Documentación del Análisis usando Random Forest

Este documento detalla el proceso completo de análisis y modelado utilizando el algoritmo Random Forest en R. Incluye la carga de datos, preprocesamiento, entrenamiento del modelo, evaluación y ajustes.

---

1. Cargar librerías necesarias

```
library(tidyverse)
library(caret)
library(randomForest)
library(PerformanceAnalytics)
```

Las librerías mencionadas permiten realizar tareas de manipulación de datos, visualización y análisis.

---

2. Carga del conjunto de datos

El conjunto de datos se carga desde un archivo CSV. A continuación, se revisan sus propiedades básicas:

```
datos = read.csv("./files/datosLaboratorio04.csv")

head(datos)
str(datos)
summary(datos)
```

Estas funciones permiten explorar las primeras filas, la estructura y un resumen estadístico de los datos.

---

3. División de los datos en conjuntos de entrenamiento y prueba

Los datos se dividen en un 70% para entrenamiento y un 30% para prueba:

```
set.seed(123)
tamano <- nrow(datos)
training <- round(tamano * 0.7)
indices <- sample(1:tamano, size = training)

train_data <- datos[indices, ]
test_data <- datos[-indices, ]
```

---

4. Selección del modelo y normalización

Se selecciona Random Forest como modelo. Los datos se normalizan para un mejor rendimiento:

```
preProc <- preProcess(train_data[, -ncol(train_data)], method = c("center",
"scale"))
train_data_scaled <- predict(preProc, train_data)
test_data_scaled <- predict(preProc, test_data)
```

---

5. Entrenamiento del modelo Random Forest

El modelo Random Forest se entrena para predecir la variable `day_minutes`:

```
rf_model <- randomForest(
  day_minutes ~ .,
  data = train_data_scaled,
  importance = TRUE,
  ntree = 500
)

print(rf_model)
```

---

6. Evaluación del modelo

Las predicciones se comparan con los valores reales mediante métricas como MSE y R-squared:

```
predictions <- predict(rf_model, test_data_scaled)

mse <- mean((predictions - test_data_scaled$day_minutes)^2)
r2 <- 1 - (sum((predictions - test_data_scaled$day_minutes)^2) /
             sum((mean(test_data_scaled$day_minutes) -
test_data_scaled$day_minutes)^2))

cat("MSE: ", mse, "\n")
cat("R-squared: ", r2, "\n")
```

También se genera un gráfico comparativo de predicciones versus valores reales:

```
plot(predictions, test_data_scaled$day_minutes,
     main = "Predicción vs Valores Reales",
     xlab = "Predicción", ylab = "Real")
abline(0, 1, col = "red")
```

---

7. Ajuste de parámetros

Se realiza un ajuste fino del modelo utilizando una búsqueda por malla:

```
tune_grid <- expand.grid(mtry = c(2, 3, 5))

tuned_model <- train(
  day_minutes ~ .,
  data = train_data_scaled,
  method = "rf",
  tuneGrid = tune_grid,
  trControl = trainControl(method = "cv", number = 5)
)

print(tuned_model)
```

---

8. Interpretación del modelo

La importancia de las variables se visualiza con un gráfico y en forma tabular:

```
varImpPlot(rf_model, main = "Importancia de las variables")
variable_importance <- importance(rf_model)
print(variable_importance)
```

Además, se utilizan valores SHAP para explicar las predicciones:

```
library(DALEX)
explainer <- explain(rf_model, data = train_data_scaled[, -
ncol(train_data_scaled)], y = train_data_scaled$day_minutes)
shap_values <- predict_parts(explainer, new_observation = test_data_scaled[1, ],
type = "shap")
plot(shap_values)
```

---

9. Ejemplo de predicción con nuevos datos

Se generan predicciones numéricas para un conjunto de datos nuevos:

```
new_data <- data.frame(
  account_length = c(0.5, 1.2),
  voicemail_message_count = c(1.0, -0.3),
  day_minutes = c(1.5, 0.8),
  day_calls = c(0.5, 0.9),
  day_charge = c(1.6, 0.7),
  evening_minutes = c(-0.1, 0.2),
```

```
  evening_calls = c(-0.2, 0.4),
  evening_charge = c(-0.1, 0.3),
  night_minutes = c(0.8, 1.0),
  night_calls = c(-0.4, 0.2),
  night_charge = c(0.9, 1.1),
  intl_minutes = c(0.1, -0.5),
  intl_calls = c(0.2, -0.4),
  intl_charge = c(0.1, -0.3),
  customer_service_calls = c(-0.5, 0.5)
)

new_data_scaled <- predict(preProc, new_data)
numeric_predictions <- predict(rf_model, new_data_scaled)

print("Predicciones numéricas:")
print(numeric_predictions)
```

Notas Finales: Este análisis Random Forest incluye exploración de datos, evaluación de desempeño, ajuste de hiperparámetros e interpretación, con ejemplos aplicados a nuevos datos.