Parte 1: Cargar bibliotecas y cargar datos

r

Copy code

install.packages(c("caret", "MASS", "glmnet", "boot"))

library(tidyverse)

library(caret)

library(MASS)

library(glmnet)

library(boot)

datos <- read.csv("datos\_salud.csv")

En esta parte, se instalan y cargan las bibliotecas necesarias para el análisis de datos. Luego, se carga un conjunto de datos desde un archivo CSV llamado "datos\_salud.csv".

Parte 2: Preprocesamiento de datos

r

Copy code

set.seed(189)

datos\_muestra <- datos %>% sample\_frac(0.01)

datos\_muestra$DiabetesBinaria <- make.names(factor(ifelse(datos\_muestra$Diabetes > 0, 1, 0)))

datos\_muestra$DiabetesBinaria <- factor(datos\_muestra$DiabetesBinaria, levels = c("X0", "X1"))

En esta parte, se toma el 1% de los datos de manera aleatoria para reducir el tamaño del conjunto de datos y hacer el análisis más manejable. Se crea una nueva variable llamada "DiabetesBinaria" que representa si una persona tiene diabetes o no (1 o 0). Luego, esta variable se convierte en un factor con niveles "X0" y "X1".

Parte 3: Dividir datos en conjuntos de entrenamiento y prueba

r

Copy code

set.seed(40)

indices\_entrenamiento <- createDataPartition(datos\_muestra$DiabetesBinaria, p = 0.8, list = FALSE, times = 1)

datos\_entrenamiento <- datos\_muestra[indices\_entrenamiento, ]

datos\_prueba <- datos\_muestra[-indices\_entrenamiento, ]

En esta sección, los datos se dividen en un conjunto de entrenamiento (80% de los datos) y un conjunto de prueba (20% de los datos) utilizando el método createDataPartition de la biblioteca caret. Esto es fundamental para evaluar la capacidad predictiva del modelo en datos no vistos.

Parte 4: Entrenar el modelo KNN (K-Nearest Neighbors)

r

Copy code

ctrl <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = twoClassSummary)

set.seed(20)

modelo\_knn <- train(DiabetesBinaria ~ ., data = datos\_entrenamiento, method = "knn", trControl = ctrl, tuneLength = 10)

En esta parte, se configura un control de entrenamiento para la validación cruzada de 10 veces (method = "cv", number = 10). Se entrena un modelo de clasificación KNN utilizando el conjunto de entrenamiento y se evalúan diferentes valores de k (de 1 a 10) para encontrar el mejor modelo. Los resultados se almacenan en la variable modelo\_knn.

Parte 5: Hacer predicciones y evaluar el modelo KNN

r

Copy code

predicciones\_knn <- predict(modelo\_knn, newdata = datos\_prueba)

confusionMatrix(predicciones\_knn, datos\_prueba$DiabetesBinaria)

En esta sección, el modelo KNN entrenado se utiliza para hacer predicciones en el conjunto de prueba. Luego, se evalúa el modelo utilizando una matriz de confusión para comparar las predicciones del modelo con las verdaderas etiquetas de clase en el conjunto de prueba.

Las partes restantes del código son similares y se utilizan para realizar regresiones lineales y multilíneas en diferentes variables de respuesta (como BMI, Salud Mental y Salud Física) utilizando validación cruzada para evaluar la calidad del ajuste del modelo.

Recuerda que este código es un ejemplo y necesitarás adaptarlo según tus datos reales y los objetivos de tu análisis. Si tienes preguntas específicas sobre alguna parte del código o si necesitas más explicaciones, no dudes en preguntar.