Proyecto Klustera

Juan Carlos Lozano

14/12/2020

Este documento es el proyecto final del curso de Ciencia de Datos impartido por DEV.F y pretende resolver la siguiente pregunta:

¿Cómo saber si una conexión registrada es efectivamente un visitante?

Para resolver esta pregunta se siguió el proceso detallado en este documento.

Librerías utilizadas

Se utilizó el paquete de librerías de Tidyverse para el filtrado, selección y trasnformación de las bases de datos utilizadas. Así como gmodels y C50 para el modelado de las predicciones y, una vez generado el modelo, para correrlo en la base de datos y resolver la pregunta.

Datasets a analizar

Se utilizaron dos datasets con las mismas variables, con la excepción de que uno no incluye la columna 'Visitor', misma que será añadida producto del análisis final de este ejercicio.

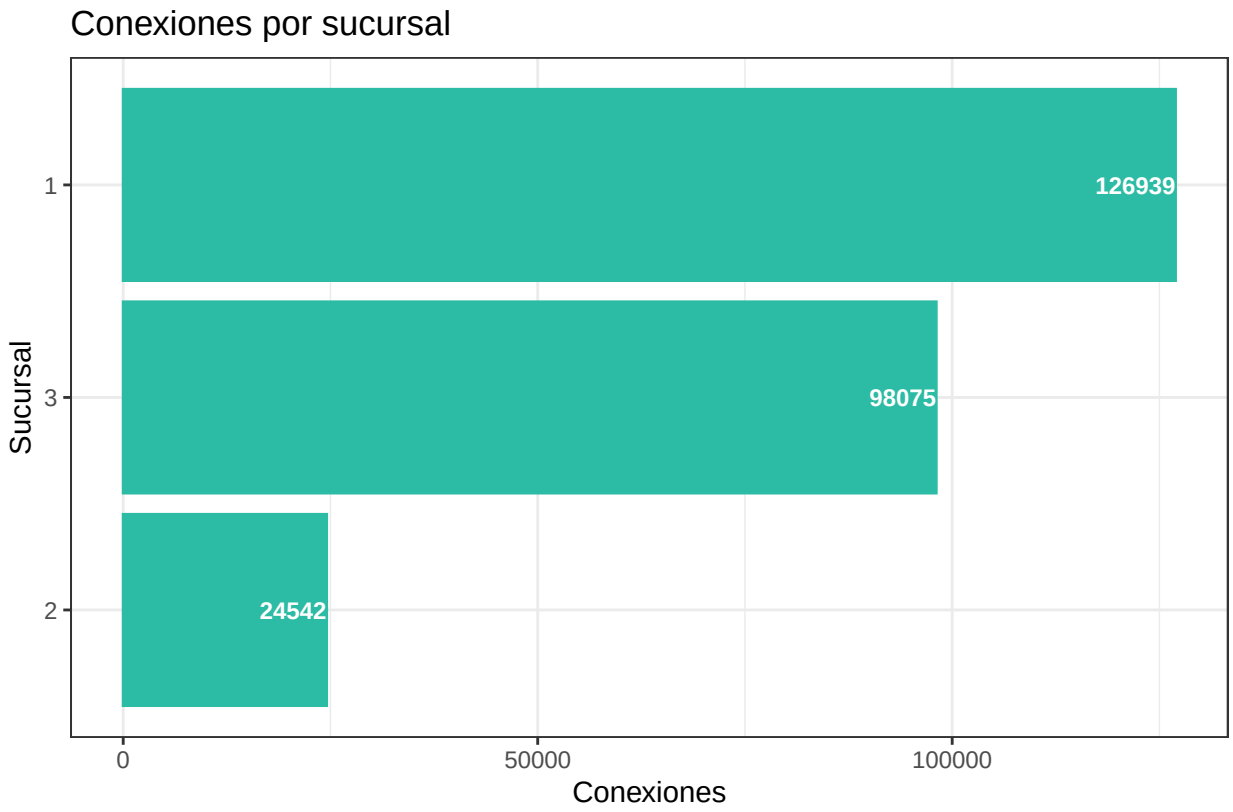
Análisis Exploratorio

Como primera etapa del análisis se definieron cinco preguntas a explorar.

1. ¿Cuál es la sucursal que recibe mas conexiones?

Utilizando el dataset 'e.csv', en el que ya se sabe cuál es un visitante. Se filtró para dejar solo las filas que sí crresponden a visitantes, de manera posterior se agrupó por sucursal, se colapsó por número de conexiones y se ordenó de manera descendente según el número de conexiones.

Como la gráfica muestra, la sucursal 1 es la que recibe más visitantes.

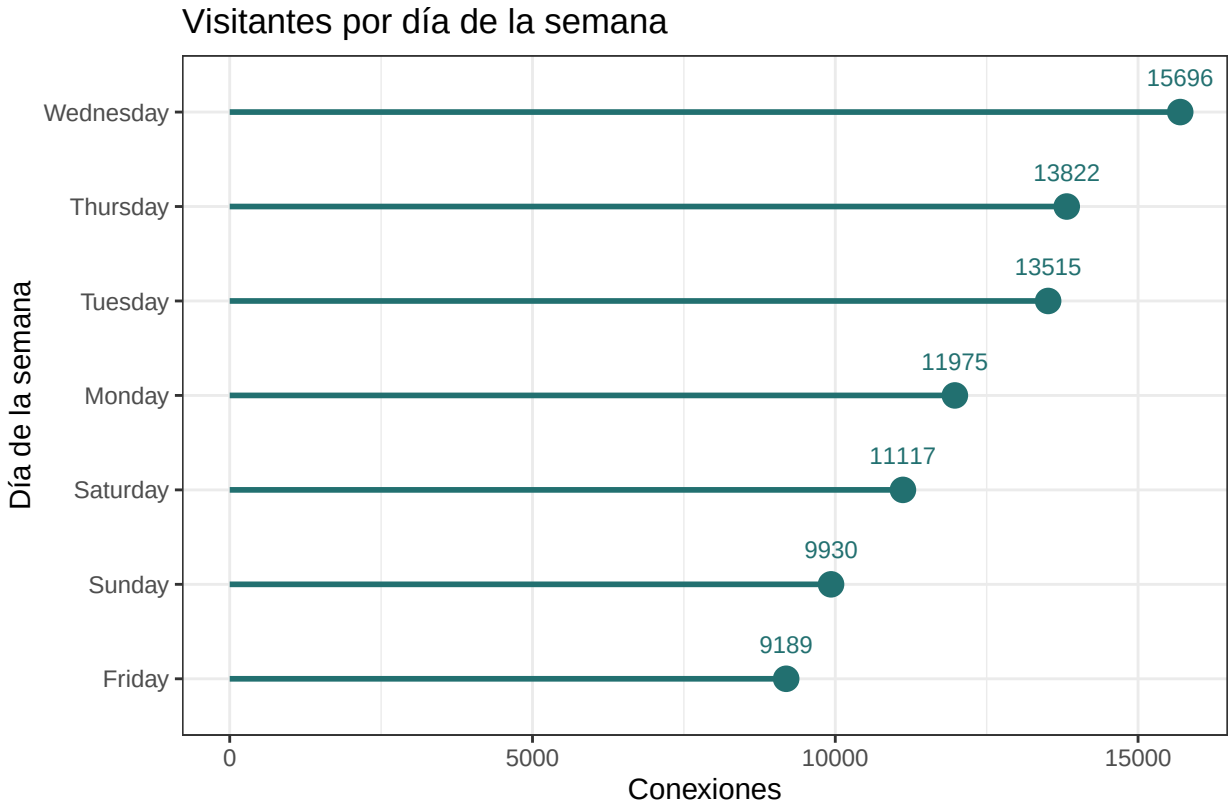


Fuente: base de datos de Klustera

2. ¿Qué día de la semana tenemos mas visitantes?

De la misma manera, se utilizó el dataset 'e.csv'. De nuevo se filtró para dejar solamente las filas que corresponden a visitantes, se agrupó por día de la semana y se colapsó según el número de visitantes por día.

Como la gráfica muestra, la sucursal 1 es la que recibe más visitantes.



Fuente: base de datos de Klustera

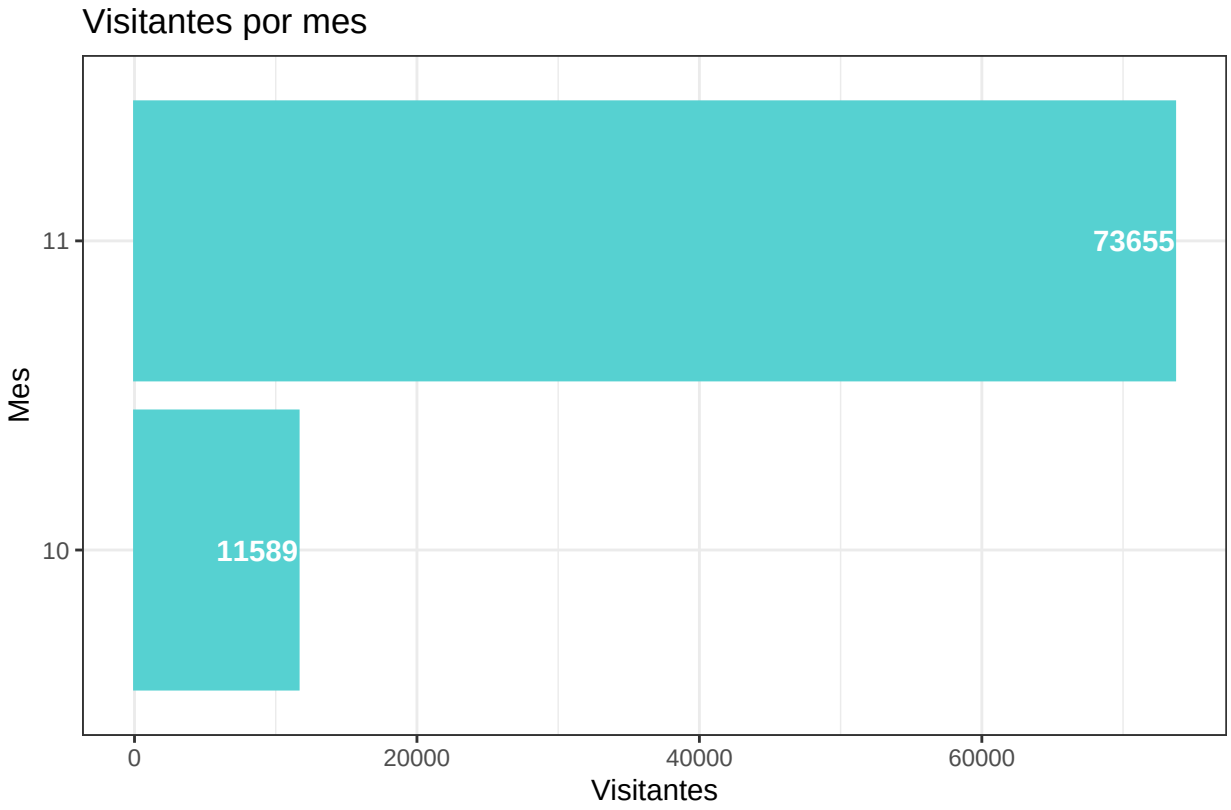
3. ¿Cuál es el tiempo promedio de conexión de un visitante?

También se respondió la pregunta respecto al tiempo promedio que dura la conexión de un visitante. Para obtener la respuesta se filtró, una vez más, para conservar solo las filas de visitantes, para luego calcular el tiempo promedio de las sesiones. Resultando que el tiempo promedio de conexión de un visitante es de: 6943.75 segundos

4. ¿Cuantas persona por mes han realizado visitas?

De nueva cuenta se utilizó el dataset 'e.csv'. Al igual que en las preguntas anteriores, se filtró para mantener solo aquellas filas con visitantes confirmado para después filtrar según el mes de visita, colapsar por número de visitantes y ordernar de manera descendente según este mismo factor.

De esta manera, podemos observar que el mes 11, Noviembre, es el que más visitantes recibió en el periodo evalu-



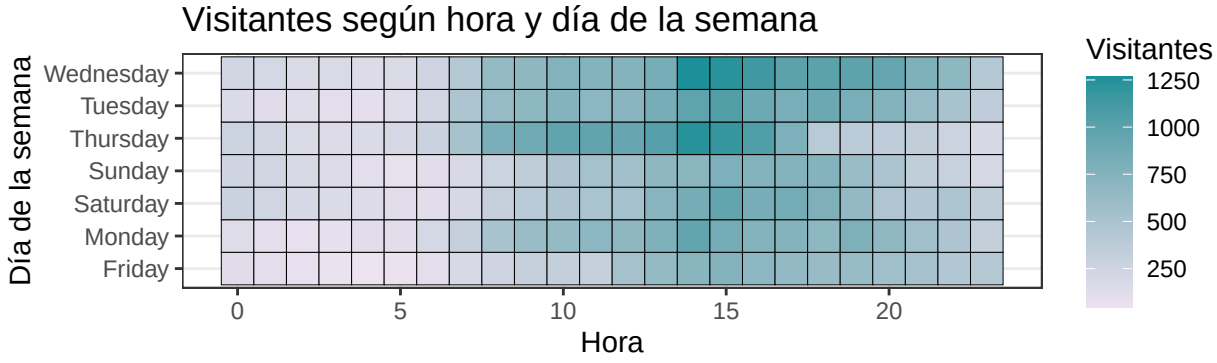
Fuente: base de datos de Klustera

ado.

5. ¿A qué hora se registran más visitantes?

Aunque la pregunta expresa responde a la hora con más visitantes, se consideró que quizá sería más útil conocer no solo la hora, sino también el día en el que se recibieron más visitantes. Por lo que, para responder a esta última pregunta fue necesario filtrar, de la misma manera, para conservar solo las filas con visitantes. Posteriormente se agrupó por hora y día de la semana y se colapsó según el número visitantes.

De esta manera, podemos observar que el mes 11, Noviembre, es el que más visitantes recibió en el periodo evaluado.



Entrenamiento para el modelo de Árboles de Decisión

Una vez concluido el anáisis exploratorio, se procedió a entrenar el modelo de Árboles de Decisión. El primer paso es tomar los datasets que hemos utilizado hasta ahora y eliminar aquellas columnas que no aportan al modelo.En este caso, se eliminaron las columnas 1 y 2, que corresponden al orden consecutivo de las

observaciones y al id de usuario, respectivamente; se eliminó también la columna 6, correspondiente al nombre de los días de la semana.

El dataset datos_e será el de entrenamiento y prueba, mientras que el dataset sujeto del modelo es datos_v.

```
datos_e <- read.csv(paste(dir1, 'databases', 'e.csv', sep = '/')) %>%
  select(-1, -2, -6)

datos_v <- read.csv(paste(dir1, 'databases', 'v.csv', sep = '/')) %>%
  select(-1, -2, -6)
```

Antes de continuar, el modelo requiere que la columna visitor sea de factores, por lo que se ejecuta el siguiente código para transformarla.

```
datos_e$visitor <- factor(datos_e$visitor,
                          levels = c('true', 'false'),
                          labels = c('TRUE', 'FALSE'))
```

Como se observa a continuación, el 65 % de las observaciones son falsas, es decir, no son visitantes, mientras que el 34.2 % corresponen a visitantes.

```
##
##  TRUE FALSE
##  34.2  65.8
```

Posteriormente, se procedió a divir el dataset datos_e en dos: el 80 % de las filas se utilizaron para entrenamiento, mientras que el 20 % restante se utilizó para la validación del modelo. En cada uno de los datasets creados se exluyó la columna 'visitor'.

```
nfilas <- nrow(datos_e) * .80
set.seed(123)
index <- sample(1:nrow(datos_e), nfilas)
datos_e_train <- datos_e[index, -5]
datos_e_test <- datos_e[-index, -5]
```

Para validar los resultados, se crearon dos datasets complementarios que contienen solamente la columna 'visitor'. Estos datasets serán utilizados para corroborar la efectividad del modelo.

```
datos_e_train_labels <- datos_e[index, 5]
datos_e_test_labels <- datos_e[-index, 5]
```

En este paso se genera el modelo con el 80 % de las filas destinadas al entrenamiento.

```
datos_e_model <- C5.0(datos_e_train, datos_e_train_labels)
```

Una vez que se generó el modelo, se ejecuta sobre el 20 % de las filas reservabas para la prueba.

```
datos_e_predict <- predict(datos_e_model, datos_e_test)
```

Par obtener el mejor resultado, el modelo se replica una cantidad de veces determinada. En esta ocasión se eligió replicarlo 50 veces.

```
datos_e_boost50_model <- C5.0(datos_e_train, datos_e_train_labels, trials = 50)
datos_e_boost50_model
```

```
##
## Call:
## C5.0.default(x = datos_e_train, y = datos_e_train_labels, trials = 50)
##
## Classification Tree
## Number of samples: 199644
```

```
## Number of predictors: 5
##
## Number of boosting iterations: 50 requested;  27 used due to early stopping
## Average tree size: 14.9
##
## Non-standard options: attempt to group attributes
```

```
datos_e_boost_pred50 <- predict(datos_e_boost50_model, datos_e_test)
```

Conclusiones

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  49912
##
##
##              | predicción
##       actual |      TRUE |     FALSE | Row Total |
## -------------|-----------|-----------|-----------|
##         TRUE |     15627 |      1223 |     16850 |
##              |     0.313 |     0.025 |           |
## -------------|-----------|-----------|-----------|
##        FALSE |       600 |     32462 |     33062 |
##              |     0.012 |     0.650 |           |
## -------------|-----------|-----------|-----------|
## Column Total |     16227 |     33685 |     49912 |
## -------------|-----------|-----------|-----------|
##
##
```

Como se observa en la tabla anterior, los falsos positivos representan tan solo el 2.5 % de las observaciones, mienstras que los falsos confirmados corresponden al 65 % de las observaciones. De la misma manera, los falsos negativos equivalen el 1.2 % y los falsos confirmados al 31.3 %. Estas proporciones se acercan lo suficiente a la distribución real de la tabla, como se presentó en este mismo documento. Así, se puede concluir que el modelo ha sido exitoso y que puede replicarse en el dataset datos_v para determinar cuáles usuarios son visitantes y cuáles no.

Para este proceso, se ejecutó la siguiete siguiente línea de código.

```
datos_v_boost_pred50 <- predict(datos_e_boost50_model, datos_v)
```

Una vez hecho este proceso, se agregaron los resultados al dataset datos_v utilizando la función mutate.

```
datos_v <- datos_v %>%
  mutate(prediccion = datos_v_boost_pred50)
```