UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Big Data Management and Analytics
Session: Introduction to Python

Lecturers: Pedro González and Alberto Gutiérrez

February 5, 2025

Uno de los KPI's de un call center de atención al cliente es el ratio de OCR (one call resolution), es decir, el ratio de consultas o incidencias de un cliente que se resuelven en la primera llamada sin que este tenga que volver a establecer contacto con el call center. Obviamente, a un call center le interesa mejorar su ratio OCR. ¿Es posible caracterizar los datos de las llamadas integrando todos los datos conocidos (cliente, operador que atiende, transcripción de la conversación, etc) para identificar aquellas llamadas que probablemente generen nuevas llamadas?. Esto es de hecho un proyecto real en el que se inspira este ejercicio.

El fichero calls_without_target.csv que se adjunta contiene un extracto muy simplificado de datos de llamadas realizadas a un call center. Se han eliminado todas las variables dejando únicamente algunos identificadores recodificados y el timestamp de la misma ya que son los que nos interesan para este ejercicio. Las variables suministradas son las siguientes:

1. call_id. Identificador único de llamada.
2. customer_id. Identificador del cliente que realiza la llamada.
3. call_ts. Timestamp de la llamada en formato "YYYY/MM/DD HH:mm:SS"

(8 puntos) El objetivo del ejercicio es identificar las llamadas que son "re-calls", es decir, aquellas llamadas de clientes que se repiten (mismo cliente) en una ventana de tiempo inferior a, por ejemplo, 48 horas. Igualmente, también queremos identificar las llamadas que son precursoras de recall, es decir, las llamadas que originan llamadas sucesivas de clientes en la ventana de tiempo siguiente de 48 horas. Se pide generar un dataset con el mismo número de filas (llamadas) que el original , incluyendo para cada llamada los siguientes datos:

1. **call_id**: Identificador de llamada para verificar los resultados.
2. **is_precursor**: Indica si hay una llamada dentro de la ventana de las próximas 48 horas contadas a partir del timestamp de la llamada, del mismo cliente. Valores ("Y/N").

3. **is_recall**: Indica si la llamada es una recall. Indica si existe una llamada del mismo cliente en la ventana de 48 horas anteriores a la llamada en cuestión contadas desde su timestamp. Valores ("Y/N").

4. **precursor_call_id**. Si la llamada se marca como recall, será el id_call de la llamada precursora, es decir, la llamada del mismo cliente inmediatamente anterior dentro de la ventana de 48 horas anteriores contadas desde el timestamp de la llamada en curso. En otro caso, valor nulo.

5. **hours_from_first_call**. Tiempo transcurrido en horas desde la llamada precursora inmediatamente anterior hasta la llamada en curso. Nulo si esa llamada no es un recall.

Indicaciones. Se sugiere un algoritmo simple, pero son libres de aportar otras soluciones . Por ejemplo, menos algorítmicas y usando pandas como soporte de datos.

- Puede usar esta función para verificar si dos timestamps están dentro de la ventana de tiempo

```
from datetime import datetime

# True if t2-t1 is smaller than 2 days
def inRecallTimeWindow(t1,t2):
    timeWindowDays = 2.0
    dt1 = datetime.strptime(t1," %Y/ %m/ %d- %H: %M: %S")
    dt2 = datetime.strptime(t2," %Y/ %m/ %d- %H: %M: %S")
    delta = dt2 - dt1
    days = int(delta.total_seconds()/86400.0)

    return(days<timeWindowDays)
```

(2 puntos) Proponga un pseudo-código o solución (mononodo) para casos en que tengamos millones de llamadas y no sea viable almacenar todos los datos en memoria