Práctica 1: Clasificación

Grupo 4

Iria Lago Portela
Mario Picáns Rey
Javier Kniffki
David Bamio Martínez

Ejercicios

En primer lugar vamos a cargar los datos y los paquetes necesarios para la realización de esta práctica:

```
# Librerías
packages <- c("rpart", "rpart.plot", "caret", "randomForest", "pdp", "kernlab")
ipak(packages)
```

```
# Datos
load("data/College4.RData")
head(College4, n = 3)
```

```
##                                Private     Apps   Accept   Enroll Top10perc
## University of Southern Colorado      No 7.244942 7.122060 6.405228        10
## University of San Francisco         Yes 7.743270 7.450661 6.287859        23
## Clarkson University                 Yes 7.684324 7.577122 6.322565        35
##                                Top25perc P.Undergrad Outstate Room.Board Books
## University of Southern Colorado        34     6.514713    7.100      4.380   5.4
## University of San Francisco            48     6.308098   13.226      6.452   7.5
## Clarkson University                    68     3.970292   15.960      5.580   7.0
##                                Personal PhD Terminal S.F.Ratio perc.alumni
## University of Southern Colorado    2.948  63       88      19.4           0
## University of San Francisco        2.450  86       86      13.6           8
## Clarkson University                1.300  95       95      15.8          32
##                                Expend Grad.Rate
## University of Southern Colorado  5.389        36
## University of San Francisco     10.074        62
## Clarkson University             11.659        77
```

```
dim(College4)
```

```
## [1] 500  17
```

Nuestro conjunto de datos contiene 500 universidades, para las cuales se observan 17 variables. Para mejorar la interpretación de los resultados modificaremos la variable tipo de Universidad, **Private**, de modo que 'Yes' sea Privada y 'No' sea Pública.

Además, nótese que la proporción de universidades públicas y privadas no está balanceada:

```
#Proporción privada-pública
table(datos$Tipo)
```

```
##
## Pública Privada
##     143     357
```

1. Obtener un árbol de decisión que permita clasificar las observaciones (universidades) en privadas (`Private = "Yes"`) o públicas (`Private = "No"`).

a) Seleccionar el parámetro de complejidad de forma automática, siguiendo el criterio de un error estándar de Breiman et al. (1984).

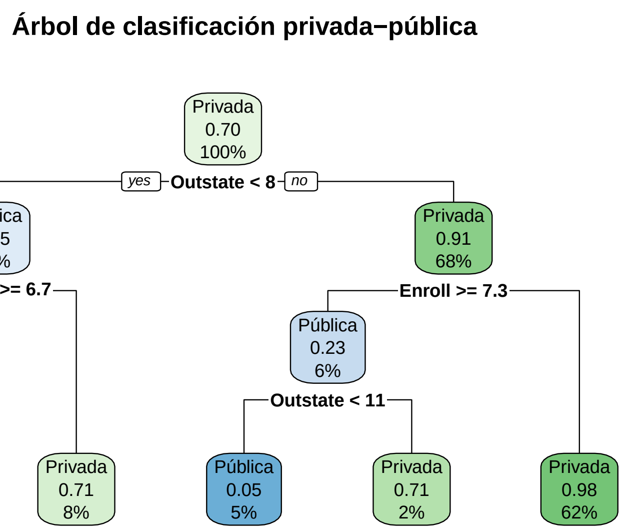
Comenzamos considerando el 80% de las observaciones como muestra de entrenamiento y el 20% restante como muestra de test.

Establecemos la semilla igual al número de grupo multiplicado por 10, utilizando la función `set.seed` de R:

```
#Semilla
set.seed(40)

nobs <- nrow(datos) #Filas
itrain <- sample(nobs, 0.8 * nobs)
train <- datos[itrain, ] # M. Entrenamiento
test <- datos[-itrain, ] # M. Test
```

A continuación, obtendremos un árbol que nos permita clasificar las universidades en privadas y públicas, utilizando la muestra de entrenamiento.

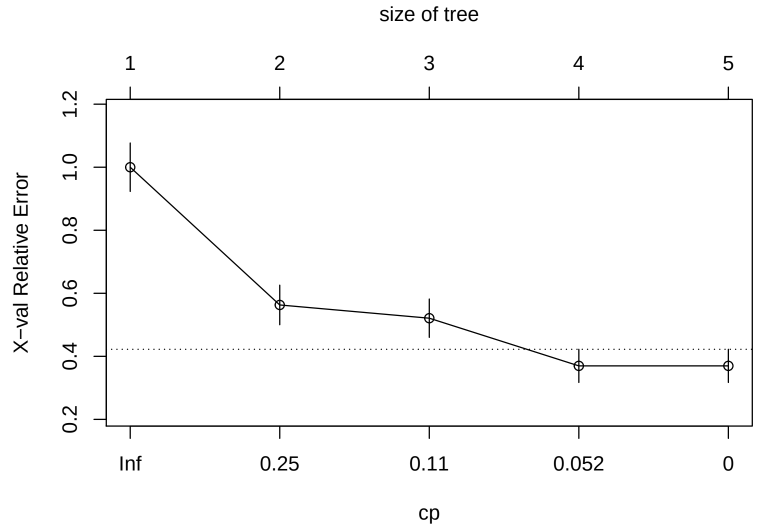
```
tree<-rpart(Tipo~.,data=train)

rpart.plot(tree,main="Árbol de clasificación privada-pública")
```



El resultado es un árbol con 5 nodos terminales, por lo que puede ser interesante podarlo.

Para el proceso de poda seleccionaremos un paramétro de complejidad de forma automática, siguiendo el criterio de un error estándar de Breiman et al. (1984).

```
tree <- rpart(Tipo ~ ., data = train, cp = 0)
plotcp(tree)
```




```
xerror <- tree$cptable[,"xerror"]
imin.xerror <- which.min(xerror)
upper.xerror <- xerror[imin.xerror] + tree$cptable[imin.xerror, "xstd"]
icp <- min(which(xerror <= upper.xerror))
cp <- tree$cptable[icp, "CP"]
cp
```

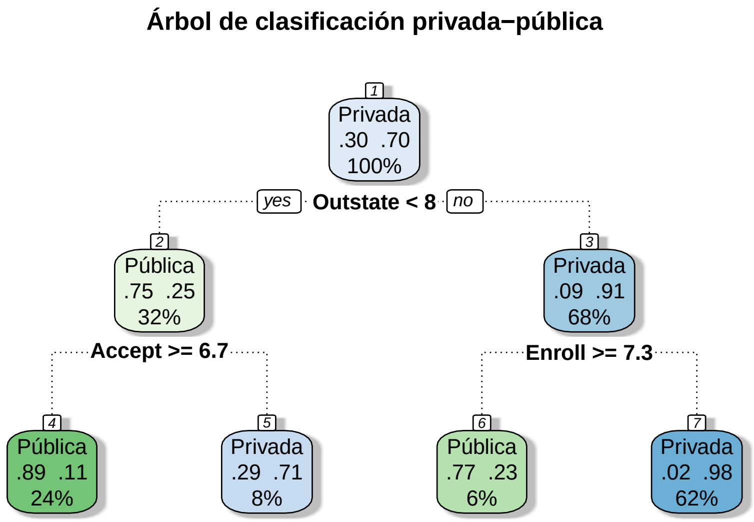
```
## [1] 0.02521008
```

En primer lugar fijamos el parámetro $cp = 0$, es decir, ajustamos el árbol completo. A continuación se calculan los errores de validación cruzada (reescalados) dependiendo del parámetro de complejidad empleado en el ajuste del árbol de decisión. Usando el criterio del error estándar de Breiman nos quedamos con el valor de cp que de lugar al mínimo error, en este caso $cp = 0.02521008$.

b) Representar e interpretar el árbol resultante.

Si podamos el árbol utilizando el valor del parámetro obtenido en el apartado anterior, obtenemos el siguiente árbol:

```
tree<-prune(tree,cp=cp)

rpart.plot(tree,
           extra = 104,          # show fitted class, probs, percentages
           box.palette = "GnBu", # color scheme
           branch.lty = 3,       # dotted branch lines
           shadow.col = "gray",  # shadows under the node boxes
           main="Árbol de clasificación privada-pública",
           nn = TRUE)
```



En este caso obtuvimos un árbol con 4 nodos terminales, que contienen un 24%, 8%, 6% y 62% del total de los datos respectivamente.

El nodo inicial o nodo padre contiene el total de los datos, para los cuales el 70% de los datos son universidades privadas y el 30% son públicas. Dado que la moda o mayoría de universidades son privadas, clasifica como privada.

A continuación el árbol se divide en dos ramas teniendo en cuenta la variable **Outstate**, es decir, el número de estudiantes de otro estado (en miles). Si el número de estudiantes de otro estado es menor que 8000 entonces clasificará como universidad pública, mientras que si es mayor clasificará como privada.

En el nodo 2 se encuentra un 32% de los datos, para los cuales el 75% son universidades públicas y el 25% privadas.

En el nodo 3 se encuentra un 68% de los datos, para los cuales el 9% de los datos son universidades públicas y el 91% son privadas.

A continuación el nodo 2 se divide en otras dos ramas teniendo en cuenta la variable **Accept**, es decir, el número de solicitudes aceptadas en escala logarítmica. Si el número de solicitudes aceptadas es mayor o igual que 6.7 entonces clasificará como universidad pública, mientras que si es menor clasificará como privada.

Por otra parte el nodo 3 se divide en dos teniendo en cuenta la variable **Enroll**, es decir, el número de nuevos estudiantes matriculados en escala logarítmica. Si el número de nuevos estudiantes es mayor o igual que 7.3, el árbol clasificará como universidad pública, mientras que si es menor clasificará como universidad privada.

En el primer nodo terminal se encuentra un 24% de los datos, de los cuales el 89% de las universidades son públicas y el 11% restante son privadas. Dado que hay un mayor número de universidades públicas clasifica en públicas.

En el segundo nodo terminal se encuentra un 8% de los datos, de los cuales el 29% de las universidades son públicas y el 71% restante son privadas, por lo que clasifica en privadas.

En el tercer nodo terminal se encuentra un 6% de los datos, de los cuales el 77% de las universidades son públicas y el 23% restante son privadas, por lo que clasifica en públicas.

En el último nodo terminal se encuentra un 62% de los datos, de los cuales el 2% de las universidades son públicas y el 98% restante son privadas, por lo que clasifica en privadas.

Nótese que tanto el primer como el último nodo terminal poseen colores más oscuros, esto indica que en estos nodos la clasificación es mejor.

c. Evaluar la precisión, de las predicciones y de las estimaciones de la probabilidad, en la muestra de test.

Por último, nos piden evaluar la precisión de las predicciones y de las estimaciones de la probabilidad en la muestra de test. Para ello debemos obtener las observaciones de la muestra de test y compararlas con las predicciones obtenidas con nuestro modelo.

```r
obs <- test$Tipo # Observaciones
pred <- predict(tree, newdata = test, type = "class") #Predicciones

confusionMatrix(pred,obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Pública Privada
##    Pública      17       6
##    Privada       7      70
##
##                Accuracy : 0.87
##                  95% CI : (0.788, 0.9289)
##     No Information Rate : 0.76
##     P-Value [Acc > NIR] : 0.004749
##
##                   Kappa : 0.6385
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7083
##             Specificity : 0.9211
##          Pos Pred Value : 0.7391
##          Neg Pred Value : 0.9091
##              Prevalence : 0.2400
##          Detection Rate : 0.1700
##    Detection Prevalence : 0.2300
##       Balanced Accuracy : 0.8147
##
##        'Positive' Class : Pública
##
```

En primer lugar obtenemos la matriz de confusión, donde enfrentamos observaciones frente a predicciones. En este caso hemos obtenido que el modelo clasifica bien 17 universidades públicas de un total de 24 y 70 universidades privadas de un total de 76. Luego nuestro modelo tiene una precisión de las predicciones de un 87%.

Sin embargo, hay que tener en cuenta que se trata de una muestra desbalanceada, puesto que contiene 143 universidades públicas y 357 universidades privadas. En estos casos conviene fijarse en el valor de Kappa, que posee un valor más bajo, del 63.85%.

Para calcular la precisión de las estimaciones de la probabilidad, debemos obtener las probabilidades estimadas de que cada Universidad sea pública o privada. Para ello usaremos la función `pred` con la opción por defecto `type="prob"`:

```r
pred_prob <- predict(tree, newdata = test) #Estimaciones de la probabilidad
head(pred_prob)
```

```
##                                    Pública   Privada
```

```
## University of San Francisco    0.02016129 0.9798387
## Clarkson University            0.02016129 0.9798387
## Marymount University           0.02016129 0.9798387
## West Virginia Wesleyan College 0.02016129 0.9798387
## Salem-Teikyo University        0.02016129 0.9798387
## Loyola Marymount University    0.02016129 0.9798387
```

Para evaluar las estimaciones de las probabilidades usaremos la curva ROC. Utilizaremos como punto de corte $c = 0.5$ para clasificar en la categoría de interés, puesto que tan sólo tenemos dos categorías, pública y privada.

```
library(pROC)
```

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

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

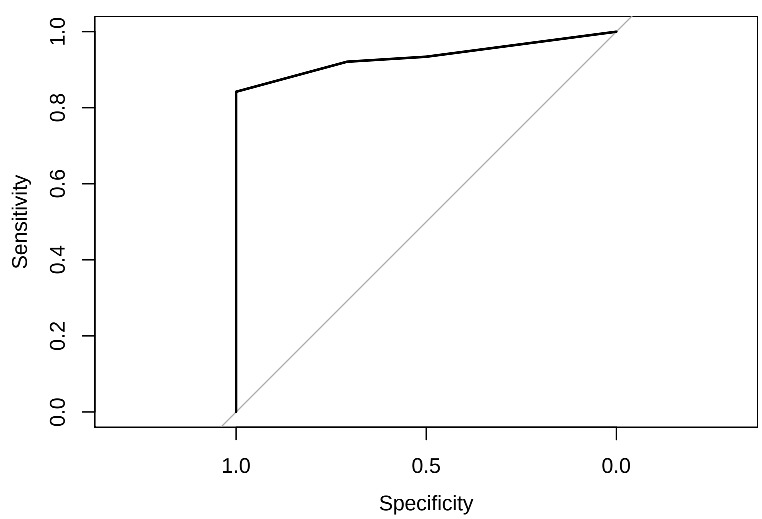
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc_tree<-roc(response=obs,predictor=pred_prob[,1])
```

```
## Setting levels: control = Pública, case = Privada
```

```
## Setting direction: controls > cases
```

```
plot(roc_tree)
```




```
roc_tree
```

```
##
## Call:
## roc.default(response = obs, predictor = pred_prob[, 1])
##
## Data: pred_prob[, 1] in 24 controls (obs Pública) > 76 cases (obs Privada).
## Area under the curve: 0.9339
```

En la curva ROC se representa la sensibilidad frente a la tasa de falsos negativos para distintos valores de corte. En nuestro caso, la curva se aproxima bastante a la esquina superior izquierda, lo que indica un valor alto de sensibilidad y especifidad. Además, el área bajo la curva ROC es 0.9339, muy próximo a 1, por lo que se trata de un buen clasificador.

2. Realizar la clasificación anterior empleando Bosques Aleatorios mediante el método "rf" del paquete caret.

a. Considerar 300 árboles y seleccionar el número de predictores empleados en cada división mtry = c(1, 2, 4, 6) mediante validación cruzada, con 10 grupos y empleando el criterio de un error estándar de Breiman.

Para seleccionar el valor de `mtry` mediante validación cruzada, se crea en primer lugar una rejilla con los valores que se van a considerar. El argumento `trControl` de la función `train` permite seleccionar el método de validación cruzada (`method="cv"`) junto con el criterio de un error estandar (`selectionFunction="oneSE"`).

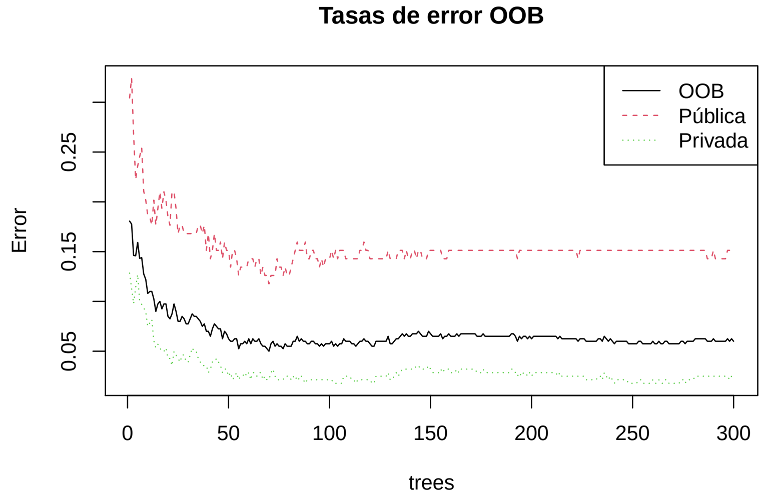
```
set.seed(40)

tuneGrid <- data.frame(mtry = c(1, 2, 4, 6))

rf.caret <-
    train(
        Tipo ~ .,
        data = train,
        method = "rf",
        ntree = 300,
        tuneGrid = tuneGrid,
        trControl = trainControl(
            method = "cv",
            number = 10,
            selectionFunction = "oneSE"
        )
    )

final <- rf.caret$finalModel;final
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = min(param$mtry,      ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 6%
## Confusion matrix:
##         Pública Privada class.error
## Pública     101      18  0.15126050
## Privada       6     275  0.02135231
```

El criterio de validación cruzada ha seleccionado como mejor modelo el que considera `mtry=1`. Los errores de clasificación marginales son del 15.1% para las universidades públicas y del 2.1% para las privadas. Esta diferencia posiblemente se deba al hecho de que la muestra está desbalanceada.

b. Representar la convergencia del error en las muestras OOB en el modelo final.

```
plot(final, main = "Tasas de error OOB")
legend("topright",
       colnames(final$err.rate),
       lty = 1:5,
       col = 1:6)
```



No parece haber problemas de convergencia del error en las muestras OOB para el modelo con 300 árboles. Esta tasa de error se estabiliza en torno al 6%.

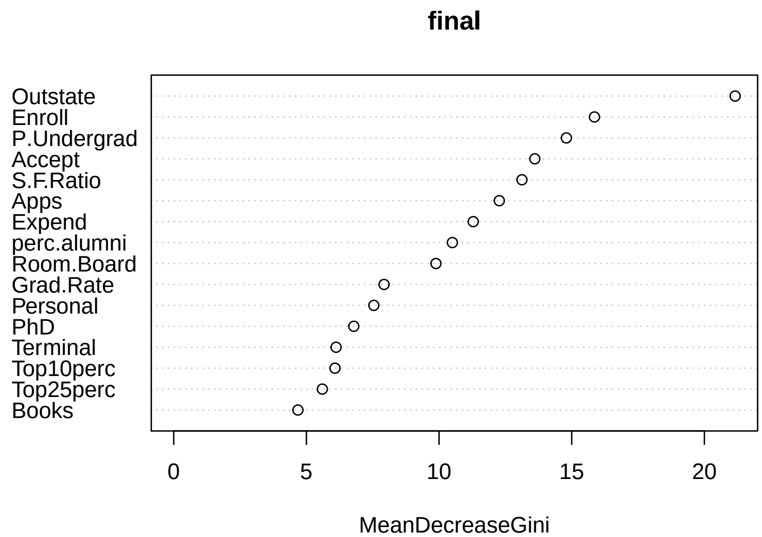
c. Estudiar la importancia de las variables y el efecto de las principales empleando algún método gráfico (para la interpretación del modelo).

Se obtiene la importancia de las variables predictoras mediante la función `importance()` de `randomForest`. También se pueden representar gráficamente con `varImpPlot()`.

```
importance(final)
```

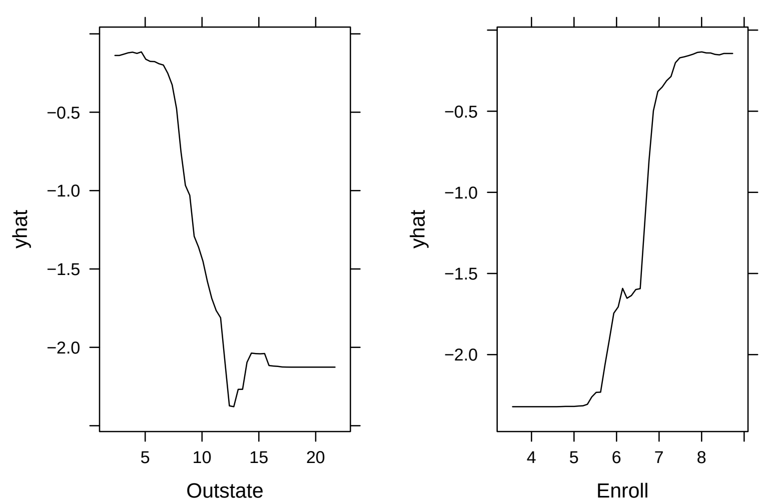
```
##             MeanDecreaseGini
## Apps               12.271024
## Accept             13.608565
## Enroll             15.857022
## Top10perc           6.078058
## Top25perc           5.600199
## P.Undergrad        14.798553
## Outstate           21.159093
## Room.Board          9.883106
## Books               4.681907
## Personal            7.542424
## PhD                 6.786413
## Terminal            6.119014
## S.F.Ratio          13.124654
## perc.alumni        10.502168
## Expend             11.288318
## Grad.Rate           7.927583
```

```
varImpPlot(final)
```



La variable predictora más importante, con cierta diferencia sobre las demás, es **Outstate**. Le siguen **Enroll** y **P.Undergrad**. Para estudiar el efecto de estas variables predictoras en la respuesta, se pueden generar mediante el paquete `pdp` unos gráficos PDP que estiman los efectos individuales de los predictores. Se muestran a continuación los gráficos para las dos variables con mayor importancia, **Outstate** y **Enroll**:

```
pdp1 <- partial(final, "Outstate", train = train)
p1 <- plotPartial(pdp1)

pdp2 <- partial(final, "Enroll", train = train)
p2 <- plotPartial(pdp2)
grid.arrange(p1, p2, ncol = 2)
```



Se observa en este gráfico que la variable **Outstate** se relaciona con la respuesta de manera inversa a **Enroll**. Como ya vimos en los árboles de clasificación del primer ejercicio, valores altos de **Outstate** (es decir, un número elevado de alumnos de otros estados) se relacionan con las universidades privadas, mientras que los valores altos de **Enroll** (es decir, un número elevado de nuevas admisiones) se relacionan más con las universidades públicas.

d. Evaluar la precisión de las predicciones en la muestra de test y comparar los resultados con los obtenidos con el modelo del ejercicio anterior.

Para evaluar la precisión de las predicciones en la muestra de test, guardamos en el objeto `obs` las observaciones de la muestra de test, y las comparamos con las predicciones obtenidas con nuestro modelo:

```
obs <- test$Tipo
pred <- predict(final, newdata = test, type = "class")
table(obs, pred)
```

```
##          pred
## obs       Pública Privada
##   Pública      18       6
##   Privada       4      72
```

```
confusionMatrix(pred, obs)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Pública Privada
##    Pública      18       4
##    Privada       6      72
##
##                Accuracy : 0.9
##                  95% CI : (0.8238, 0.951)
##     No Information Rate : 0.76
##     P-Value [Acc > NIR] : 0.0003075
##
##                   Kappa : 0.7178
##
##  Mcnemar's Test P-Value : 0.7518296
##
##             Sensitivity : 0.7500
##             Specificity : 0.9474
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.9231
##              Prevalence : 0.2400
##          Detection Rate : 0.1800
##    Detection Prevalence : 0.2200
##       Balanced Accuracy : 0.8487
##
##        'Positive' Class : Pública
##
```

La matriz de confusión del nuevo modelo indica que se clasifican correctamente 18 de las 24 universidades públicas y 72 de las 76 universidades privadas. Esto supone una cierta mejora con respecto al modelo del ejercicio 1, como refleja el aumento en la precisión (90% frente al 87% anterior) así como en el valor de Kappa (71.7% frente al 63.8% anterior).

3. Realizar la clasificación anterior empleando SVM mediante la función `ksvm()` del paquete `kernlab`,

a. Ajustar el modelo con las opciones por defecto.

En primer lugar, vamos a ajustar el modelo con los ajustes por defecto que trae la función `ksvm` del paquete `kernlab`, es decir, indicarle la fórmula (`Tipo~.`), y los datos de entrenamiento que se emplean para la clasificación (`train`).

```
set.seed(40)
svm <- ksvm(Tipo ~ ., data = train,prob.model=TRUE)
svm
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0543868376683745
##
## Number of Support Vectors : 114
##
## Objective Function Value : -66.589
## Training error : 0.0425
## Probability model included.
```

```
dim(train)
```

```
## [1] 400  17
```

Observamos que por defecto emplea el parámetro de coste `C = 1`. Calcula $\hat{\sigma} = 0.054$, que es la inversa de la ventana; y emplea 114 vectores soporte, que es aprox. el 50% de los datos.

Podemos realizar predicciones con la función `predict` del paquete base de R. Con el modelo ajustado `svm`, y con el conjunto de datos de test obtenemos las predicciones.

```
pred <- predict(svm, newdata = test)
```

En el apartado c) analizaremos la precisión de las predicciones del modelo ajustado en este apartado.

b. Ajustar el modelo empleando validación cruzada con 10 grupos para seleccionar los valores "óptimos" de los hiperparámetros, considerando las posibles combinaciones de `sigma = c(0.01, 0.05, 0.1)` y `C = c(0.5, 1, 10)` (sin emplear el paquete `caret`; ver Ejercicio 3.1 en *03-bagging_boosting-ejercicios.html*).

En primer lugar, creamos el grid con todas las combinaciones posibles de los hiperparámetros.

```
tune.grid <- expand.grid(
    sigma = c(0.01, 0.05, 0.1),
    C = c(0.5, 1, 10),
    error = NA
)
```

A continuación, como se realiza de forma análoga en *03-bagging_boosting-ejercicios.html*, ajustamos un modelo con cada combinación de hiperparámetros, comprobando en cada iteración si hay una mejora en el modelo.

```
best.err <- Inf
set.seed(40)
for (i in 1:nrow(tune.grid)) {
    fit <-
        ksvm(
            Tipo ~ .,
            prob.model=TRUE,
            data = train[, ],
```

```r
            cross = 10,
            C = tune.grid$C[i],
            kpar = list(tune.grid$sigma[i])
        )
    fit.error <- fit@cross
    tune.grid$error[i] <- fit.error
    if (fit.error < best.err) {
        final.model <- fit
        best.err <- fit.error
        best.tune <- tune.grid[i,]
    }
}
```

Además de obtener el ajuste que se requería, obtenemos los errores para cada combinación (almacenados en `tune.grid`). El ajuste elegido es el que genera un menor error, y para ver con qué combinación de hiperparámetros se ajusta el modelo podemos consultar `best.tune`:

```r
best.tune
```

```
##   sigma   C  error
## 1  0.01 0.5 0.0525
```

Observamos que el mejor modelo es el que emplea $\sigma = 0.01$, y un parámetro de coste $C = 0.5$. Podemos comprobar el ajuste:

```r
final.model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.01
##
## Number of Support Vectors : 145
##
## Objective Function Value : -53.98
## Training error : 0.055
## Cross validation error : 0.0525
## Probability model included.
```

De la misma forma que en el apartado a), podemos realizar predicciones con la función `predict`, el ajuste `final.model`, y el conjunto de datos de test.

```r
pred2 <- predict(final.model, newdata = test)
```

En el próximo apartado mediremos tanto su capacidad predictiva, como la del ajuste calculado en el apartado anterior. Además, compararemos los modelos obtenidos en este ejercicio, con los del ejercicio 2.

c. Evaluar la precisión de las predicciones de ambos modelos en la muestra de test y comparar también los resultados con los obtenidos en el ejercicio anterior.

En primer lugar, evaluaremos la precisión del modelo obtenido las matrices de confusión de las dos predicciones. Lo realizamos con la función `confusionMatrix` del paquete `caret`:

```r
# Predicción con el modelo por defecto (C = 1, sigma = 0.05438):
caret::confusionMatrix(pred, test$Tipo)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Pública Privada
##    Pública      19       3
##    Privada       5      73
##
##                Accuracy : 0.92
##                  95% CI : (0.8484, 0.9648)
##     No Information Rate : 0.76
##     P-Value [Acc > NIR] : 3.001e-05
##
##                   Kappa : 0.7743
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.7917
##             Specificity : 0.9605
##          Pos Pred Value : 0.8636
##          Neg Pred Value : 0.9359
##              Prevalence : 0.2400
##          Detection Rate : 0.1900
##    Detection Prevalence : 0.2200
##       Balanced Accuracy : 0.8761
##
##        'Positive' Class : Pública
##
```

```
# Predicción con el modelo del apartado b) (C=0.05, sigma = 0.01):
caret::confusionMatrix(pred2,test$Tipo)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Pública Privada
##    Pública      19       3
##    Privada       5      73
##
##                Accuracy : 0.92
##                  95% CI : (0.8484, 0.9648)
##     No Information Rate : 0.76
##     P-Value [Acc > NIR] : 3.001e-05
##
##                   Kappa : 0.7743
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.7917
##             Specificity : 0.9605
##          Pos Pred Value : 0.8636
##          Neg Pred Value : 0.9359
##              Prevalence : 0.2400
##          Detection Rate : 0.1900
##    Detection Prevalence : 0.2200
##       Balanced Accuracy : 0.8761
```

```
##
##        'Positive' Class : Pública
##
```

Como podemos observar, se realiza la misma predicción con los dos modelos:

```
##          pred
## obs       Pública Privada
##   Pública      19       5
##   Privada       3      73
```

Y por tanto, obtenemos la misma precisión del modelo (92%),kappa (0.7743), sensibilidad y especificidad en las dos matrices de confusión. Obtener exactamente la misma predicción puede parecer razonable, ya que predecimos entre dos clases (`Privada` o `Pública`). Para asegurarnos de que no hemos hecho nada mal en la construcción de los modelos, comprobamos las probabilidades de las dos predicciones:

```r
#Comprobación de las probabilidades del primer modelo (por defecto):

p.est.1<-predict(svm,newdata = test, type = "probabilities")
head(p.est.1)
```

```
##          Pública   Privada
## [1,] 0.029193617 0.9708064
## [2,] 0.008455068 0.9915449
## [3,] 0.009986539 0.9900135
## [4,] 0.001628170 0.9983718
## [5,] 0.033090830 0.9669092
## [6,] 0.028808907 0.9711911
```

```r
#Comprobación de las probabilidades del segundo modelo (apartado b):

p.est.2<-predict(final.model,newdata = test, type = "probabilities")
head(p.est.2)
```

```
##         Pública   Privada
## [1,] 0.08813235 0.9118676
## [2,] 0.01055591 0.9894441
## [3,] 0.01118644 0.9888136
## [4,] 0.01097525 0.9890247
## [5,] 0.00189130 0.9981087
## [6,] 0.06116127 0.9388387
```

Como se puede observar en los resultados, las probabilidades con las que clasifica no son las mismas, por tanto, no "predicen" igual los dos modelos, aunque luego clasifiquen cada una de las observaciones de la muestra de test en la misma clase (siempre con una probabilidad superior a 0.9).

Respecto a lo obtenido en el ejercicio 2, con estas predicciones mejoramos ligeramente la precisión (0.92 frente a 0.9), la especifidad (0.96 frente a 0.94), y la kappa de Cohen (0.77 frente a 0.71). También se mejora la sensibilidad (0.79 frente a 0.75), pero con valores por debajo del 80%.