Proyecto Final1.0 v2

Carlos Alvarado Ronald Guerra Rene Hernandez

r Sys.Date()

```
dataset = read.csv("train.csv")
```

```
# Cargando las librerías necesarias
library(mice)
```

```
## Warning: package 'mice' was built under R version 4.2.3
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# Asumiendo que tu dataset se llama 'dataset'

# Imputación de datos faltantes (ya lo has hecho)
tempData <- mice(dataset, method='pmm', m=5)
```

```
##
##  iter imp variable
##   1   1  total_bedrooms
##   1   2  total_bedrooms
##   1   3  total_bedrooms
##   1   4  total_bedrooms
##   1   5  total_bedrooms
##   2   1  total_bedrooms
##   2   2  total_bedrooms
##   2   3  total_bedrooms
##   2   4  total_bedrooms
##   2   5  total_bedrooms
##   3   1  total_bedrooms
##   3   2  total_bedrooms
##   3   3  total_bedrooms
##   3   4  total_bedrooms
##   3   5  total_bedrooms
##   4   1  total_bedrooms
##   4   2  total_bedrooms
##   4   3  total_bedrooms
##   4   4  total_bedrooms
##   4   5  total_bedrooms
##   5   1  total_bedrooms
##   5   2  total_bedrooms
##   5   3  total_bedrooms
##   5   4  total_bedrooms
##   5   5  total_bedrooms
```

```
## Warning: Number of logged events: 1
```

```r
dataset <- complete(tempData)

# Asegurándose de que 'ocean_proximity' sea un factor
dataset$ocean_proximity <- as.factor(dataset$ocean_proximity)

# Dividiendo el dataset en conjuntos de entrenamiento y prueba
set.seed(123)
trainIndex <- createDataPartition(dataset$median_house_value, p=0.8, list=FALSE)
trainData <- dataset[trainIndex,]
testData <- dataset[-trainIndex,]

# Creando un modelo de Random Forest
set.seed(123)
rf_model <- randomForest(median_house_value ~ ., data=trainData, ntree=100)
```

```r
# Haciendo predicciones en el conjunto de prueba
predictions <- predict(rf_model, newdata=testData)

# Evaluando el rendimiento del modelo
RMSE <- sqrt(mean((testData$median_house_value - predictions)^2))
print(paste("Root Mean Squared Error:", RMSE))
```

```
## [1] "Root Mean Squared Error: 47986.7613321812"
```

```r
# Puedes usar el modelo para hacer predicciones en nuevos datos
# new_predictions <- predict(rf_model, newdata=new_data)
print(paste("% que representa el valor del error sobre la media:", (RMSE / mean(dataset$median_house_va
```

```
## [1] "% que representa el valor del error sobre la media: 23.1961646920024"
```

```r
dataset = read.csv("train.csv")

# Cargando las librerías necesarias
library(mice)
library(caret)

# Asumiendo que tu dataset se llama 'dataset'

# Imputación de datos faltantes
tempData <- mice(dataset, method='pmm', m=5)
```

```
##
##  iter imp variable
##   1   1  total_bedrooms
##   1   2  total_bedrooms
##   1   3  total_bedrooms
##   1   4  total_bedrooms
##   1   5  total_bedrooms
##   2   1  total_bedrooms
##   2   2  total_bedrooms
##   2   3  total_bedrooms
##   2   4  total_bedrooms
##   2   5  total_bedrooms
##   3   1  total_bedrooms
##   3   2  total_bedrooms
##   3   3  total_bedrooms
##   3   4  total_bedrooms
##   3   5  total_bedrooms
##   4   1  total_bedrooms
##   4   2  total_bedrooms
##   4   3  total_bedrooms
##   4   4  total_bedrooms
##   4   5  total_bedrooms
##   5   1  total_bedrooms
##   5   2  total_bedrooms
##   5   3  total_bedrooms
```

```
##   5   4  total_bedrooms
##   5   5  total_bedrooms
```

```
## Warning: Number of logged events: 1
```

```
dataset <- complete(tempData)

# Asegurándose de que 'ocean_proximity' sea un factor
dataset$ocean_proximity <- as.factor(dataset$ocean_proximity)

# Dividiendo el dataset en conjuntos de entrenamiento y prueba
set.seed(123)
trainIndex <- createDataPartition(dataset$median_house_value, p=0.8, list=FALSE)
trainData <- dataset[trainIndex,]
testData <- dataset[-trainIndex,]

# Creando un modelo de regresión lineal
lm_model <- lm(median_house_value ~ ., data=trainData)

# Haciendo predicciones en el conjunto de prueba
predictions <- predict(lm_model, newdata=testData)

# Evaluando el rendimiento del modelo
RMSE <- sqrt(mean((testData$median_house_value - predictions)^2))
print(paste("Root Mean Squared Error:", RMSE))
```

```
## [1] "Root Mean Squared Error: 68434.0888864998"
```

```
# Puedes usar el modelo para hacer predicciones en nuevos datos
# new_predictions <- predict(lm_model, newdata=new_data)
print(paste("% que representa el valor del error sobre la media:", (RMSE / mean(dataset$median_house_va
```

```
## [1] "% que representa el valor del error sobre la media: 33.0801319424286"
```

```
# Cargando las librerías necesarias
library(mice)
library(caret)
dataset = read.csv("train.csv")
# Asumiendo que tu dataset se llama 'dataset'

# Imputación de datos faltantes
tempData <- mice(dataset, method='pmm', m=5)
```

```
##
##  iter imp variable
##   1   1  total_bedrooms
##   1   2  total_bedrooms
##   1   3  total_bedrooms
##   1   4  total_bedrooms
##   1   5  total_bedrooms
##   2   1  total_bedrooms
```

```
##   2   2  total_bedrooms
##   2   3  total_bedrooms
##   2   4  total_bedrooms
##   2   5  total_bedrooms
##   3   1  total_bedrooms
##   3   2  total_bedrooms
##   3   3  total_bedrooms
##   3   4  total_bedrooms
##   3   5  total_bedrooms
##   4   1  total_bedrooms
##   4   2  total_bedrooms
##   4   3  total_bedrooms
##   4   4  total_bedrooms
##   4   5  total_bedrooms
##   5   1  total_bedrooms
##   5   2  total_bedrooms
##   5   3  total_bedrooms
##   5   4  total_bedrooms
##   5   5  total_bedrooms
```

```
## Warning: Number of logged events: 1
```

```r
dataset <- complete(tempData)

# Asegurándose de que 'ocean_proximity' sea un factor
dataset$ocean_proximity <- as.factor(dataset$ocean_proximity)

# Dividiendo el dataset en conjuntos de entrenamiento y prueba
set.seed(123)
trainIndex <- createDataPartition(dataset$median_house_value, p=0.8, list=FALSE)
trainData <- dataset[trainIndex,]
testData <- dataset[-trainIndex,]

# Normalizando los datos (excluyendo la variable objetivo y las categóricas)
num_vars <- sapply(trainData, is.numeric)
num_vars["median_house_value"] <- FALSE
num_vars["ocean_proximity"] <- FALSE

preProcValues <- preProcess(trainData[, num_vars], method = c("center", "scale"))
trainData[, num_vars] <- predict(preProcValues, trainData[, num_vars])
testData[, num_vars] <- predict(preProcValues, testData[, num_vars])

# Creando un modelo de regresión lineal
lm_model <- lm(median_house_value ~ ., data=trainData)

# Haciendo predicciones en el conjunto de prueba
predictions <- predict(lm_model, newdata=testData)

# Evaluando el rendimiento del modelo
RMSE <- sqrt(mean((testData$median_house_value - predictions)^2))
print(paste("Root Mean Squared Error:", RMSE))
```

```
## [1] "Root Mean Squared Error: 68429.3731555065"
```

```r
# Puedes usar el modelo para hacer predicciones en nuevos datos
# new_predictions <- predict(lm_model, newdata=new_data)
print(paste("% que representa el valor del error sobre la media:", (RMSE / mean(dataset$median_house_va
```

```
## [1] "% que representa el valor del error sobre la media: 33.0778524205412"
```

```r
# Cargando las librerías necesarias
library(mice)
library(caret)

# Leyendo el dataset
dataset <- read.csv("train.csv")

# Imputación de datos faltantes
tempData <- mice(dataset, method='pmm', m=5)
```

```
##
##  iter imp variable
##   1   1  total_bedrooms
##   1   2  total_bedrooms
##   1   3  total_bedrooms
##   1   4  total_bedrooms
##   1   5  total_bedrooms
##   2   1  total_bedrooms
##   2   2  total_bedrooms
##   2   3  total_bedrooms
##   2   4  total_bedrooms
##   2   5  total_bedrooms
##   3   1  total_bedrooms
##   3   2  total_bedrooms
##   3   3  total_bedrooms
##   3   4  total_bedrooms
##   3   5  total_bedrooms
##   4   1  total_bedrooms
##   4   2  total_bedrooms
##   4   3  total_bedrooms
##   4   4  total_bedrooms
##   4   5  total_bedrooms
##   5   1  total_bedrooms
##   5   2  total_bedrooms
##   5   3  total_bedrooms
##   5   4  total_bedrooms
##   5   5  total_bedrooms
```

```
## Warning: Number of logged events: 1
```

```r
dataset <- complete(tempData)

# Guardar la columna median_house_value
median_house_value <- dataset$median_house_value

# Convertir 'ocean_proximity' en variables dummy
```

```r
dummies <- dummyVars(~ ., data=dataset)
dataset <- data.frame(predict(dummies, newdata = dataset))

# Agregar de nuevo la columna median_house_value
dataset$median_house_value <- median_house_value

# Dividiendo el dataset en conjuntos de entrenamiento y prueba
set.seed(123)
trainIndex <- createDataPartition(dataset$median_house_value, p=0.8, list=FALSE)
trainData <- dataset[trainIndex,]
testData <- dataset[-trainIndex,]

# Normalizando los datos (excluyendo la variable objetivo)
num_vars <- colnames(trainData) != "median_house_value"

preProcValues <- preProcess(trainData[, num_vars], method = c("center", "scale"))
trainData[, num_vars] <- predict(preProcValues, trainData[, num_vars])
testData[, num_vars] <- predict(preProcValues, testData[, num_vars])

# Creando un modelo de regresión lineal
lm_model <- lm(median_house_value ~ ., data=trainData)

# Haciendo predicciones en el conjunto de prueba
predictions <- predict(lm_model, newdata=testData)
```

```
## Warning in predict.lm(lm_model, newdata = testData): prediction from a
## rank-deficient fit may be misleading
```

```r
# Evaluando el rendimiento del modelo
RMSE <- sqrt(mean((testData$median_house_value - predictions)^2))
print(paste("Root Mean Squared Error:", RMSE))
```

```
## [1] "Root Mean Squared Error: 68429.3731555065"
```

```r
# Puedes usar el modelo para hacer predicciones en nuevos datos
# new_predictions <- predict(lm_model, newdata=new_data)
print(paste("% que representa el valor del error sobre la media:", (RMSE / mean(dataset$median_house_va
```

```
## [1] "% que representa el valor del error sobre la media: 33.0778524205412"
```

```r
# Cargando las librerías necesarias
library(mice)
library(caret)

# Leyendo el dataset
dataset <- read.csv("train.csv")

# Imputación de datos faltantes
tempData <- mice(dataset, method='pmm', m=2)
```

```
##
```

```
##  iter imp variable
##   1   1  total_bedrooms
##   1   2  total_bedrooms
##   2   1  total_bedrooms
##   2   2  total_bedrooms
##   3   1  total_bedrooms
##   3   2  total_bedrooms
##   4   1  total_bedrooms
##   4   2  total_bedrooms
##   5   1  total_bedrooms
##   5   2  total_bedrooms

## Warning: Number of logged events: 1
```

```
dataset <- complete(tempData)

# Ingeniería de características
# Crear una nueva característica, por ejemplo, habitaciones por hogar
dataset$rooms_per_household <- dataset$total_rooms / dataset$households

# Convertir 'ocean_proximity' en variables dummy
dummies <- dummyVars(~ ., data=dataset)
dataset <- data.frame(predict(dummies, newdata = dataset))

# Normalizando los datos (excluyendo la variable objetivo)
num_vars <- colnames(dataset) != "median_house_value"
preProcValues <- preProcess(dataset[, num_vars], method = c("center", "scale"))
dataset[, num_vars] <- predict(preProcValues, dataset[, num_vars])

# Dividiendo el dataset en conjuntos de entrenamiento y prueba
set.seed(123)
trainIndex <- createDataPartition(dataset$median_house_value, p=0.8, list=FALSE)
trainData <- dataset[trainIndex,]
testData <- dataset[-trainIndex,]

# Creando un modelo de regresión lineal con validación cruzada
control <- trainControl(method="cv", number=5)
set.seed(123)
lm_model <- train(median_house_value ~ ., data=trainData, method="lm", trControl=control)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
# Mostrando los resultados de la validación cruzada
print(lm_model)
```

```
## Linear Regression
##
## 11560 samples
##    15 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9250, 9249, 9247, 9248, 9246
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   67485.41  0.6563775  49376.88
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Haciendo predicciones en el conjunto de prueba
predictions <- predict(lm_model, newdata=testData)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```r
# Evaluando el rendimiento del modelo
RMSE <- sqrt(mean((testData$median_house_value - predictions)^2))
print(paste("Root Mean Squared Error:", RMSE))
```

```
## [1] "Root Mean Squared Error: 68402.062495095"
```

```r
print(paste("% que representa el valor del error sobre la media:", (RMSE / mean(dataset$median_house_va
```

```
## [1] "% que representa el valor del error sobre la media: 33.064650809114"
```