



Aprendizaje supervisado con R

Francisco Charte



En esta sesión nos ocuparemos de:

- Fundamentos sobre aprendizaje supervisado
 - Paquetes para aprendizaje supervisado en R
 - Particionamiento de datos
 - Evaluación del rendimiento
- Ejercicios prácticos
 - Máquinas de vectores soporte
 - Boosting, Bagging, Random forest
 - Deep learning



Fundamentos sobre aprendizaje supervisado en R

Paquetes para aprendizaje supervisado en R

Paquetes para aprendizaje supervisado en R

- CRAN Task View: Machine Learning
cran.r-project.org/web/views/MachineLearning.html
 - Paquetes por tipo de modelo
 - 80+ paquetes en la actualidad
 - Implementaciones heterogéneas
- Paquetes que nos interesan
 - e1071: Máquinas de vectores soporte
 - neuralnet: Redes neuronales
 - gbm: Boosting
 - randomForest: Random forest
 - h2o, SAENET: Deep learning



Paquetes para aprendizaje supervisado en R

- El paquete `caret`
 - Particionamiento, preprocesamiento y validación
 - Interfaz común para entrenamiento y predicción
 - Procedimiento homogéneo de evaluación
 - Uso desde línea de comandos (reproducibilidad)
- El paquete `rattle`
 - Interfaz gráfica de usuario
 - Carga y exploración de datos
 - Funciones de preprocesamiento y aprendizaje
 - Facilidad de uso sin línea de comandos



Fundamentos sobre aprendizaje supervisado en R

Particionamiento de datos

Particionamiento de datos - básico

- Primeras n muestras para entrenamiento, resto para test

```
nTraining <- as.integer(nrow(iris) * .75)

particion <- list(
  training = iris[1:nTraining, ],
  test     = iris[(nTraining+1):nrow(iris), ])

particion$training # Muestras de entrenamiento
particion$test     # Muestras de test
```

- Muestreo aleatorio

```
set.seed(4242)
indices <- sample(1:nrow(iris), nTraining)

particion <- list(
  training = iris[indices, ],
  test     = iris[-indices, ])
```

Particionamiento de datos - caret

► Muestreo estratificado

```
library(caret)
indices <- createDataPartition(
  iris$Species, p = .75, list = FALSE)

particion <- list(
  training = iris[indices,],
  test      = iris[-indices, ])
```

► Creación de múltiples *folds* para validación cruzada

```
folds <- createFolds(iris$Species, k = 10)

particion <- lapply(folds, function(indices)
  list(training = iris[-indices, ],
        test = iris[indices, ]))

# Muestras de entrenamiento del cuarto fold
particion$Folds04$training
```


Entrenamiento del modelo - caret

- La función **train()** entrena el modelo haciendo las evaluaciones que se indiquen sobre datos de training:

```
library(caret)
```

```
# 10 validación cruzada
```

```
train10CV <- trainControl(method = "cv",  
                           number = 10)
```

```
# 5 validación cruzada con 2 repeticiones
```

```
train2x5 <- trainControl(method = "repeatedcv",  
                           number = 5, repeats = 2)
```

```
# Uso de la configuración para entrenar
```

```
modelo <- train(formula, data = training,  
               method = "classifier",  
               trControl = train10CV)
```

```
names(getModelInfo()) # Lista métodos disponibles
```

Evaluación del modelo - caret

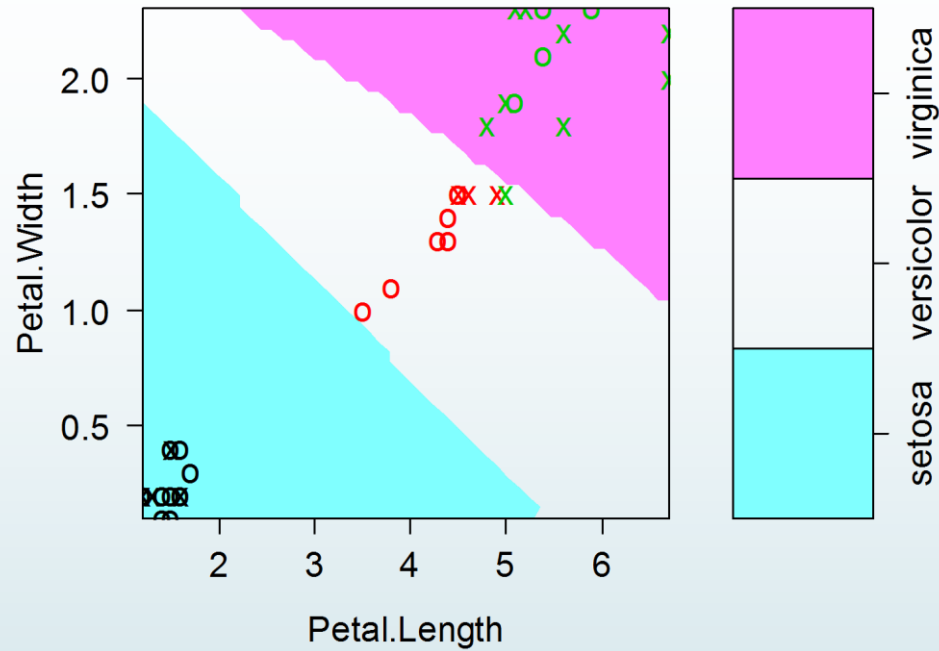
- La función **predict()** devuelve predicciones para datos de test:

```
modelo <- train(formula, data = training,  
               method = "classifier",  
               trControl = train10CV)
```

```
pred <- predict(modelo, testData)
```

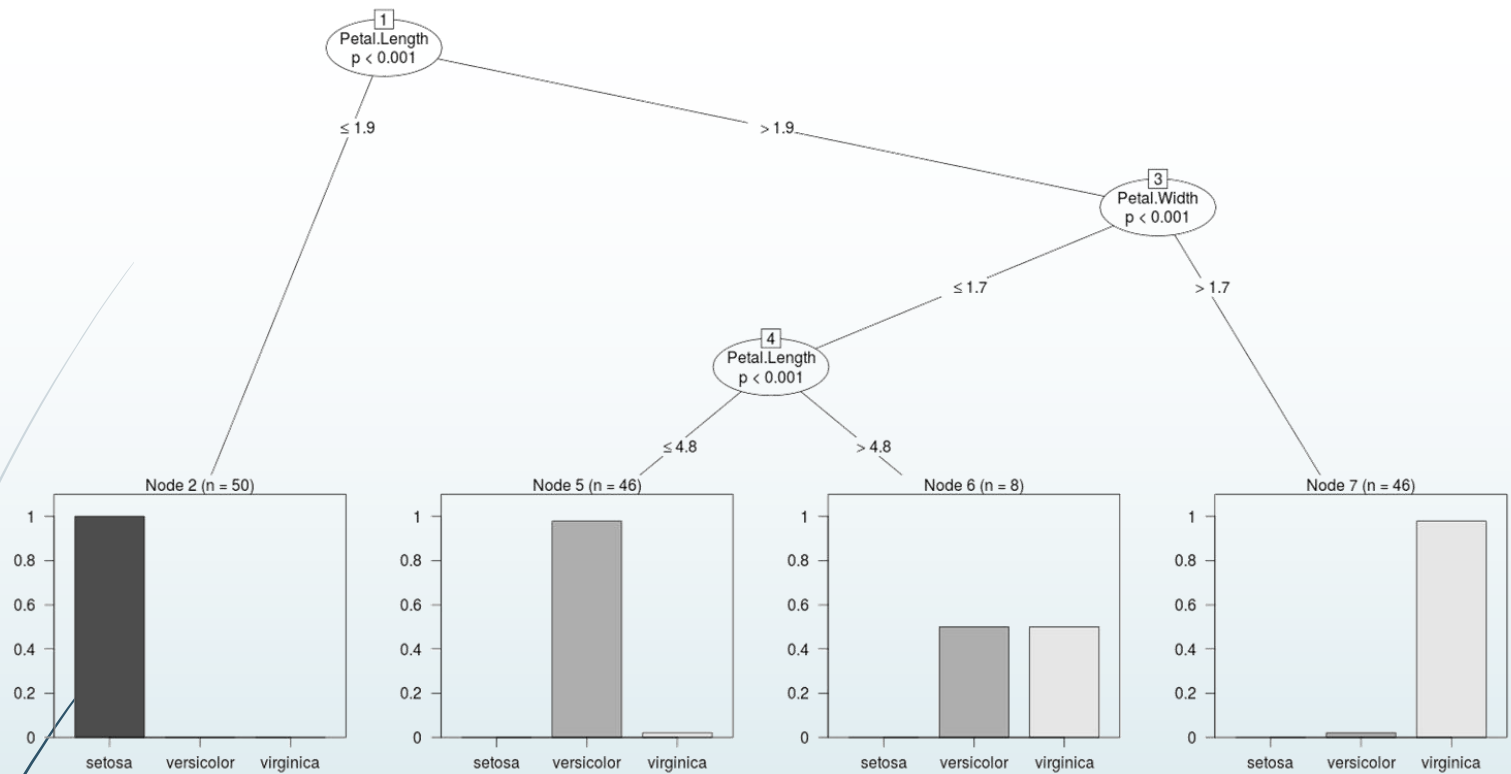
- Sobre el resultado devuelto por **predict()** se pueden aplicar diversas funciones:
 - **confusionMatrix(pred, testData\$Clase)**
 - **sensitivity(pred, testData\$Clase)**
 - **postResample(pred, testData\$Clase)**
 - **multiClassSummary(pred, levels(Clase))**

SVM classification plot



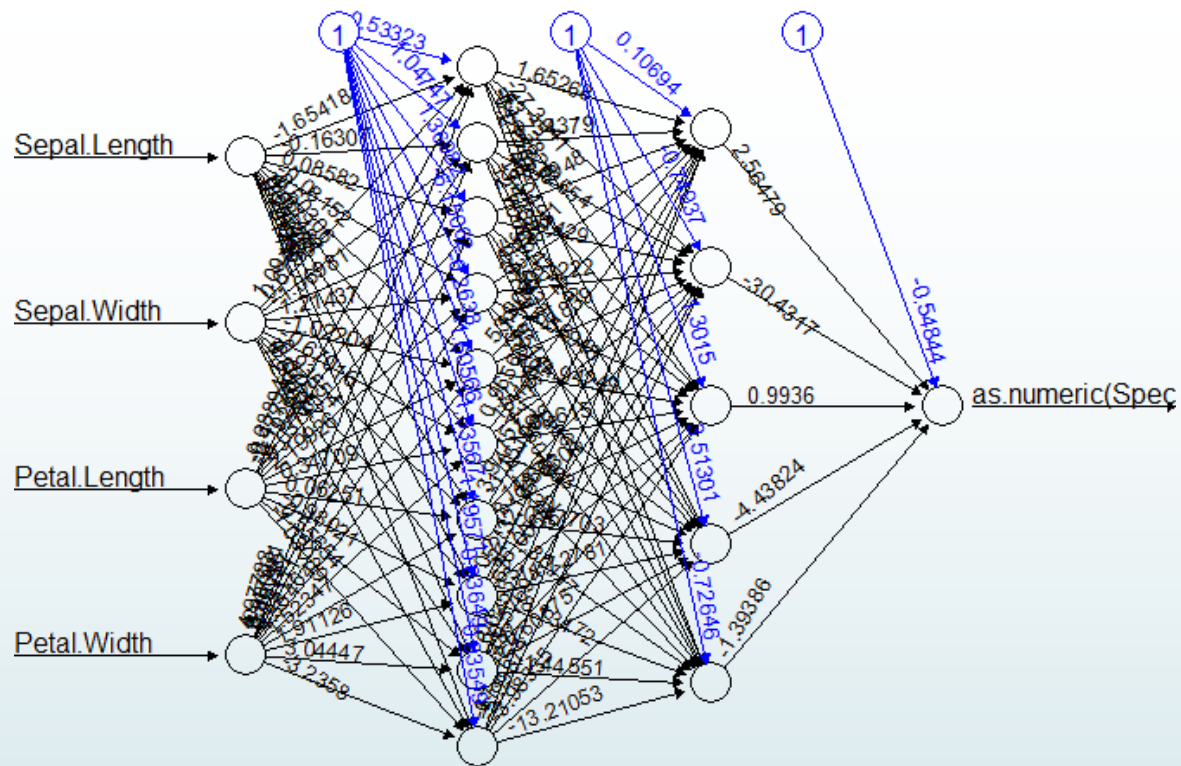
Ejercicios prácticos

Máquinas de vectores soporte



Ejercicios prácticos

Boosting, Bagging, Random forest



Ejercicios prácticos

Deep Learning



Aprendizaje supervisado con R

Francisco Charte