Naive Bayes Classificator with FAMD

Melissa Vargas

2025-11-05

Load data

```
load("dataaaaaaaaaaaaaa.RData")
```

Data preparation:

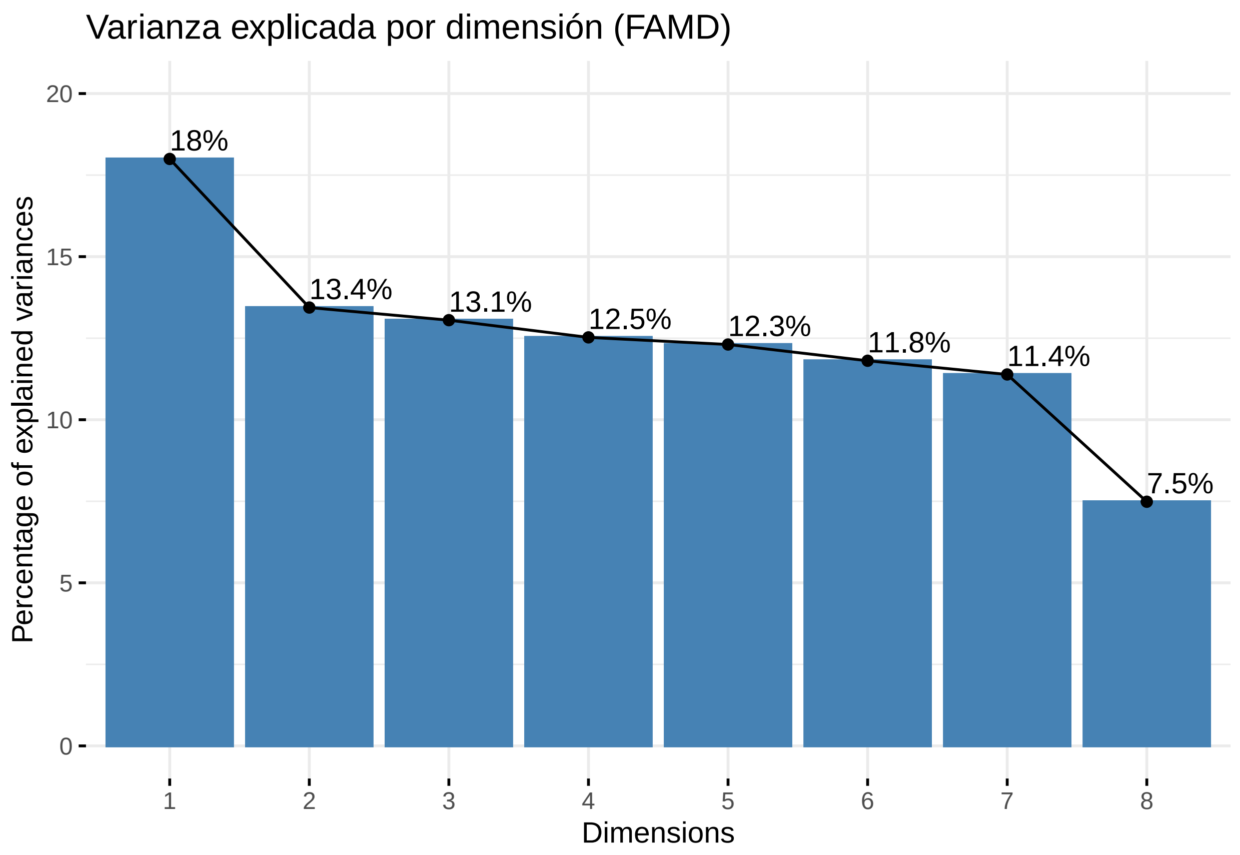
```
train_reducida <- subset(data_reducida, group == "train") # 7000 obs
test_reducida  <- subset(data_reducida,
                         group == "test") # 3000 obs variable respuesta vacia

vars_drop <- c("ID", "Surname", "group")
train_reducida <- train_reducida[, !(names(train_reducida) %in% vars_drop)]
test_reducida  <- test_reducida[,  !(names(test_reducida) %in% vars_drop)]

train_reducida$Exited <- factor(train_reducida$Exited,
                                levels = c("1","0"),
                                labels = c("Yes","No"))
```

Partition

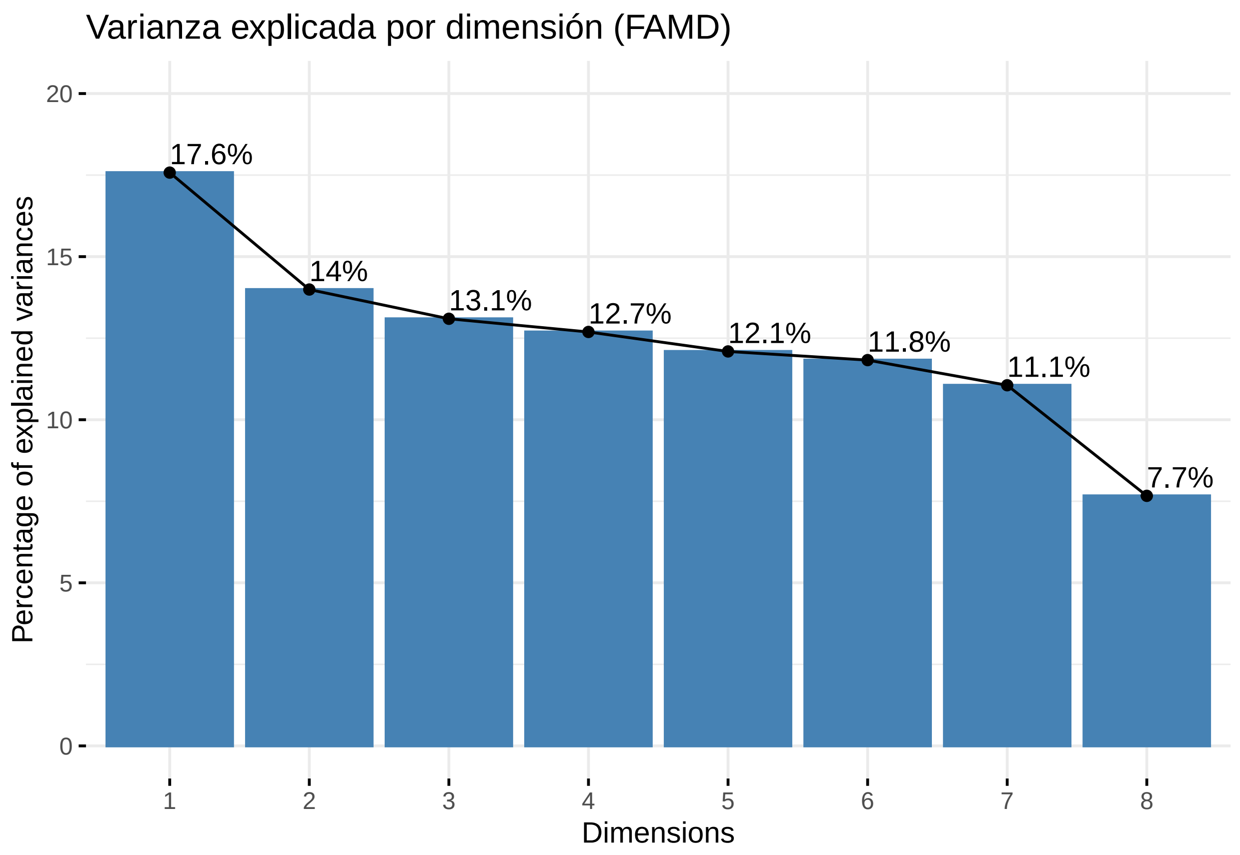
```
set.seed(123)
index <- createDataPartition(train_reducida$Exited, p = 0.7, list = FALSE)
train_reducida2 <- train_reducida[index, ] # train interno
test_reducida2  <- train_reducida[-index, ] # test interno
```

FAMD

```
train_famd <- FAMD(train_reducida2[, !names(train_reducida2) %in% "Exited"],
                   ncp = 9,  # se puede cambiar el nº de comp a mantener
                   graph = FALSE)
fviz_screeplot(train_famd, addlabels = TRUE, ylim = c(0, 20)) +
  ggtitle("Varianza explicada por dimensión (FAMD)")
```



```r
train_famd_coord <- as.data.frame(train_famd$ind$coord)
train_famd_coord$Exited <- train_reducida2$Exited

test_famd <- FAMD(test_reducida2[, !names(test_reducida2) %in% "Exited"],
                   ncp = 9,  # se puede cambiar el nº de comp a mantener
                   graph = FALSE)
test_famd_coord <- as.data.frame(test_famd$ind$coord)
test_famd_coord$Exited <- test_reducida2$Exited
fviz_screeplot(test_famd, addlabels = TRUE, ylim = c(0, 20)) +
  ggtitle("Varianza explicada por dimensión (FAMD)")
```



```r
# esto es porque daba un error porque los nombre de las dims no eran iguales en train y test
names(train_famd_coord) <- c("Dim.1", "Dim.2", "Dim.3", "Dim.4", "Dim.5",
                             "Dim.6", "Dim.7", "Dim.8", "Exited")
```

Modelind

```r
# cross-validation
control <- trainControl(method = "repeatedcv", repeats = 3)
hiperparametros <- data.frame(usekernel = FALSE, fL = 1, adjust = 0)

mod_nb_famd <- train(
  y = train_famd_coord$Exited,
  x = train_famd_coord[, 1:8],  # Usar las primeras 8 dimensiones (se puede cambiar pero primero cambia
  method = "nb",
  tuneGrid = hiperparametros,
  metric = "Accuracy",
  trControl = control
)

# Predicciones
train_pred <- predict(mod_nb_famd, train_famd_coord, type = "raw")
test_pred <- predict(mod_nb_famd, test_famd_coord, type = "raw")

# Matrices de confusión
conf_train <- confusionMatrix(train_pred, train_famd_coord$Exited)
conf_test <- confusionMatrix(test_pred, test_famd_coord$Exited)
```

```r
# F1-score
f1_score <- function(cm){
  precision <- cm$byClass["Precision"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- 2 * (precision * recall) / (precision + recall)
  return(as.numeric(f1))
}

f1_train <- f1_score(conf_train)
f1_test <- f1_score(conf_test)
```

KPI's

```r
# KPIs
resultados_famd <- data.frame(
  Dataset = c("Train", "Test"),
  Error_rate = c(1 - conf_train$overall["Accuracy"], 1 - conf_test$overall["Accuracy"]),
  Accuracy = c(conf_train$overall["Accuracy"], conf_test$overall["Accuracy"]),
  Precision = c(conf_train$byClass["Pos Pred Value"], conf_test$byClass["Pos Pred Value"]),
  Recall_Sensitivity = c(conf_train$byClass["Sensitivity"], conf_test$byClass["Sensitivity"]),
  Specificity = c(conf_train$byClass["Specificity"], conf_test$byClass["Specificity"]),
  F1_Score = c(f1_train, f1_test)
)
resultados_famd
```

```
##   Dataset Error_rate  Accuracy Precision Recall_Sensitivity Specificity
## 1   Train  0.2053061 0.7946939 0.5186722          0.1231527   0.9701416
## 2    Test  0.2066667 0.7933333 0.5035971          0.1609195   0.9585586
##    F1_Score
## 1 0.1990446
## 2 0.2439024
```