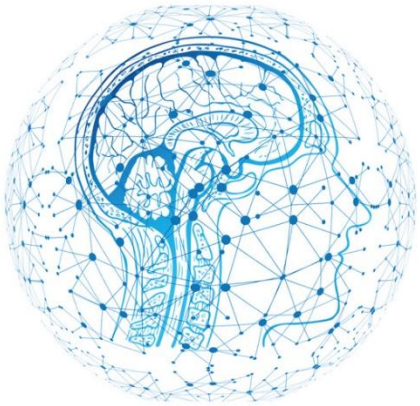
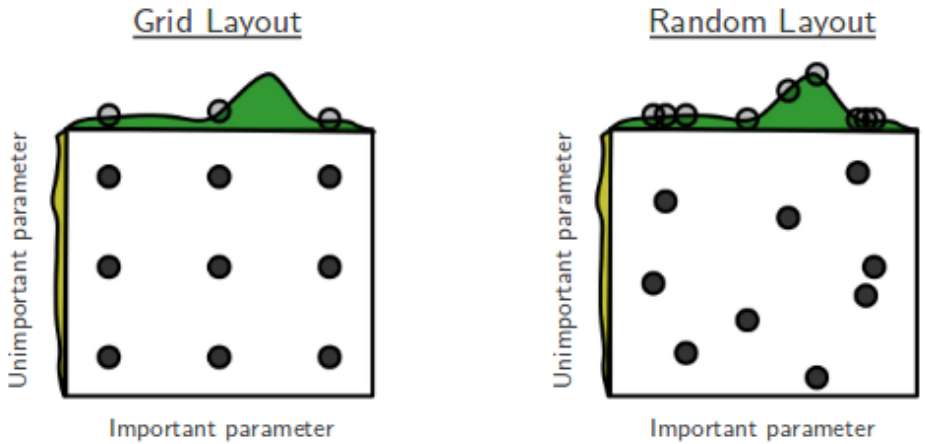
Mejora del Rendimiento y Paquete Caret R



Machine Learning
Dra. María del Carmen Villar Patiño

Hiperparámetros

- Parámetro
  - Característica interna del modelo que se puede estimar de los datos
  - Ejemplo: Coeficientes de las regresiones lineales y logísticas
- Hiperparámetros
  - Característica externa del modelo (se establece antes de usar el modelo) que no se estima de los datos
  - Ejemplo: El valor de k en kNN, profundidad de un árbol
- Técnicas para buscar los mejores hiperparámetros
  - Grid search
  - Random search



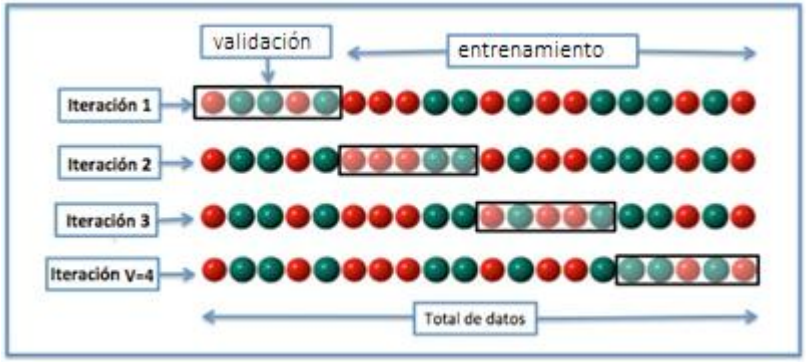

```
> modelLookup('knn')
  model parameter      label forReg forClass probModel
1   knn         k #Neighbors   TRUE     TRUE      TRUE
> modelLookup('rpart')
  model parameter                label forReg forClass probModel
1 rpart        cp Complexity Parameter   TRUE     TRUE      TRUE
```

Técnicas de remuestreo (resampling)

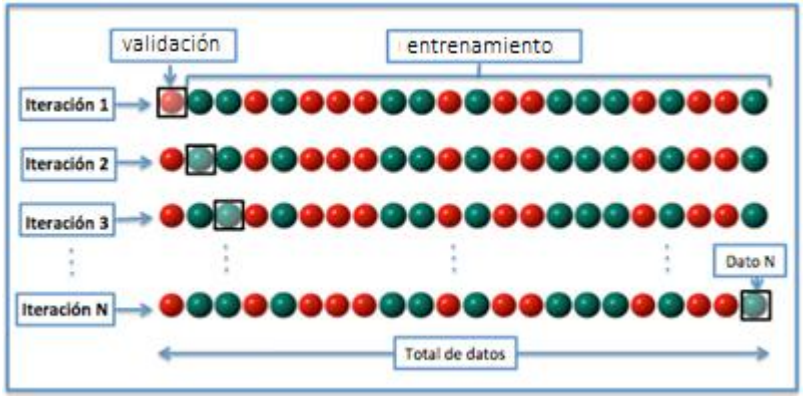
- Idea
  - Ajustar y evaluar el modelo múltiples veces
  - Usar distintos subconjuntos creados a partir de los datos
  - Obtener en cada repetición una estimación del error
  - El promedio de todas las estimaciones tiende al valor real del error de prueba
- Se aplica sobre los elementos de la muestra que no pertenecen al CP
- Objetivo
  - Disminuir sobreajuste
  - Afinar hiperparámetros
- Ventajas
  - Sencillo
  - Ayuda cuando se trabaja con muestras pequeñas
- Técnicas importantes
  - Bootstraping
  - Validación cruzada
- Desventajas
  - Costo computacional

Tipos validaciones cruzadas

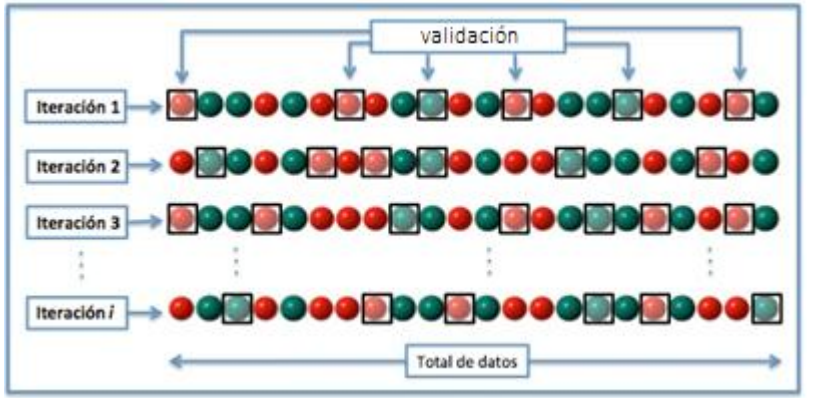
- Validación cruzada de (K)*V*-iteraciones
  - *V-fold cross-validation*
  - Divide los datos en *V* grupos excluyentes de igual tamaño, se saca uno de los grupos para CV y crea el clasificador con la combinación de los *V*-1 grupos que forman el CE
  - Cada elemento de la muestra fue usado una vez como parte del CV



- Validación cruzada dejando uno fuera
  - **LOOCV**, *leave one out cross-validation*
  - El caso extremo cuando *V*=*N* donde *N* es el tamaño de la muestra y cada CV contiene sólo un elemento



- Validación cruzada aleatoria o de Monte Carlo
  - *Repeated random sub-sampling validation*
  - Se establece el número de elementos en el CE y el número de iteraciones
  - Algunas muestras quedan sin evaluar y otras se evalúan más de una vez



Bootstrapping

- Muestra bootstrap
  - Mismo tamaño que la muestra original (CE)
  - Se obtiene por muestreo aleatorio con reemplazo
    - Después de que una observación es extraída, se "regresa" para las siguientes extracciones

- Algunos elementos aparecen múltiples veces en la muestra boot y otros ninguna
  - Las observaciones no seleccionadas reciben el nombre de out-of-bag (OOB)

- En cada iteración
  - Se genera una nueva muestra Bootstrap
  - Se ajusta el modelo con ella
  - Se evalúa/valida con las observaciones OOB

Ejercicio

```
> datos
   num letra
1   24      A
2   23      A
3   30      A
4   22      A
5   21      A
6   11      A
7   17      A
8   20      B
9   36      B
10  22      B
11  27      B
12  50      B
13  39      B
14  21      B
15  65      C
16  59      C
17  54      C
18  47      C
19  69      C
20  63      C
21  52      C
```

```
> library(caTools)
> div = sample.split(datos$letra, SplitRatio=0.75)
> CE = subset(datos, div == TRUE)
> CP = subset(datos, div == FALSE)
> CP
   num letra
1   24      A
3   30      A
9   36      B
10  22      B
16  59      C
18  47      C
> CE
   num letra
2   23      A
4   22      A
5   21      A
6   11      A
7   17      A
8   20      B
11  27      B
12  50      B
13  39      B
14  21      B
15  65      C
17  54      C
19  69      C
20  63      C
21  52      C
```

- Obtener los CE y CV generados por
  - Validación cruzada de 3-folds
  - Validación dejando 1 fuera
  - Validación de Montecarlo 3 iteraciones, 4 elementos CV
  - Bootstrap de 3 iteraciones
- Responder, en cuáles métodos es válido pedir:
  - CE = 60%, CP = 30% y CV = 10%

Biblioteca caret de R

- Classification and Regression Training
  - Interfaz que unifica cientos de funciones de distintos paquetes
  - Contempla las etapas
    - Preprocesado, entrenamiento, optimización y validación de modelos predictivos
  - Incluye funciones de visualización de datos
- Paquete base
  - library(**caret**)
- Paquetes requeridos por algunas funciones
  - library(lattice)
  - library(e1071)

- Visualizar características

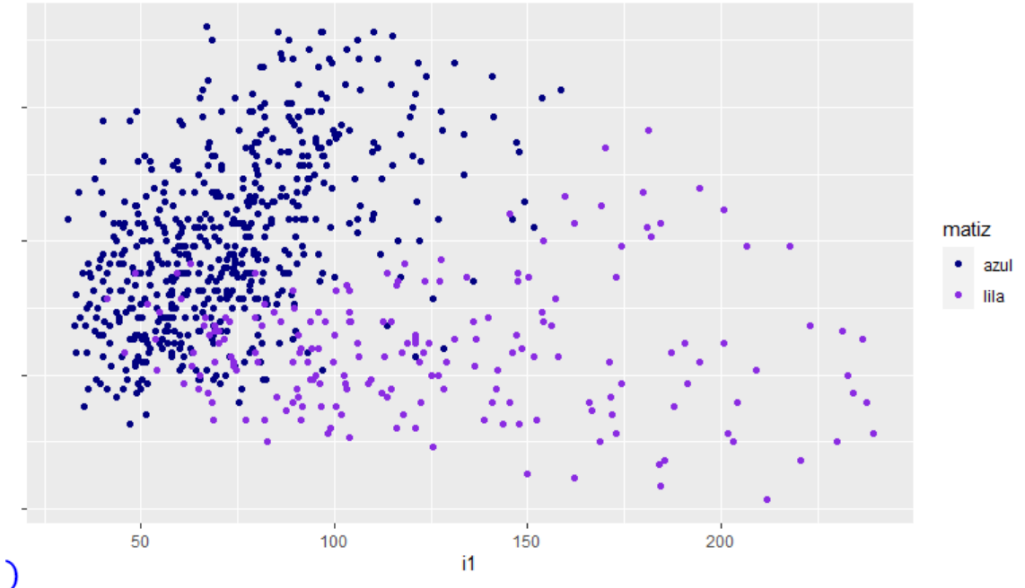
  featurePlot(x = *caracteristicas*, y = *clase*, plot = *gráfico*)

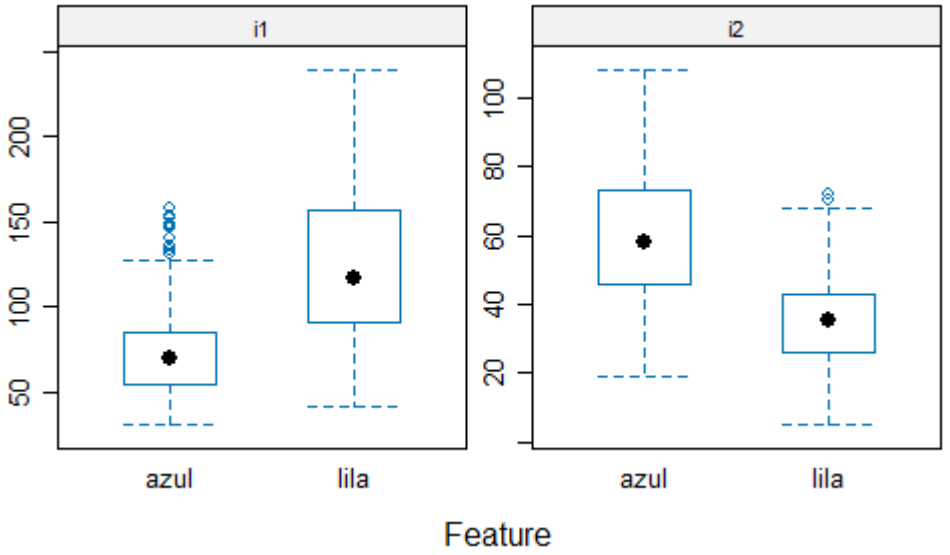
- Dividir en CE y CP

  createDataPartition(y=*clase*, p=*%enCE*, list=FALSE)

Visualización de datos y CE-CV-CP en caret

```
> #Lectura y resumen de datos
> datos = read.table("colorOtha.dat",header=TRUE, stringsAsFactors=TRUE)
> summary(datos)
       i1               i2           matiz
 Min.   : 31.00   Min.   :  2.00   azul:570
 1st Qu.: 59.23   1st Qu.: 38.00   lila:198
 Median : 76.70   Median : 52.00
 Mean   : 85.73   Mean   : 53.81
 3rd Qu.: 98.78   3rd Qu.: 68.00
 Max.   :239.30   Max.   :108.00
> #gráfica distribución de datos
> library(ggplot2)
> ggplot(datos, aes(x=i1,y=i2,color=matiz)) +  geom_point() +
+   scale_color_manual(values = c("azul"="navyblue","lila"="blueviolet"))
> #Bibliotecas
> library(lattice)
> library(e1071)
> library(caret)
> #Dividir datos en conjuntos de entrenamiento y prueba
> set.seed(55)
> div <- createDataPartition(y=datos$matiz, p=0.70, list = FALSE)
> CECV <- datos[div,]
> CP <- datos[-div,]
> featurePlot(x = CECV[,1:2],
+             y = CECV$matiz,
+             plot = "box",
+             strip=strip.custom(par.strip.text=list(cex=.7)),
+             scales = list(x = list(relation="free"),
+                           y = list(relation="free")))
```





Construcción de modelos kNN en caret

- Función para crear el modelo: **train()**
- Algunos argumentos
  - Formula
  - method: algoritmo a emplear
    - Con sus argumentos propios
  - metric: medidas de la capacidad predictiva del modelo
  - preProcess: escalamiento
- Problemas de clasificación binaria y multiclase

```
> # kNN con valores default
> knn.m1 <- train(matiz ~ .,
+                 data = CECV,
+                 method = 'knn',
+                 preProcess = c("center","scale"))
> knn.m1
k-Nearest Neighbors

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 538, 538, 538, 538, 538, 538, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.8896178  0.7109320
  7  0.9003920  0.7386289
  9  0.9044943  0.7475614

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Hiperparámetros y remuestreo en caret

- Hiperparámetros
  - Se determinan de forma automática (default)
  - Se especifican valores con la función **tuneGrid()**
- Resampling
  - El método y sus características se colocan en la función **trainControl()**
    - 'boot': Bootstrap sampling
    - 'boot632': Bootstrap sampling with 63.2% bias correction applied
    - 'optimism_boot': The optimism bootstrap estimator
    - 'boot_all': All boot methods
    - 'cv': k-Fold cross validation
    - 'repeatedcv': Repeated k-Fold cross validation
    - 'oob': Out of Bag cross validation
    - 'LOOCV': Leave one out cross validation
    - 'LGOCV': Leave group out cross validation (Monte Carlo)
  - Es posible
    - Paralelizar el proceso para que sea más rápido
    - Establecer semillas para asegurar que cada remuestreo o partición pueda crearse de nuevo con exactamente las mismas observaciones

Hiperparámetros y Remuestreo con KNN

```
> # kNN  valor fijo k
> parametros <- expand.grid(k = 1)
> knn.m2 <- train(matiz ~ .,
+                 data = CECV,
+                 method = 'knn',
+                 preProcess = c("center","scale"),
+                 tuneGrid = parametros)
> knn.m2
k-Nearest Neighbors

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 538, 538, 538, 538, 538, 538,
Resampling results:

  Accuracy   Kappa
  0.8795051  0.6853199

Tuning parameter 'k' was held constant at a value of 1
```

```
> # Define el grid de valores de k a probar
> parametros <- expand.grid(k = seq(1,15,2))
> # Define el método de remuestreo a utilizar
> ajustes <- trainControl(method='cv',  # validación cruzada
+                         number = 10)  # diez submuestras v=10
> # Se construye el modelo
> knn.m3 <- train(matiz ~ .,
+                 data = CECV,
+                 method = 'knn',
+                 preProcess = c("center","scale"),
+                 tuneGrid = parametros,
+                 trControl = ajustes)
> knn.m3
k-Nearest Neighbors

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 484, 484, 484, 484, 484, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8811254  0.6894253
   3  0.8977920  0.7291344
   5  0.9145299  0.7751760
   7  0.9051282  0.7482829
   9  0.9143162  0.7711773
  11  0.9107550  0.7621888
  13  0.9069088  0.7508968
  15  0.9106125  0.7571406

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

```
> # Define el grid de parámetros a probar
> parametros <- expand.grid(k = c(6,8,11,15))
> set.seed(2020)
> ajustes <- trainControl(
+             method = "repeatedcv", # 10-fold CV
+             number = 10, # v=10
+             repeats = 5  # repetido 5 veces
+             )
> set.seed(2020)
> knn.m4 <- train(matiz ~ .,
+                 data = CECV,
+                 method = 'knn',
+                 preProcess = c("center","scale"),
+                 tuneGrid = parametros,
+                 trControl = ajustes)
> knn.m4
k-Nearest Neighbors

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 485, 484, 484, 484, 484, 484, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   6  0.9140429  0.7721244
   8  0.9118629  0.7633805
  11  0.9178173  0.7812189
  15  0.9189066  0.7802251

Accuracy was used to select the optimal model using the largest
The final value used for the model was k = 15.
```

```
> # Monte Carlo CV y AUC
> set.seed(2020)
> ajustes <- trainControl(method = "LGOCV",
+                         number = 10,
+                         classProbs = TRUE,
+                         summaryFunction = twoClassSummary #ROC
+                         )
> set.seed(2020)
> knn.m5 <- train(matiz ~ .,
+                 data = CECV,
+                 method = 'knn',
+                 preProcess = c("center","scale"),
+                 trControl = ajustes)
Warning message:
In train.default(x, y, weights = w, ...) :
  The metric "Accuracy" was not in the result set. ROC will be used instead.
> knn.m5
k-Nearest Neighbors

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Repeated Train/Test Splits Estimated (10 reps, 75%)
Summary of sample sizes: 405, 405, 405, 405, 405, 405, ...
Resampling results across tuning parameters:

  k  ROC        Sens       Spec
  5  0.9433155  0.9484848  0.8323529
  7  0.9522133  0.9515152  0.8205882
  9  0.9589721  0.9555556  0.8117647

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Predicción

- Función

```
predict(modelo,
        newdata=CP,
        type="raw" o "prob")
```

```
> # Predicción
> y_pred <- predict(knn.m1,CP,type="prob")
> head(y_pred)
      azul      lila
1 0.5555556 0.4444444
2 0.8888889 0.1111111
3 1.0000000 0.0000000
4 0.6666667 0.3333333
5 0.8888889 0.1111111
6 1.0000000 0.0000000
> y_pred.m1 <- predict(knn.m1, CP)
> head(y_pred.m1)
[1] azul azul azul azul azul azul
Levels: azul lila
> confusionMatrix(y_pred.m1, CP$matiz)$table
          Reference
Prediction azul lila
     azul  165   14
     lila    6   45
```

```
> y_pred.m2 <- predict(knn.m2, CP)
> confusionMatrix(y_pred.m2, CP$matiz)$table
          Reference
Prediction azul lila
     azul  157   13
     lila   14   46
> y_pred.m3 <- predict(knn.m3, CP)
> confusionMatrix(y_pred.m3, CP$matiz)$table
          Reference
Prediction azul lila
     azul  163   15
     lila    8   44
> y_pred.m4 <- predict(knn.m4, CP)
> confusionMatrix(y_pred.m4, CP$matiz)$table
          Reference
Prediction azul lila
     azul  163   15
     lila    8   44
> y_pred.m5 <- predict(knn.m5, CP)
> confusionMatrix(y_pred.m5, CP$matiz)$table
          Reference
Prediction azul lila
     azul  165   14
     lila    6   45
```

Árboles en Caret

```
> #Lectura de datos
> datos = read.table("colorOtha.dat",header=TRUE)
> library(caret)
> #Dividir datos en conjuntos de entrenamiento y prueba
> set.seed(55)  #Fijar semilla para replicar resultado
> div <- createDataPartition(y=datos$matiz, p=0.70, list = FALSE)
> CECV <- datos[div,]
> CP <- datos[-div,]
> # Arbol valores default
> arbol.m1 <- train(matiz ~ .,
+                   data = CECV,
+                   method = 'rpart')
> arbol.m1
CART

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 538, 538, 538, 538, 538, 538, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.09352518  0.8618228  0.6065002
  0.11510791  0.8530113  0.5767600
  0.38848921  0.7781546  0.2552699

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.09352518.
```

```
> # Arbol con cp como hiperparámetro
> modelLookup('rpart')
  model parameter                label forReg forClass probMode
1 rpart        cp Complexity Parameter   TRUE     TRUE      TRU
> parametros <- expand.grid(cp = seq(.01,.1,0.02))
> set.seed(19)
> ajustes <- trainControl(method='cv',
+                         number = 5)  # v=5
> set.seed(19)
> arbol.m2 <- train(matiz ~ .,
+                   data = CECV,
+                   method = 'rpart',
+                   tuneGrid = parametros,
+                   trControl = ajustes)
> arbol.m2
CART

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 431, 430, 430, 431, 430
Resampling results across tuning parameters:

  cp    Accuracy   Kappa
  0.01  0.8903600  0.7116579
  0.03  0.8866563  0.7045457
  0.05  0.8811007  0.6776523
  0.07  0.8829699  0.6772311
  0.09  0.8792662  0.6481693

Accuracy was used to select the optimal model using the largest
The final value used for the model was cp = 0.01.
```

Hiperparámetro: profundidad árbol

```
> # Arbol con profundidad como hiperparámetro
> modelLookup('rpart2')
   model parameter          label forReg forClass probModel
1 rpart2  maxdepth Max Tree Depth   TRUE     TRUE      TRUE
> parametros <- expand.grid(maxdepth = c(3:6))
> set.seed(19)
> ajustes <- trainControl(method='cv',
+                         number = 5)
> set.seed(19)
> arbol.m3 <- train(matiz ~ .,
+                   data = CECV,
+                   method = 'rpart2',
+                   tuneGrid = parametros,
+                   trControl = ajustes)
> arbol.m3
CART

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 431, 430, 430, 431, 430
Resampling results across tuning parameters:

  maxdepth  Accuracy   Kappa
  3         0.8866563  0.7045457
  4         0.8866563  0.7045457
  5         0.8922118  0.7183172
  6         0.8903600  0.7116579

Accuracy was used to select the optimal model using the largest
The final value used for the model was maxdepth = 5.
```

```
> y_pred.m3 <- predict(arbol.m3, CP)
> confusionMatrix(y_pred.m3, CP$matiz)
Confusion Matrix and Statistics

          Reference
Prediction azul lila
      azul  164   17
      lila    7   42

               Accuracy : 0.8957
                 95% CI : (0.8487, 0.932)
    No Information Rate : 0.7435
    P-Value [Acc > NIR] : 6.641e-09

                  Kappa : 0.7104

 Mcnemar's Test P-Value : 0.06619

            Sensitivity : 0.9591
            Specificity : 0.7119
         Pos Pred Value : 0.9061
         Neg Pred Value : 0.8571
             Prevalence : 0.7435
         Detection Rate : 0.7130
   Detection Prevalence : 0.7870
      Balanced Accuracy : 0.8355

       'Positive' Class : azul
```

Regresión logística en caret

- Método
  - glm con family binomial

```
> # Modelo
> modelLookup('glm')
  model parameter     label forReg forClass probModel
1   glm parameter parameter   TRUE     TRUE      TRUE
> # Dividir datos en conjuntos de entrenamiento y prueba
> set.seed(55)
> div <- createDataPartition( y=datos$matiz, p=0.70, list=FALSE)
> CECV <- datos[div,]
> CP <- datos[-div,]
> # Con valores de default
> rl.m1 <- train(matiz ~ .,
+                data = CECV,
+                method = 'glm',
+                family = 'binomial',
+                preProcess = c("center","scale"),
+                trControl = trainControl(method = "cv",
+                                         number = 10))
> rl.m1
Generalized Linear Model

538 samples
  2 predictor
  2 classes: 'azul', 'lila'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 485, 484, 485, 484, 484, ...
Resampling results:

  Accuracy   Kappa
  0.9144654  0.7709442
```

```
> confusionMatrix(rl.m1)
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction azul lila
      azul 71.0  5.4
      lila  3.2 20.4

 Accuracy (average) : 0.9145

> varImp(rl.m1)
glm variable importance

   Overall
i1     100
i2       0
> y_pred <- predict(rl.m1, CP)
> confusionMatrix(y_pred, CP$matiz)
Confusion Matrix and Statistics

          Reference
Prediction azul lila
      azul  161   15
      lila   10   44

               Accuracy : 0.8913
                 95% CI : (0.8437, 0.9284)
    No Information Rate : 0.7435
    P-Value [Acc > NIR] : 1.931e-08

                  Kappa : 0.7069

 Mcnemar's Test P-Value : 0.4237

            Sensitivity : 0.9415
            Specificity : 0.7458
         Pos Pred Value : 0.9148
         Neg Pred Value : 0.8148
             Prevalence : 0.7435
         Detection Rate : 0.7000
   Detection Prevalence : 0.7652
      Balanced Accuracy : 0.8436

       'Positive' Class : azul
```