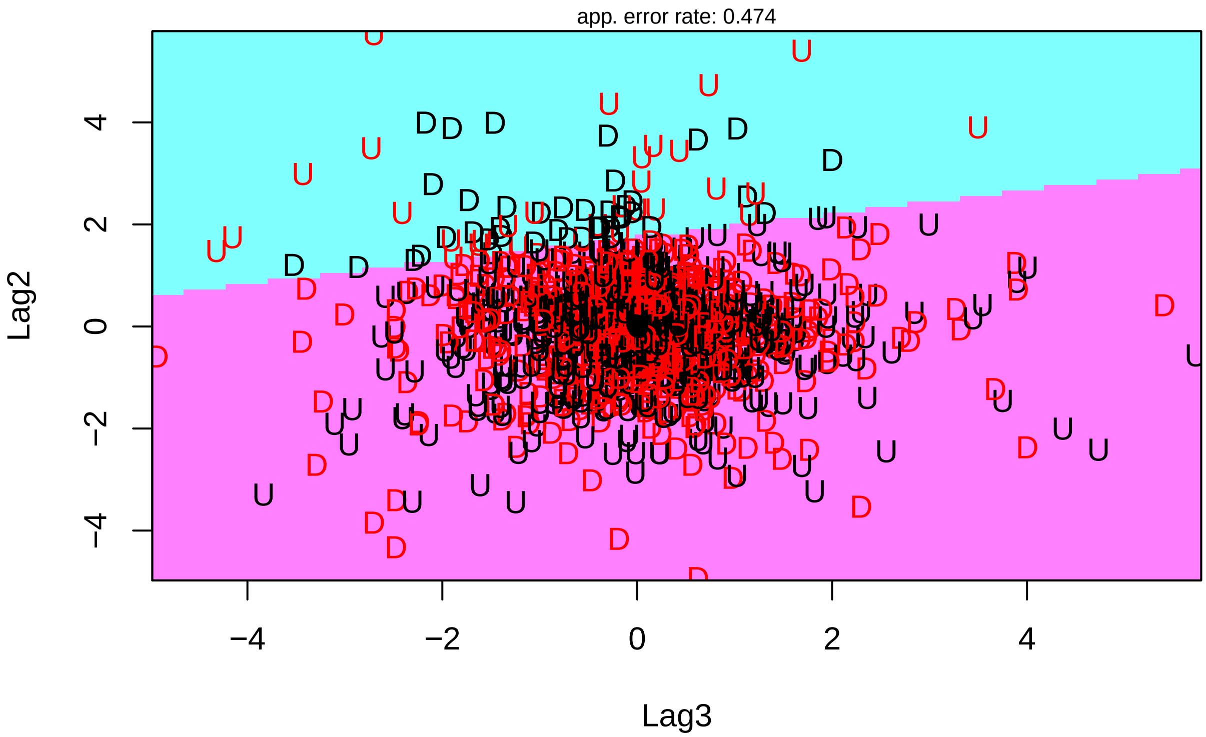
Se obtiene un acierto del 56.7%, 2 puntos por debajo de lo que conseguimos con lda.

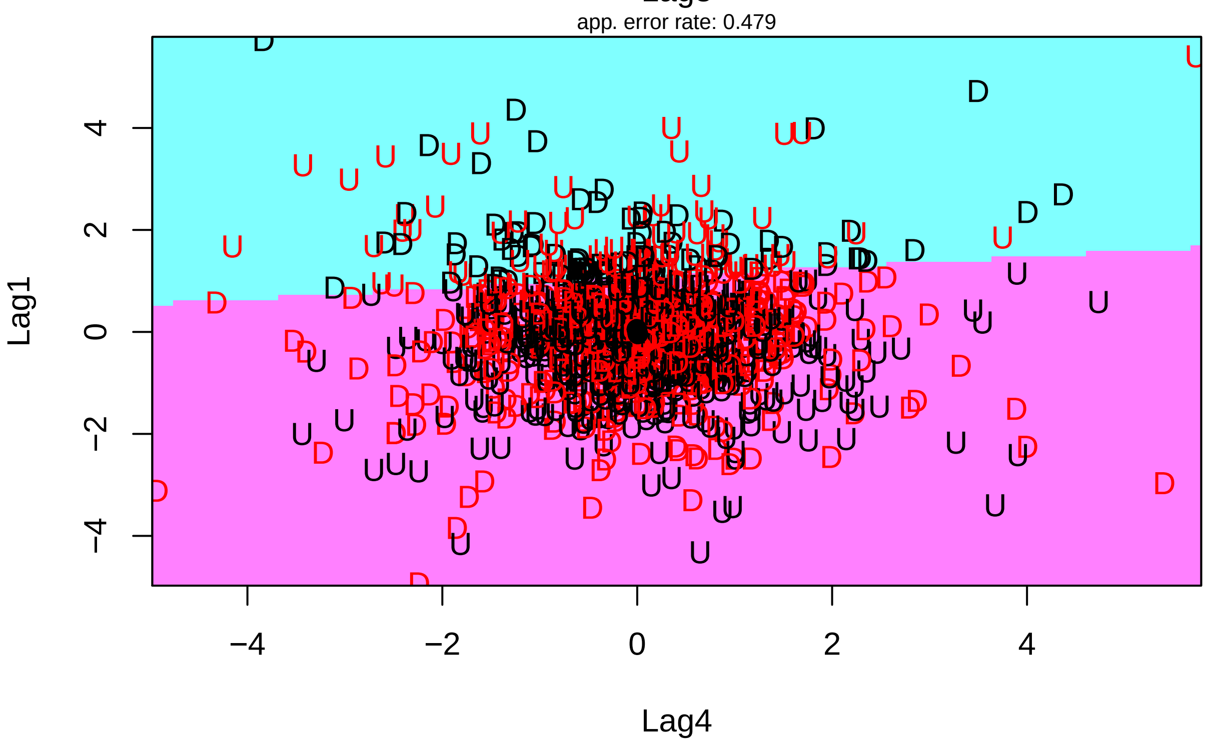
Podemos incluso observar cómo han realizado la partición de ambos algoritmos, enfrentando 1 vs 1 las 5 variables. Estos son los ajustes para LDA:

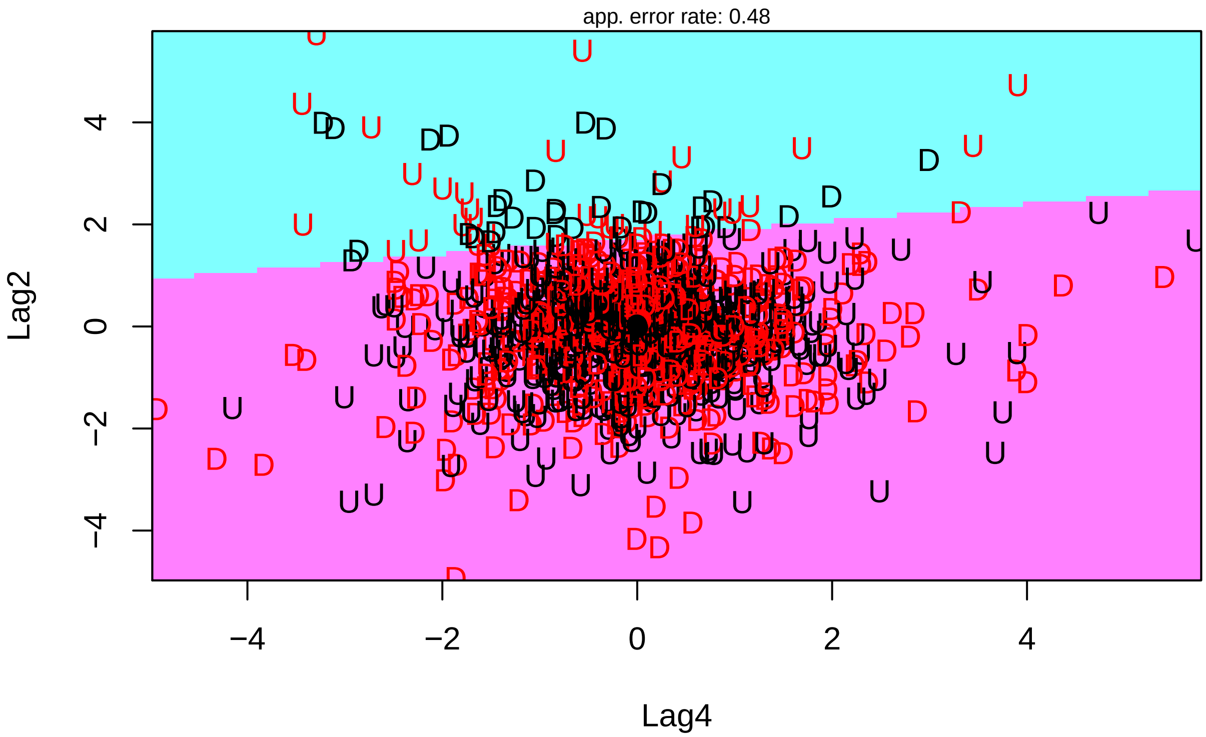
```
library(klaR)
partimat(Direction~Lag1+Lag2+Lag3+Lag4+Lag5, data=Smarket, method="lda",nplots.vert=1,nplots.hor=1)
```


app. error rate: 0.472
Lag1
Lag2
−4
−2
0
2
4

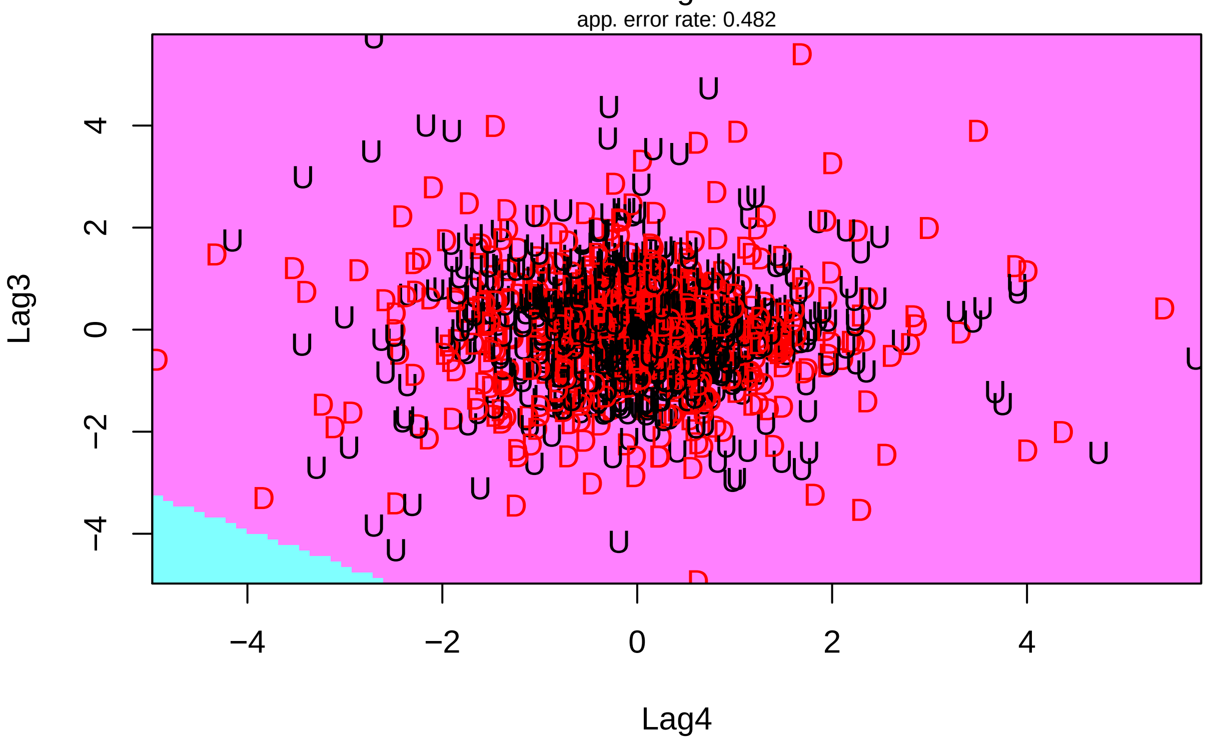

app. error rate: 0.475
Lag1
Lag3
−4
−2
0
2
4


app. error rate: 0.474
Lag2
Lag3
−4
−2
0
2
4

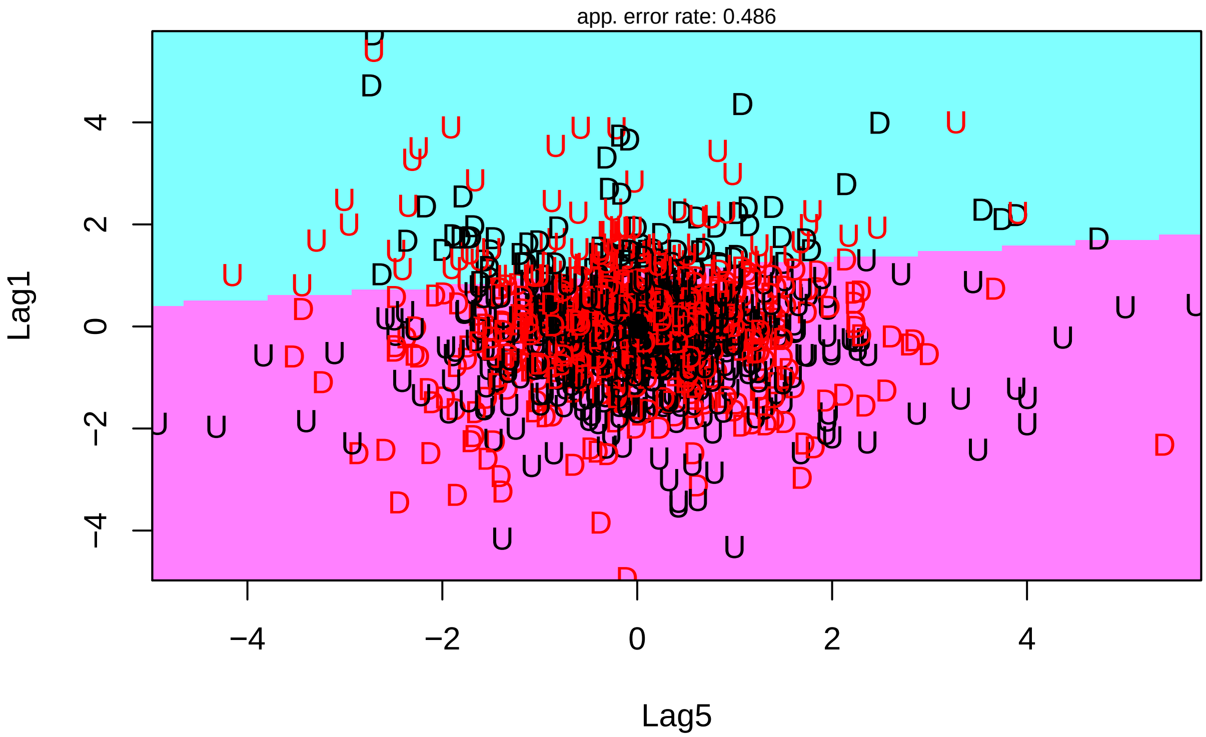

app. error rate: 0.479
Lag1
Lag4
−4
−2
0
2
4

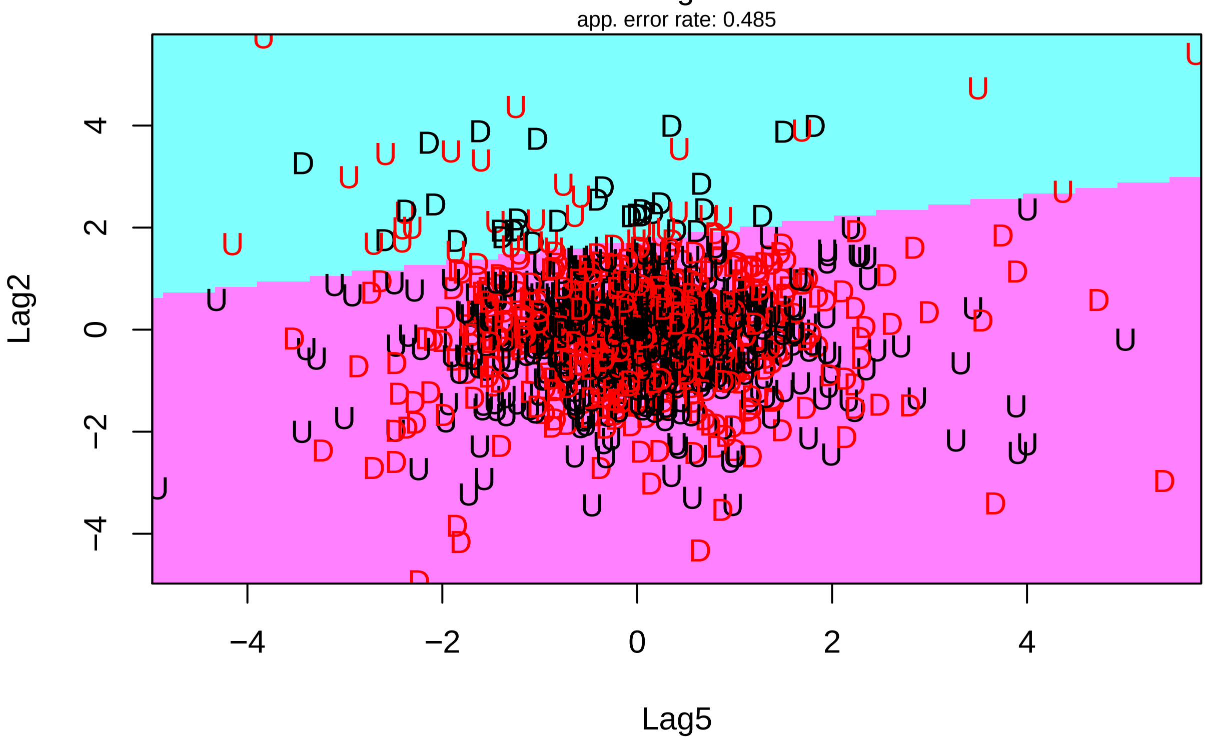


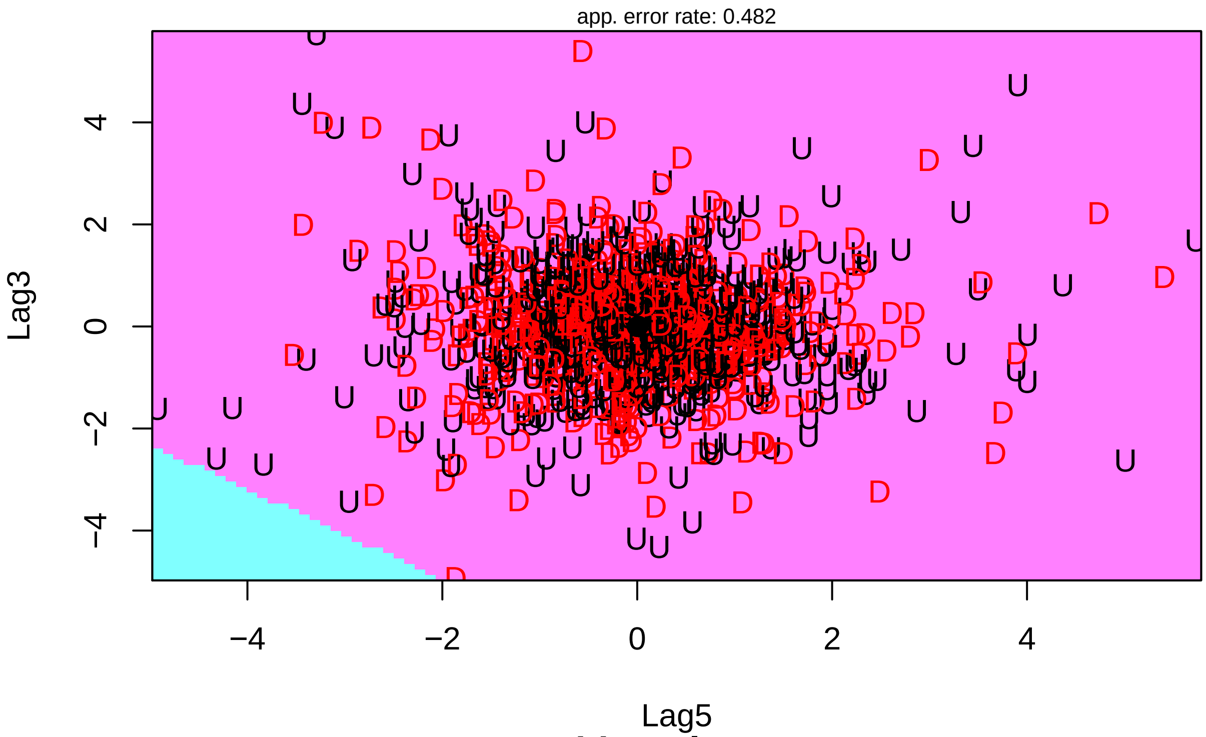
app. error rate: 0.48
Lag2
Lag4

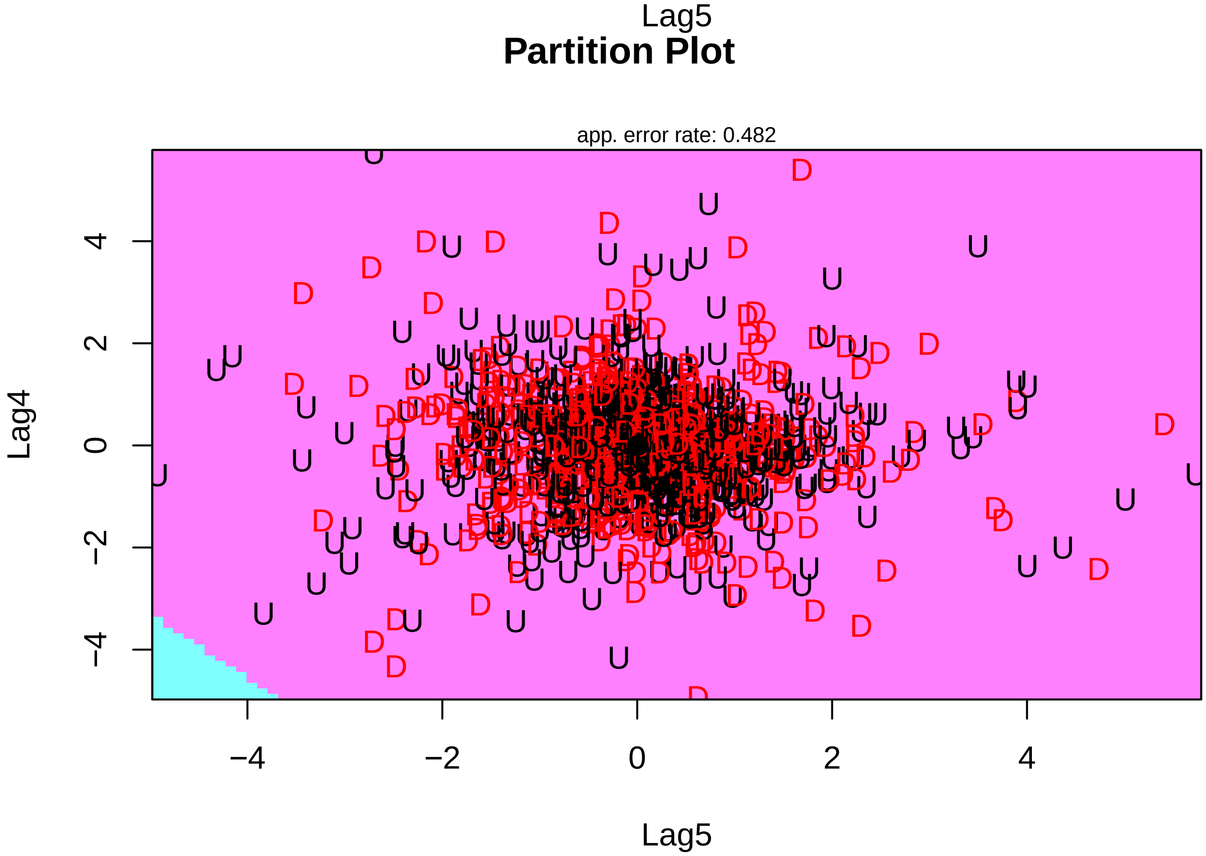


app. error rate: 0.482
Lag3
Lag4


app. error rate: 0.486
Lag1
Lag5

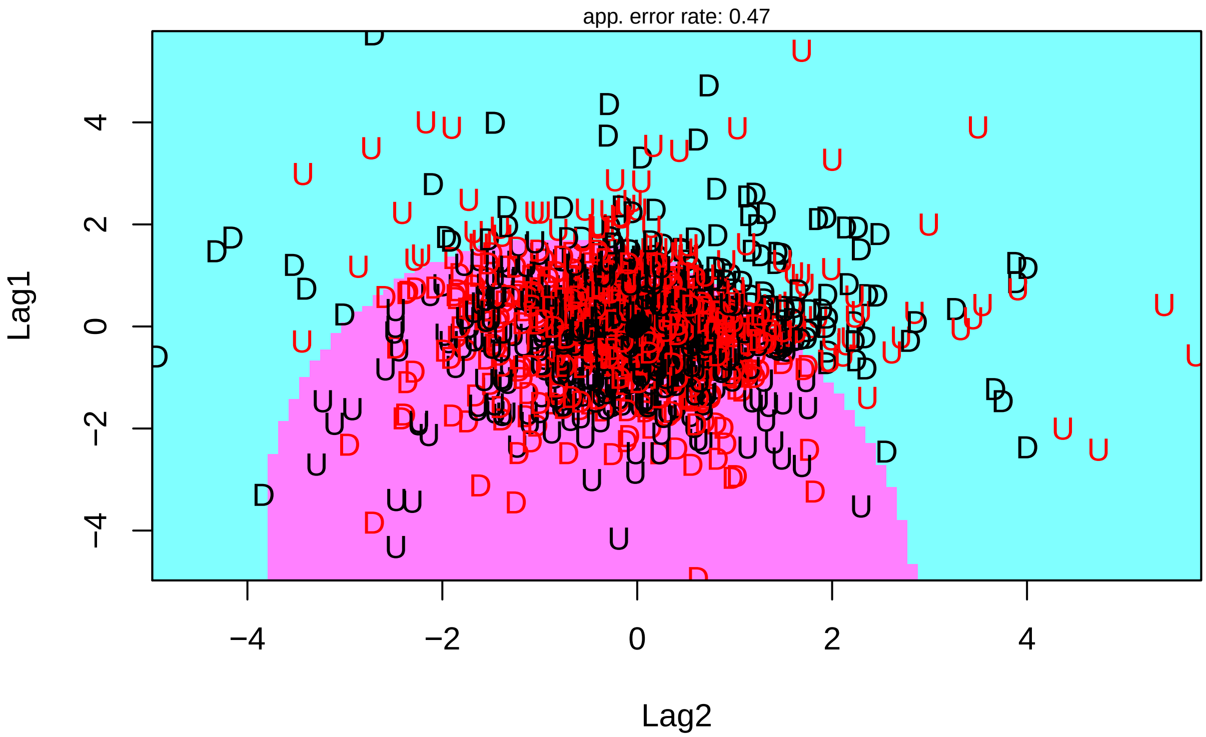

app. error rate: 0.485
Lag2
Lag5

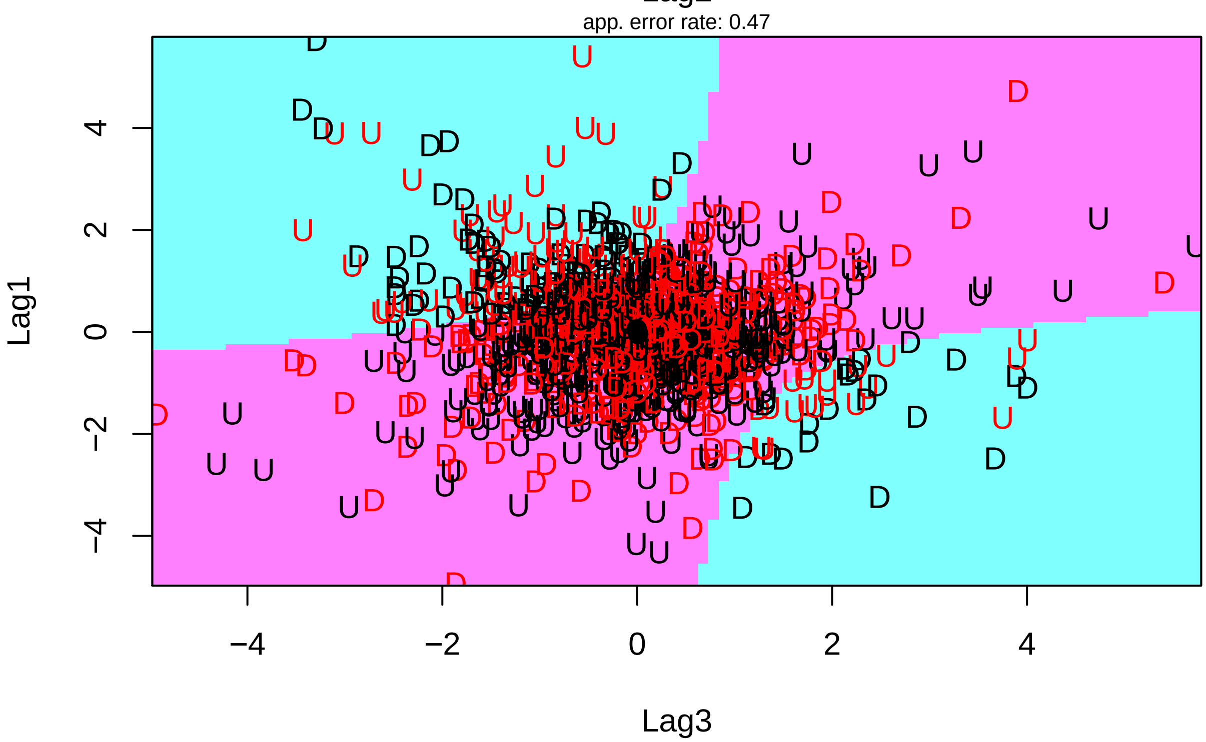


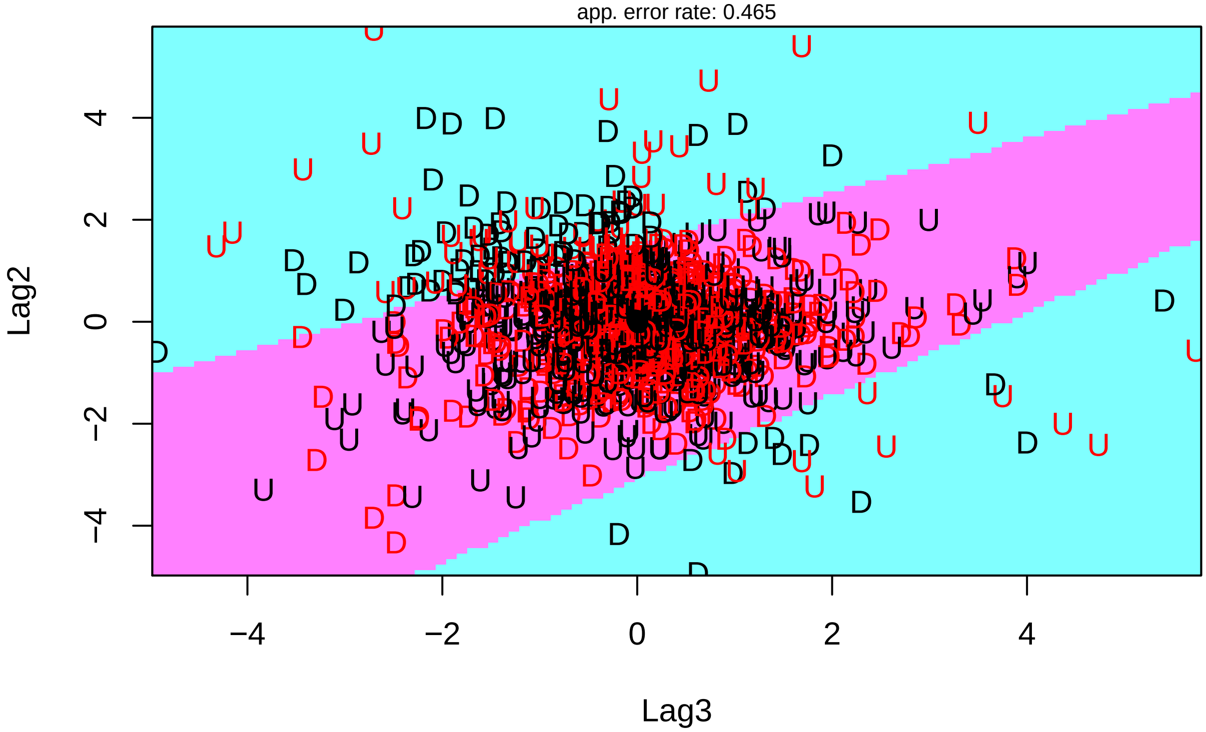


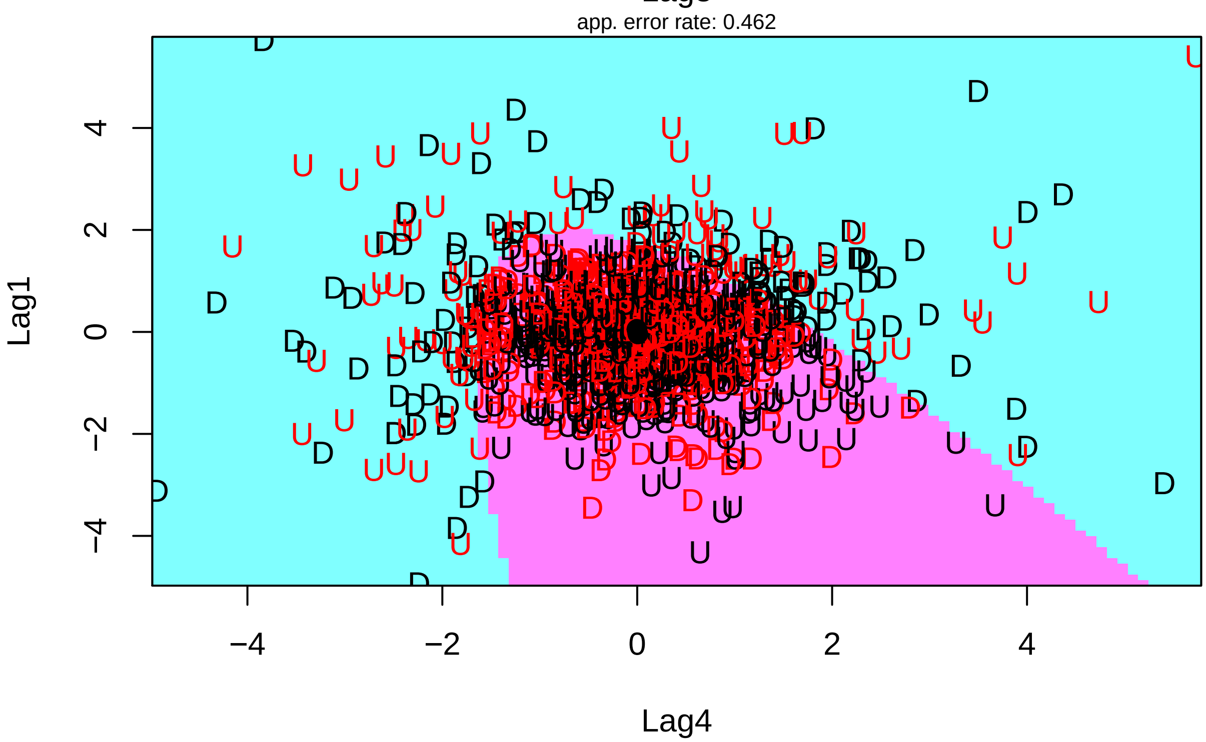
Y aquí los ajustes para QDA en los que se pueden observar ajustes muchos más complejos:

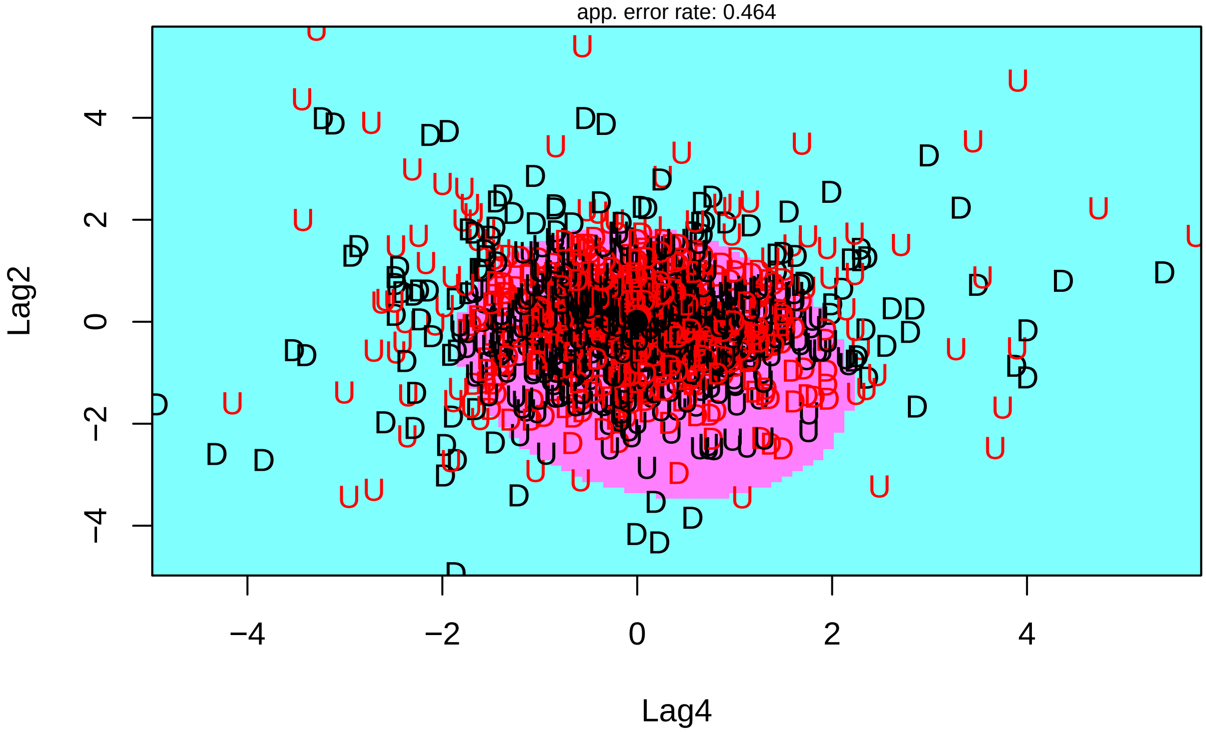
```
partimat(Direction~Lag1+Lag2+Lag3+Lag4+Lag5, data=Smarket, method="qda",nplots.vert=1,nplots.hor=1)
```

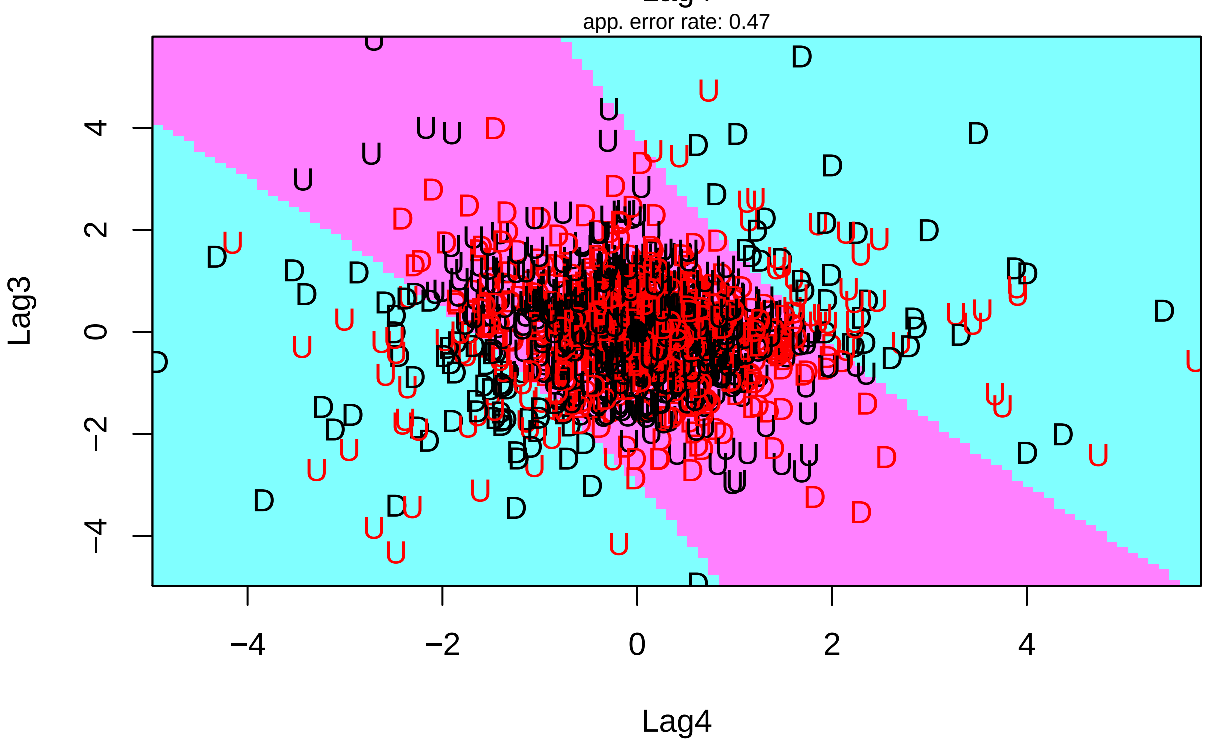

app. error rate: 0.47
Lag1
Lag2

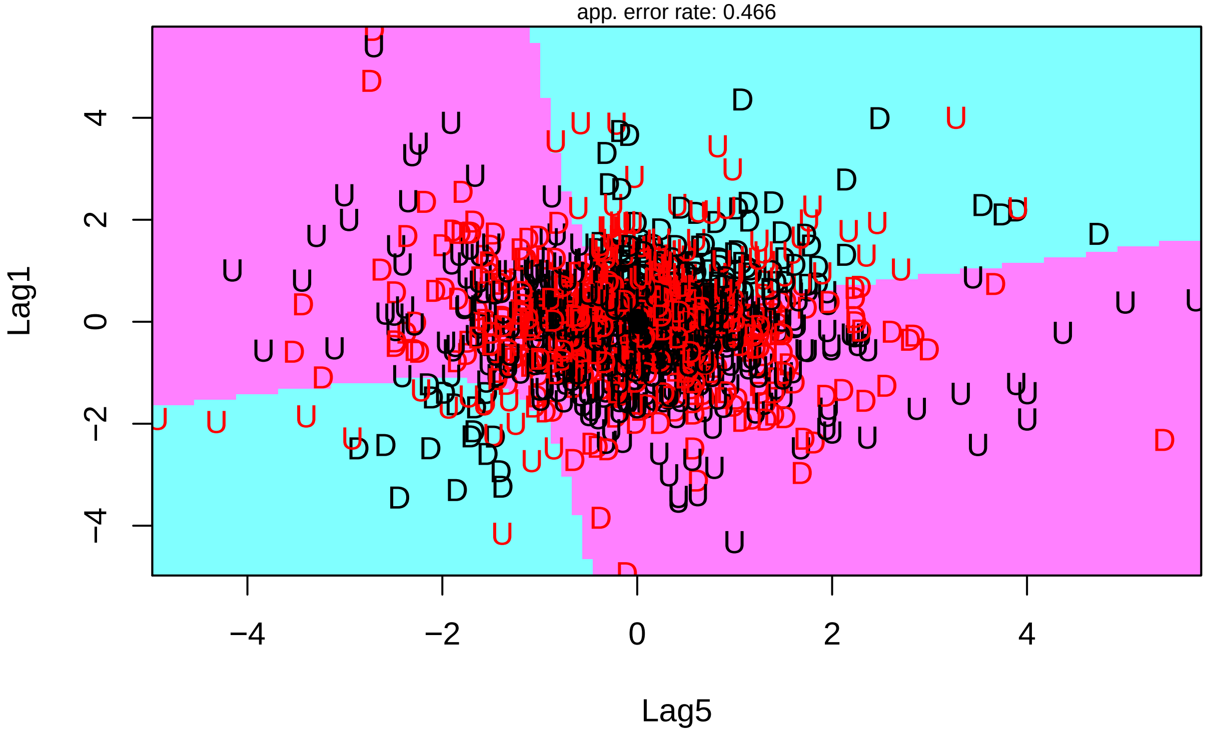

app. error rate: 0.47
Lag1
Lag3


app. error rate: 0.465
Lag2
Lag3
−4
−2
0
2
4

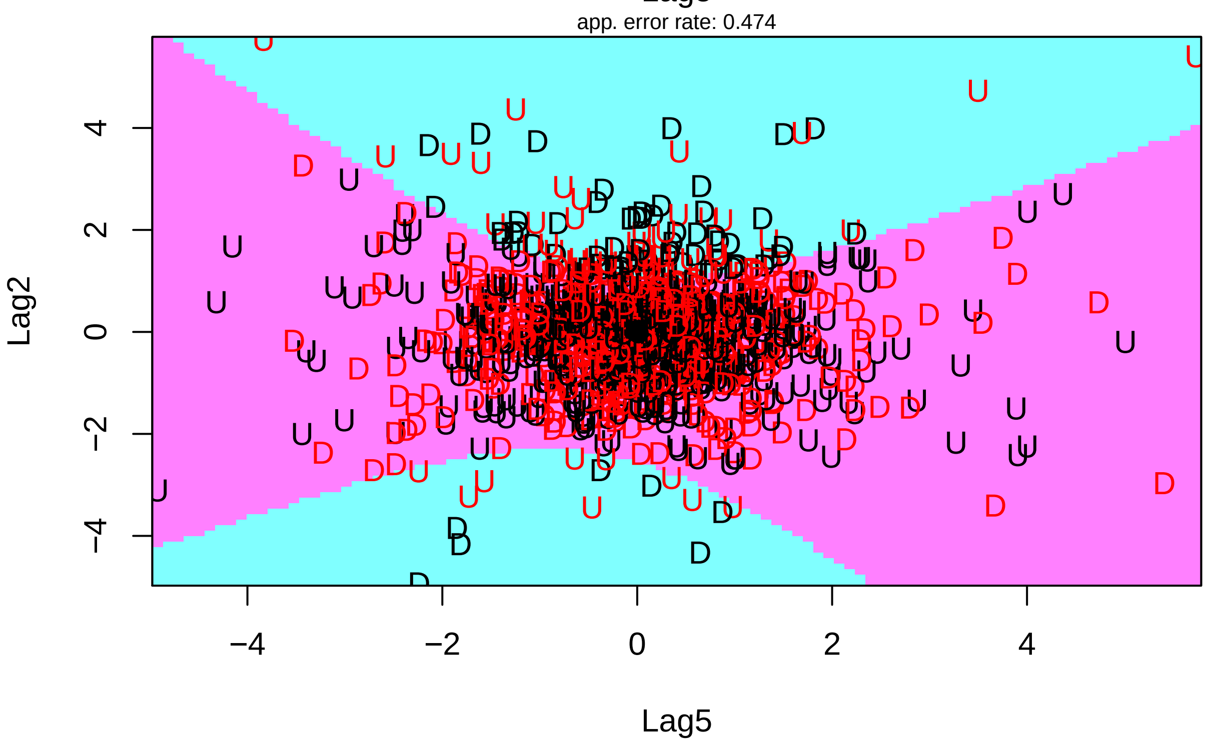

app. error rate: 0.462
Lag1
Lag4
−4
−2
0
2
4


app. error rate: 0.464
Lag2
Lag4
−4
−2
0
2
4

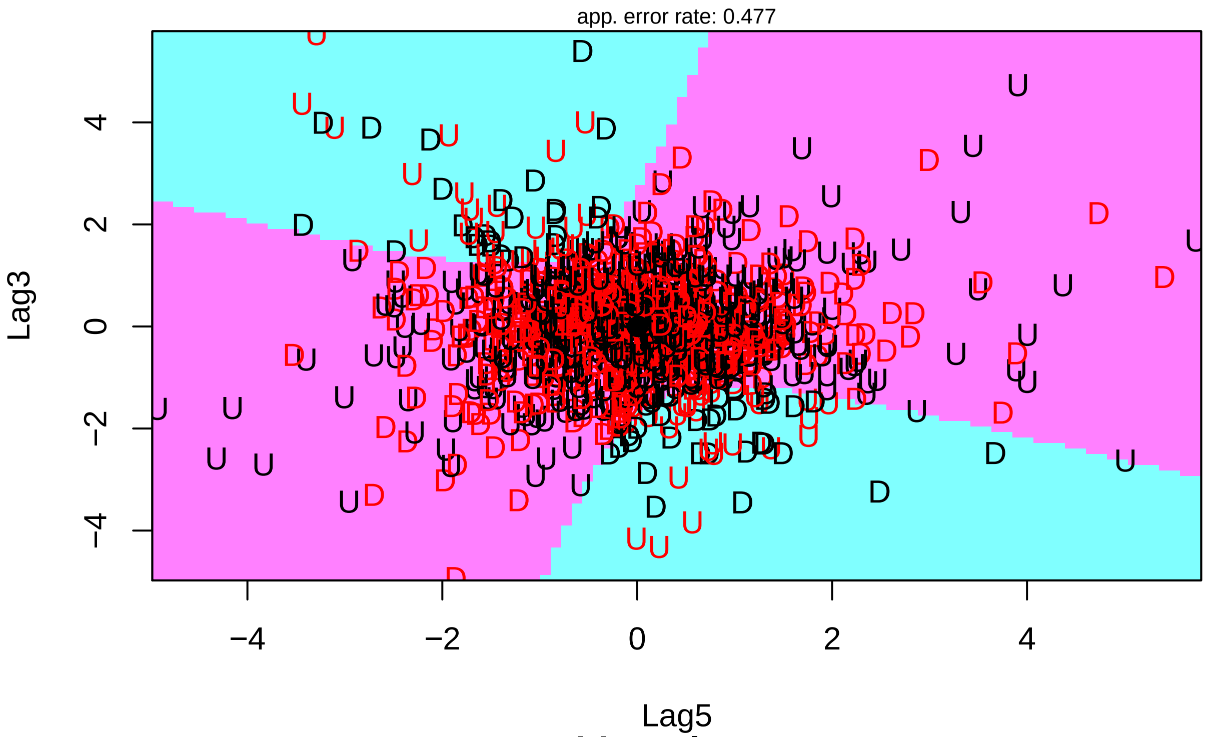

app. error rate: 0.47
Lag3
Lag4
−4
−2
0
2
4

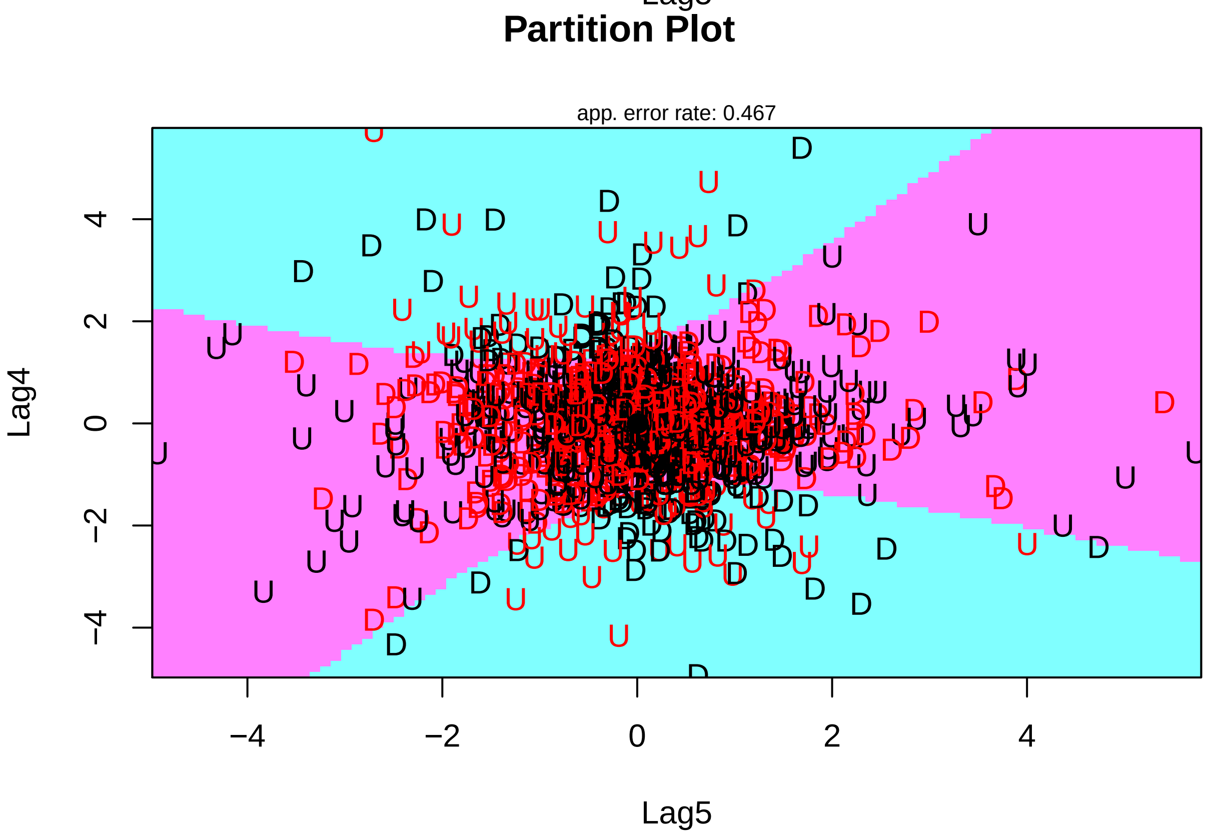


app. error rate: 0.466
Lag1
Lag5
4
2
0
−2
−4



app. error rate: 0.474
Lag2
Lag5
4
2
0
−2
−4





Podemos también comparar ambos métodos la regresión logística vista anteriormente:

```
train <- (Smarket$Year < 2005)
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5, data=Smarket, family=binomial, subset=train)
glm.probs <- predict(glm.fit,newdata=Smarket[!train,], type="response")
```

```r
glm.pred <- ifelse(glm.probs >0.5,"Up","Down")
Direction.2005 <- Smarket$Direction[!train]
table(glm.pred,Direction.2005)
```

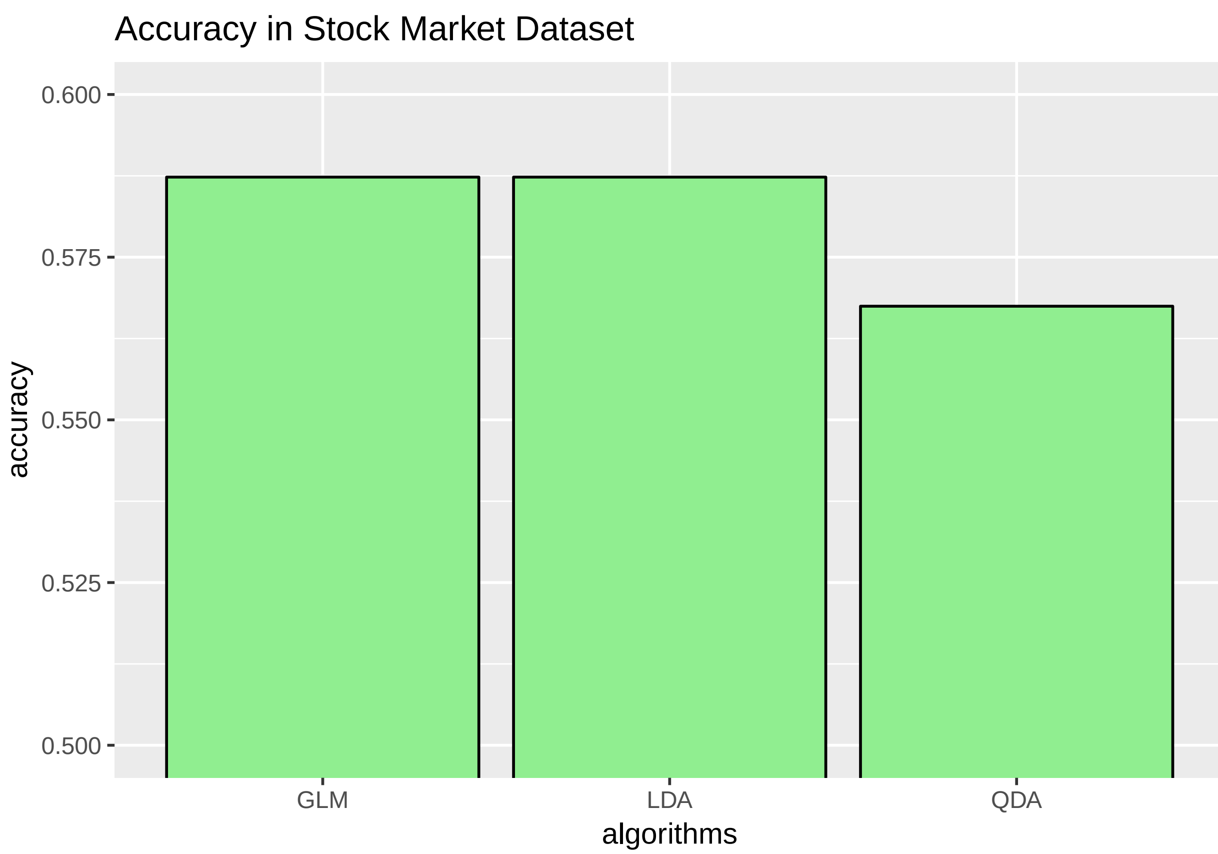
```
##         Direction.2005
## glm.pred Down  Up
##     Down   37  30
##     Up     74 111
```

```r
resultGLM <- mean(glm.pred==Direction.2005)
resultGLM
```

```
## [1] 0.5873016
```

Se obtiene nuevamente un resultado demasiado bajo, un 58.7% de acierto. Viendo los resultados de los 3 algoritmos:

```r
df = data.frame(algorithms=c("LDA","QDA","GLM"),accuracy=c(resultLDA,resultQDA,resultGLM))
ggplot(df,aes(x=algorithms,y=accuracy)) + geom_histogram(stat="identity",color="black",fill="lightgreen
```



Se muestran algo superiores los algoritmos LDA y la regresión logística, aunque igualmente siguen teniendo resultados pobres para este conjunto de datos.

Comparación entre algoritmos

Por último vamos a realizar la comparación de estos algoritmos utilizando para ello diferentes tests.

Primero comparamos lda y qda utilizando Wilcoxon:

```r
resultados <- read.csv("/home/antonio/Descargas/clasif_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]

difs <- (tablatra[,2] - tablatra[,3]) / tablatra[,2]
wilc_2_3 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_2_3) <- c(colnames(tablatra)[2], colnames(tablatra)[3])
head(wilc_2_3)
```

```
##      out_train_lda out_train_qda
## [1,]     0.1000000     0.1142045
## [2,]     0.1000000     0.1619386
## [3,]     0.1428702     0.1000000
## [4,]     0.1000000     0.1820867
## [5,]     0.1148372     0.1000000
## [6,]     0.1062741     0.1000000
```

```r
LDAvsQDAtst <- wilcox.test(wilc_2_3[,1], wilc_2_3[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LDAvsQDAtst$statistic
pvalue <- LDAvsQDAtst$p.value
LDAvsQDAtst <- wilcox.test(wilc_2_3[,2], wilc_2_3[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LDAvsQDAtst$statistic
Rmas
```

```
##   V
## 144
```

```r
Rmenos
```

```
##  V
## 66
```

```r
pvalue
```

```
## [1] 0.1536465
```

Este test nos indica que hay diferencias significativas entre ambos algoritmos con una confianza del 84.7%

Utilizando ahora el test de Friedman:

```r
test_friedman <- friedman.test(as.matrix(tablatra))
test_friedman
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatra)
## Friedman chi-squared = 1.3, df = 2, p-value = 0.522
```

Con un p-value de 0.52, este test nos indica que puede haber ciertas diferencias entre algunos de los algoritmos con una confianza del 48%.

Finalmente aplicamos post-hoc Holm:

```r
tam <- dim(tablatra)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatra), groups, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups
##
##   1    2
## 2 0.65 -
## 3 0.59 0.53
##
## P value adjustment method: holm
```

Este test nos indica que “QDA” no parece tener diferencias muy significativas ni con “LDA” ni con “KNN”. Además, indica que “LDA” y “KNN” son incluso más similares entre ellos.