R Evaluacion

Rafael Camarero Rodríguez

2025-03-02

Sección 1: Código fuente y Resultados de la ejecución

La presente sección se ha generado usando Rmarkdown, en dataspell, el código fuente se puede encontrar tanto en formato r como rmd adjuntado en la entrega y en github https://github.com/Rcamrods/R_evaluacion/

```
install.packages("caret", repos = "https://cran.rstudio.com/")
```

```
## Installing package into 'C:/Users/rafac/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'caret' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'caret'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problema al copiar
## C:\Users\rafac\AppData\Local\R\win-library\4.4\00LOCK\caret\libs\x64\caret.dll
## a C:\Users\rafac\AppData\Local\R\win-library\4.4\caret\libs\x64\caret.dll:
## Permission denied
```

```
## Warning: restored 'caret'
```

```
##
## The downloaded binary packages are in
##  C:\Users\rafac\AppData\Local\Temp\Rtmp6to0Fz\downloaded_packages
```

```
library(caret)
```

```
## Cargando paquete requerido: ggplot2
```

```
## Cargando paquete requerido: lattice
```

```
install.packages("AUC", repos = "https://cran.rstudio.com/")
```

```
## Installing package into 'C:/Users/rafac/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'AUC' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\rafac\AppData\Local\Temp\Rtmp6to0Fz\downloaded_packages
```

```
library(AUC)
```

```
## AUC 0.3.2
```

```
## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```
##
## Adjuntando el paquete: 'AUC'
```

```
## The following objects are masked from 'package:caret':
##
##     sensitivity, specificity
```

```
install.packages("ROCR", repos = "https://cran.rstudio.com/")
```

```
## Installing package into 'C:/Users/rafac/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'ROCR' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\rafac\AppData\Local\Temp\Rtmp6to0Fz\downloaded_packages
```

```
library(ROCR)
install.packages("tinytex", repos = "https://cran.rstudio.com/")
```

```
## Installing package into 'C:/Users/rafac/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'tinytex' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\rafac\AppData\Local\Temp\Rtmp6to0Fz\downloaded_packages
```

```
library(tinytex)
```

```
print("1. Cargar datos en R")
```

```
## [1] "1. Cargar datos en R"
```

```
print("====================")
```

```
## [1] "===================="
```

```
set.seed(12345)

data <- read.csv("tic-tac-toe.data.txt", header = FALSE)

names(data) <- c("top_left", "top_mid", "top_right",
                 "mid_left", "mid_mid", "mid_right",
                 "bot_left", "bot_mid", "bot_right",
                 "Class")

missing_vals <- sum(is.na(data))
cat("Numero de valores faltantes:", missing_vals, "\n")
```

```
## Numero de valores faltantes: 0
```

```
print("2. Partir datos en train y test")
```

```
## [1] "2. Partir datos en train y test"
```

```
print("================================")
```

```
## [1] "================================"
```

```
trainIndex <- createDataPartition(data$Class, p = 0.7, list = FALSE)
trainData <- data[trainIndex, ]
testData <- data[-trainIndex, ]
```

```r
cat("Proporcion en entrenamiento:\n")
```

```
## Proporcion en entrenamiento:
```

```r
print(prop.table(table(trainData$Class)))
```

```
##
##  negative  positive
## 0.3467262 0.6532738
```

```r
cat("Proporcion en test:\n")
```

```
## Proporcion en test:
```

```r
print(prop.table(table(testData$Class)))
```

```
##
##  negative  positive
## 0.3461538 0.6538462
```

```r
print("3. Entrenamiento usando validacion cruzada")
```

```
## [1] "3. Entrenamiento usando validacion cruzada"
```

```r
print("==========================================")
```

```
## [1] "=========================================="
```

```r
control <- trainControl(method = "repeatedcv", number = 10, repeats = 1, classProbs = TRUE)

model_nb   <- train(Class ~ ., data = trainData, method = "nb",        trControl = control)
model_dt   <- train(Class ~ ., data = trainData, method = "rpart",     trControl = control)
model_nn   <- train(Class ~ ., data = trainData, method = "nnet",      trControl = control, trace = FALS
model_knn  <- train(Class ~ ., data = trainData, method = "knn",       trControl = control)
model_svm  <- train(Class ~ ., data = trainData, method = "svmLinear", trControl = control)

results <- resamples(list(NB = model_nb, DecisionTree = model_dt,
                          NeuralNet = model_nn, KNN = model_knn,
                          SVM = model_svm))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: NB, DecisionTree, NeuralNet, KNN, SVM
## Number of resamples: 10
##
## Accuracy
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB           0.6029412 0.6937555 0.7014925 0.6936461 0.7126866 0.7575758    0
## DecisionTree 0.7014925 0.7201493 0.7388060 0.7425373 0.7500000 0.8208955    0
## NeuralNet    0.9411765 0.9849050 0.9926471 0.9866543 1.0000000 1.0000000    0
## KNN          0.8787879 0.9253731 0.9259219 0.9314477 0.9514925 0.9705882    0
## SVM          0.9402985 0.9851295 0.9926471 0.9866111 1.0000000 1.0000000    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## NB           0.1546961 0.2939937 0.3340755 0.3157949 0.3611909 0.4661274    0
## DecisionTree 0.2061611 0.2581686 0.3264707 0.3256844 0.3476298 0.5462754    0
## NeuralNet    0.8661417 0.9665284 0.9837476 0.9699558 1.0000000 1.0000000    0
## KNN          0.7158235 0.8263646 0.8318824 0.8434230 0.8900232 0.9343629    0
## SVM          0.8618557 0.9667864 0.9837476 0.9695788 1.0000000 1.0000000    0
```

```
pred_nb  <- predict(model_nb, newdata = testData)
pred_dt  <- predict(model_dt, newdata = testData)
pred_nn  <- predict(model_nn, newdata = testData)
pred_knn <- predict(model_knn, newdata = testData)
pred_svm <- predict(model_svm, newdata = testData)


testData$Class <- factor(testData$Class, levels = c("negative", "positive"))

pred_nb <- factor(pred_nb, levels = levels(testData$Class))
pred_dt <- factor(pred_dt, levels = levels(testData$Class))
pred_nn <- factor(pred_nn, levels = levels(testData$Class))
pred_knn <- factor(pred_knn, levels = levels(testData$Class))
pred_svm <- factor(pred_svm, levels = levels(testData$Class))

eval_nb <- postResample(pred_nb, testData$Class)
eval_dt  <- postResample(pred_dt,  testData$Class)
eval_nn  <- postResample(pred_nn,  testData$Class)
eval_knn <- postResample(pred_knn, testData$Class)
eval_svm <- postResample(pred_svm, testData$Class)

cat("Naive Bayes:\n");  print(eval_nb)
```

```
## Naive Bayes:
```

```
##  Accuracy     Kappa
## 0.6853147 0.2878880
```

```
cat("Decision Tree:\n"); print(eval_dt)
```

```
## Decision Tree:
```

```
##  Accuracy     Kappa
## 0.7412587 0.3370090
```

```
cat("Neural Network:\n"); print(eval_nn)
```

```
## Neural Network:
```

```
##  Accuracy     Kappa
## 0.9720280 0.9379104
```

```
cat("KNN:\n");          print(eval_knn)
```

```
## KNN:
```

```
##  Accuracy     Kappa
## 0.9265734 0.8317930
```

```
cat("SVM:\n");  print(eval_svm)
```

```
## SVM:
```

```
##  Accuracy     Kappa
```

```
## 0.9755245 0.9450151
```

```
print("4. Mostrar matrices de confusion y aniadir el AUC a la tabla de accuracy y kappa")
```

```
## [1] "4. Mostrar matrices de confusion y aniadir el AUC a la tabla de accuracy y kappa"
```

```
print("================================================================================")
```

```
## [1] "================================================================================"
```

```
cm_nb <- confusionMatrix(pred_nb, testData$Class)
cm_dt <- confusionMatrix(pred_dt, testData$Class)
cm_nn <- confusionMatrix(pred_nn, testData$Class)
cm_knn <- confusionMatrix(pred_knn, testData$Class)
cm_svm <- confusionMatrix(pred_svm, testData$Class)

print("Matrices de confusion")
```

```
## [1] "Matrices de confusion"
```

```
cat("Naive Bayes:\n");  print(cm_nb$table)
```

```
## Naive Bayes:
```

```
##           Reference
## Prediction negative positive
##   negative       49       40
##   positive       50      147
```

```
cat("Decision Tree:\n"); print(cm_dt$table)
```

```
## Decision Tree:
```

```
##           Reference
## Prediction negative positive
##   negative       33        8
##   positive       66      179
```

```
cat("Neural Network:\n"); print(cm_nn$table)
```

```
## Neural Network:
```

```
##           Reference
## Prediction negative positive
##   negative       94        3
##   positive        5      184
```

```
cat("KNN:\n");          print(cm_knn$table)
```

```
## KNN:
```

```
##           Reference
## Prediction negative positive
##   negative       81        3
##   positive       18      184
```

```
cat("SVM:\n");  print(cm_svm$table)
```

```
## SVM:
```

```
##           Reference
## Prediction negative positive
```

```
##   negative       92        0
##   positive        7      187
```

```
calcular_auc <- function(model, testData) {
  prob <- predict(model, newdata = testData, type = "prob")
  roc_obj <- roc(prob$positive, testData$Class)
  auc_val <- auc(roc_obj)
  return(auc_val)
}

auc_nb  <- calcular_auc(model_nb,  testData)
auc_dt  <- calcular_auc(model_dt,  testData)
auc_nn  <- calcular_auc(model_nn,  testData)
auc_knn <- calcular_auc(model_knn, testData)
auc_svm <- calcular_auc(model_svm, testData)

print("Tabla con accuracy, kappa y auc")
```

```
## [1] "Tabla con accuracy, kappa y auc"
```

```
resultados_test <- data.frame(
  Modelo = c("Naive Bayes", "Decision Tree", "Neural Network", "Nearest Neighbour", "SVM (linear)"),
  Accuracy = c(eval_nb[1], eval_dt[1], eval_nn[1], eval_knn[1], eval_svm[1]),
  Kappa = c(eval_nb[2], eval_dt[2], eval_nn[2], eval_knn[2], eval_svm[2]),
  AUC = c(auc_nb, auc_dt, auc_nn, auc_knn, auc_svm)
)
print(resultados_test)
```

```
##              Modelo  Accuracy     Kappa       AUC
## 1       Naive Bayes 0.6853147 0.2878880 0.7280830
## 2     Decision Tree 0.7412587 0.3370090 0.7308378
## 3    Neural Network 0.9720280 0.9379104 0.9911414
## 4 Nearest Neighbour 0.9265734 0.8317930 0.9971101
## 5      SVM (linear) 0.9755245 0.9450151 0.9547345
```

```
print("5. Representar curvas ROC")
```

```
## [1] "5. Representar curvas ROC"
```

```
print("==========================")
```

```
## [1] "=========================="
```

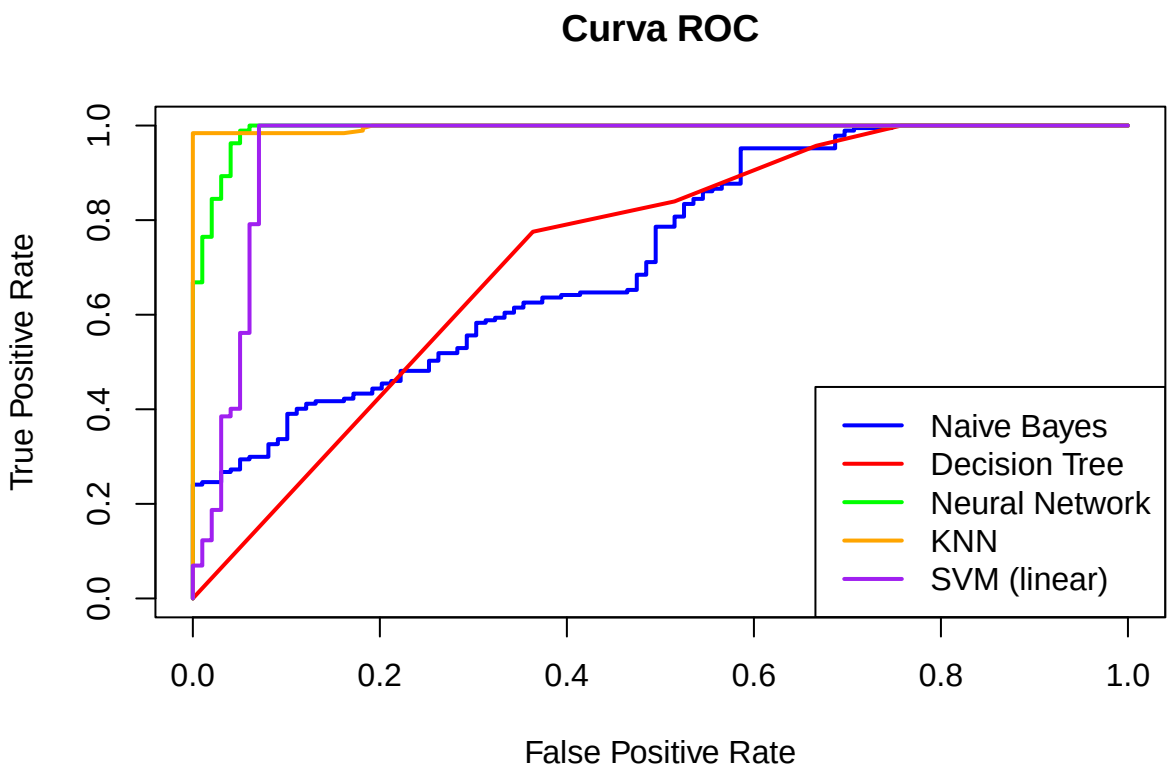
```
trazar_roc <- function(model, testData, color) {
  # 5. a) recalculamos las predicciones con el parámetro prob
  prob <- predict(model, newdata = testData, type = "prob")
  # 5. b) creamos el objeto predicción
  pred <- prediction(prob$positive, testData$Class)
  # 5. c) calculamos TPR y FPR
  perf <- performance(pred, "tpr", "fpr")
  # 5. d) Dibujamos la curva
  plot(perf, col = color, lwd = 2, add = TRUE)
}

prob_nb <- predict(model_nb, newdata = testData, type = "prob")
pred_nb <- prediction(prob_nb$positive, testData$Class)
perf_nb <- performance(pred_nb, "tpr", "fpr")
```

```r
plot(perf_nb, col = "blue", lwd = 2,
     main = "Curva ROC",
     xlab = "False Positive Rate", ylab = "True Positive Rate")

trazar_roc(model_dt,  testData, "red")
trazar_roc(model_nn,  testData, "green")
trazar_roc(model_knn, testData, "orange")
trazar_roc(model_svm, testData, "purple")

legend("bottomright", legend = c("Naive Bayes", "Decision Tree", "Neural Network",
                                 "KNN", "SVM (linear)"),
       col = c("blue", "red", "green", "orange", "purple"), lwd = 2)
```



Sección 2: Preguntas

Q1. ¿Si el modelo A tiene mayor Accuracy que B, siempre tendrá mayor Kappa que B? Justifica tu respuesta.

No hay una correlación directa entre Accuracy y Kappa, el accuracy mide la proporción de aciertos, mientras que kappa compara los aciertos totales contra los valores que podrían darse por el azar, aquí un ejemplo:

```r
library(caret)

actual <- factor(c(rep("Positivo", 90), rep("Negativo", 10)),
                 levels = c("Positivo", "Negativo"))
```

```r
pred_A <- factor(rep("Positivo", 100), levels = c("Positivo", "Negativo"))

pred_B <- c(rep("Positivo", 70), rep("Negativo", 30))
pred_B <- factor(pred_B, levels = c("Positivo", "Negativo"))

cm_A <- confusionMatrix(pred_A, actual)
cm_B <- confusionMatrix(pred_B, actual)

acc_A <- cm_A$overall["Accuracy"]
kappa_A <- cm_A$overall["Kappa"]

acc_B <- cm_B$overall["Accuracy"]
kappa_B <- cm_B$overall["Kappa"]

results_table <- data.frame(
        Model = c("Clasificador A", "Clasificador B"),
        Accuracy = c(as.numeric(acc_A), as.numeric(acc_B)),
        Kappa = c(as.numeric(kappa_A), as.numeric(kappa_B))
)
print(results_table)
```

```
##            Model Accuracy     Kappa
## 1 Clasificador A      0.9 0.0000000
## 2 Clasificador B      0.8 0.4117647
```

En este ejemplo, tenemos un dataset desbalanceado en 90 10, el primer clasificador siempre predice positivo, por lo que su accuracy es 0.9, mientras que su kappa, al ser su accuracy esperado igual al accuracy real, es 0, mientras que el clasificador b predice los 70 primeros valores como positivos, y el resto como negativos, por lo que el accuracy esperado es de pe = (0.9x0.7) + (0.1*0.3) = 0.66, al tener el modelo un accuracy superior al esperado, el kappa es positivo, en este caso 0.411, lo que es superior al clasificador A, teniendo un accuracy inferior.

Q2. ¿Vemos eso en tus resultados? *Respuesta:* No, en mis resultados se puede observar una correlación entre el accuracy y el kappa.

Q3. ¿Te cambian los resultados cuando cambias la semilla?

Sí, al cambiar la semilla se generan particiones diferentes de los datos en entrenamiento y test, lo que modifica el rendimiento de los modelos al ser entrenados y probados con un reparto diferente datos.

Q4. ¿Es recomendable quedarse con los resultados mejores después de cambiar las semillas varias veces? Justifica la respuesta.

No, no es recomendable seleccionar el resultado mejor obtenido, ya que esto implica un sesgo de selección, lo que puede hacer que nuestro modelo solo sea util para ese set de datos concreto.

Q5. ¿Qué modelos puedes descartar porque van a ser siempre subóptimos (asumiendo una buena evaluación)?

Podemos descartar los modelos que solamente predicen una de las clases.

Q6. ¿Por qué puedes descartar esos modelos?

Porque, pese a que puedan obtener "buenas métricas", al tener sobre todo datasets desbalanceados, otros modelos pueden aprender realmente de dichos datos y presentar mejores resultados aunque tengan peores métricas

Sección 3: Uso de LLMs

Se ha empleado ChatGPT-4o para la elaboración de esta práctica, y aunque ha supuesto una buena aproximación inicial, hicieron falta numerosos ajustes para que fuera funcional, ya que su código presentaba errores graves, como el reparto incorrecto entre train y test, hacía referencia a funciones inexistentes o librerias incorrectas, además de fallos en el tratamiento de los datos.