Equipo Jerry

Reporte Etapa 3: Data Preparation

Descripción de los datos

Selección de datos

- Decidir sobre los datos que se utilizarán para el análisis

- Los criterios deben considerar:
  - La relevancia de los datos en el objetivo del proyecto
  - La calidad de los datos
  - Las restricciones técnicas como los límites en el volúmen de los datos o los tipos de datos

Base lógica para la inclusión / exclusión de datos

Para esta parte, comenzamos a revisar los datos que nos podrían funcionar para llegar al objetivo que propusimos durante la primera etapa.

- Decidir qué datos se van a usar para el análisis
- Lista de los datos que se usarán con las razones por las que se tomó esta decisión

  Los datos que vamos a usar para el análisis de acuerdo a nuestro objetivo y que nos parecen relevantes son:

  Datos principales:
  - PHONE_ID
    - Este dato es el id del usuario que nos permitirá saber el comportamiento de cada uno cuando estemos tratando de analizar hacia a dónde van y vienen.
  - lat y lon
    - La latitud y la longitud son unidades de distancia astronómica, nos van a servir para obtener las coordenadas y por consiguiente las ubicaciones de las antenas que queremos analizar.
  - timestamp
    - Es la hora en la que el usuario está conectado en la antena en ese preciso momento, donde nos ayudarán los datos para obtener el tiempo del viaje que realizó un usuario.

  Datos a generar:
  - Comuna
    - Una vez obtenidas todas las comunas, las asociamos a las antenas para tener las ubicaciones de cada una.
  - Tiempo de viaje

    - Con el timestamp que se nos proporciona de cada usuario podremos calcular el tiempo de viaje que llegó a realizar un usuario de una antena a otra.
  - Distancia de viaje
    - Es la distancia que le toma al usuario para llegar de una antena a otra, donde puede tener varias conexiones de antena en un solo viaje.

- Lista de los datos que serán excluidos con las razones por las que se tomó esta decisión
  Del dataset original se utilizarán las columnas de PHONE_ID, lon y lat, timestamp, así como las columnas generadas con estas como la comuna, el tiempo de viaje y la distancia de viaje.
  Las demás columnas del dataset original se descartan debido a que agregan cierto nivel de complejidad dentro del análisis y al descartarla se simplifica de gran manera y hace más sencillo el proceso de limpieza de datos, así como el procesamiento del modelo.

- Recolecta datos adicionales según sea apropiado (internos y externos)

  Los datos adicionales que vamos a requerir para nuestro análisis de acuerdo a nuestro objetivo son los siguientes:

  Datos externos:
  - Top 5 Universidades de Chile
    - Para lograr el objetivo de nuestro proyecto, decidimos utilizar el top 5 de las mejores Universidades de Santiago de Chile, ya que hay demasiadas universidades y no iba a ser posible poder analizar todas. Las 5 Universidades que encontramos fueron: Pontificia Universidad Católica de Chile, Universidad de Chile, Universidad Diego Portales, Universidad de Santiago de Chile, Universidad del Desarrollo. Se eligieron estas universidades debido a que son las que cuentan con mayor puesto dentro de rankings como “THE World University Rankings - Times Higher Education” y “QS World University Rankings ”
  - Ubicaciones geográficas de las Universidades
    - La ubicación de las universidades las ocupamos para poder localizar las antenas que se encuentran en esas universidades para la realización del análisis. La longitud y latitud de estas se obtuvo vía Google Maps.
  - Datos de OpenStreetMap
    - Es la página que nos ayudará a sacar los datos más específicamente de las Universidades de Santiago de Chile.

Consideramos que para procesar y realizar la limpieza de nuestros datos es necesario reducirlos, ya que al tratarse de una gran cantidad de registros, esto alentará el tratamiento del dataset ya que genera más carga al CPU por leer una gran cantidad de datos en cada iteración

Limpieza de datos

- Elevar la calidad de los datos al nivel requerido por las técnicas de análisis seleccionadas

- Esto puede incluir:
  - La selección de subconjuntos de datos limpios
  - Insertar valores por defecto adecuados
  - Estimación por modelado de datos faltantes

Reporte de limpieza de datos

Obtener ubicaciones de las antenas en base a las coordenadas usando Geopy.
Al tener datos inconsistentes con los IDs de las antenas y las ubicaciones registradas, consideramos corregir el ruido del dataset generando las ubicaciones en base a las coordenadas, auxiliandonos de Geopy.
En esta implementación no se hace uso del atributo Phone_ID ni de Timestamp por lo que se eliminan ambas columnas.

- Comenzamos nuestro proceso convirtiendo los atributos de latitud y longitud a datos tipo string, (esto nos permite realizar la lectura de los datos con Geopy), posteriormente generamos una columna de “coordenadas”.en la que se junta la latitud y longitud.
- Establecemos userAgent de Nominatim para realizar peticiones al servidor, de esta manera generamos otra columna en la que se agrega la dirección obtenida mediante las coordenadas. Esta columna la generamos como objeto para separar las comunas con un listado.
- Continuamos definiendo una función que nos permita realiza la busqueda de la comuna dentro de la dirección y nos da como resultado el nombre de la comuna (en caso de que coincida). A partir del uso de nuestra función para encontrar las comunas, generamos una columna extra en la que se almacenan los resultados de la función.
- Durante nuestro proceso, observamos que tenemos 53 comunas en lugar de 52, por lo que consideramos que Open StreetMao no pudo localizar o asignar de forma correcta una comuna. Al tener una comuna no identificada, realizamos la asignación manual, utilizando

datos de Microsoft Bing Maps, tenemos: Angostura Paine Región Metropolitana, Chile.
- Para terminar, eliminamos las columnas que ya no son necesarias para generar nuestro dataset, por lo que solo mantendremos el ID de la atena, las coordenadas y el nombre de la comuna.

Obtener las trazas de los viajes (distancia, velocidad y tiempo)
Se considera una traza cuando un usuario hace un salto de una antena a otra. Dentro de este proceso solo se hizo uso de las columnas de latitud y longitud, el ID del usuario, así como el timestamp. Se utilizó el siguiente algoritmo:

1. Para la obtención de las trazas inicialmente se ordenaron los datos para poder agruparlos datos por usuario, al hacer eso nos aseguramos que no se mezclan trazas de un usuario con otro.
2. Se hizo uso de la fórmula de Haversine para el cálculo de la distancia de la traza, esta fórmula toma la longitud y latitud de dos puntos y utilizando el radio de la tierra es capaz de calcular la distancia entre estos dos puntos.
3. Utilizando el timestamp, se calculó el tiempo en horas que tomó la traza.
4. Utilizado la distancia y el tiempo que se tomó la traza se calcula la velocidad de la traza.
5. Después se coloca 0 en los primeros registros de cada usuario debido a que ese el punto de partida de cada usuario
6. Finalmente se exporta ese dataset a un csv

Limpieza de usuarios que no están en movimiento
En un inicio el plan era recorrer el dataset con un for del tamaño del dataset original mediante el siguiente algoritmo

1. Por cada usuario
    a. Crear un nuevo registro en un nuevo dataset y añadir la información
    b. Si el siguiente registro es del mismo usuario, ir sumando la cantidad del movimiento que tiene en una variable
    c. Si el siguiente registro es de otro usuario, cortarlo y pasar al siguiente
    d. Repetir hasta el fin del dataset

En este punto vimos que el procesamiento era demasiado lento y que probablemente tardaría más de 2 días en correr por unas estimaciones rápidas que realizamos. La nueva solución estuvo en agrupar a todos los usuarios y sumar las velocidades de cada uno usando Pandas directamente, así obtuvimos el total y ahora el dataset era mucho más pequeño comparado al original.

La siguiente operación fue crear una columna lógica la cual resulta verdadera cuando la suma de la distancia es 0, lo cual indica que no se mueve, por lo cual en caso de que sea verdadero, se borraron los registros correspondientes. Hecho esto, con el fin de optimizar el dataset se mantuvieron únicamente las columnas

de Phone_ID y borrador lógico y por otro lado, se cargó el dataset con distancia, velocidad y tiempo para juntar ambos y hacer el drop de los usuarios que no tienen movimiento, después se eliminó la columna de borrado y finalmente se guardó el dataset.

Quitar las velocidades mayores a 150 km/h

Al principio de esta parte, creímos que tomando todas las velocidades inicialmente calculadas y luego eliminando todos los resultados que fueran mayores a 150 km/h, nos ayudaría por completo en la limpieza de datos.

Lamentablemente no fue así porque había que considerar que si eliminábamos un registro, podríamos dejar el siguiente inutilizable ya que existía la posibilidad de tener otro 'teletransporte' entre los nuevos puntos del viaje del usuario, por lo que ahora tuvimos que realizar un algoritmo que cumpliera los siguientes puntos:

1. Si la velocidad es mayor a 150 km/h
    a. Eliminar el renglón de información que encontramos
    b. Recalcular el tiempo del siguiente punto y el anterior
    c. Recalcular la distancia de ambos puntos con la fórmula de Haversine
    d. Obtener la velocidad nueva para re-evaluar.
    e. Repetir hasta el final del dataset.

Pero nos encontramos con otro problema, nuestro algoritmo iba extremadamente lento.

La principal causa de ello era que estábamos metiendo el dataset completo, esto le genera más carga al CPU por leer bastantes datos en cada iteración, aparte el script sólo corría sobre un núcleo.

La estimación de que terminara de correr iba hasta los 40 días, lo que sobrepasa nuestro plan de trabajo.

Para mejorar el rendimiento, decidimos separar el dataset en 5 partes diferentes, sin cortar abruptamente al usuario a un nuevo dataset, o el algoritmo planeado para quitar a las personas con velocidades mayores ya no nos funcionaría.

Para cortarlo de esta manera buscamos el número de usuarios únicos y lo dividimos entre cinco, de esta manera teníamos el número máximo que tomaríamos para cada dataset.

El plan inicial estuvo en armar un dataset con appends:

1. El primer dato siempre lo añadimos
2. Verificamos si ya se alcanzó el límite de usuarios por parte de dataset
3. En caso de que no, añadir el siguiente dato si coincide con el anterior
4. Si el dato no coincide con el anterior, sumar el cambio de usuario

5. Repetir hasta llegar al límite de usuarios y hacer un break

Nuevo problema: al estar creando un dataset completamente nuevo y de forma muy manual, estábamos haciendo algo muy ineficiente y que según mis estimaciones, estaríamos tardando alrededor de 60 horas.

La nueva solución estuvo en no hacer absolutamente nada dentro del algoritmo (comentar/deshabilitar las acciones que queríamos realizar), sólo contar hasta qué rango del dataset cumplíamos el límite de usuarios e ir tomando nota, eso redujo el tiempo a 20 minutos, y 5 minutos más usando la función iloc (para cortar) datos con Pandas.

Salida del programa:

```
. . .
Progreso:99.71738576289219%
Progreso:99.8204568376021%
Progreso:99.91724759593184%
Alcance el max de usuarios en 49618000
```

Corrimos el programa varias veces hasta el número donde se había detenido la ejecución y conseguimos los siguientes resultados:

```
Rango 1:   [        0 :  9930737 ]
Rango 2:   [  9930738 : 19865945 ]
Rango 3:   [ 19865946 : 29791098 ]
Rango 4:   [ 29791099 : 39704304 ]
Rango 5:   [ 39704305 : 49618131 ]
```

Corriendo la prueba de una sola parte, nos dimos cuenta de que numpy no estaba ayudando mucho como tal, leímos que Pandas estaba construído sobre numpy entonces traspasamos el algoritmo de numpy a Pandas por completo.
Ahora sí, ya era hora, abrimos 3 instancias de Python distintas en Jupyter Notebook de Oracle Cloud para usar 3 de los 4 núcleos disponibles en ‘paralelo’ y dos computadoras más, la de Manolo y la de Eduardo.

En rendimiento, el servidor de Oracle estaba más cercano a la laptop de Manolo y la laptop de Eduardo era más rápida haciendo el trabajo ya que disponía de un procesador más potente.

Esto fue lo que tardamos por núcleo / computadora utilizada:

- Oracle Instance 1:
  - 1 día, 3 horas, 1 minuto.
- Oracle Instance 2:
  - 1 día, 3 horas, 3 minutos.

- Oracle Instance 3:
    - 1 día, 2 horas, 59 minutos.
- Laptop Manolo:
    - 21 horas, 37 minutos.
- Laptop Eduardo:
    - 21 horas, 49 minutos.

Podría decirse que nos ahorramos como 5 días en el proceso. Ocurrió un incidente en donde Windows Update interrumpió por completo lo que se estaba procesando en la laptop de Eduardo, pero “nos recuperamos” por la velocidad de su procesador.

Una vez que terminamos con eso, volvimos a juntar todo el dataset procesado, donde obtuvimos un resultado de 294666 de registros eliminados en el dataset correspondientes a las anomalías de los usuarios entre las conexiones de las antenas, dándonos un resultado final de 49323466 registros.

Ahora sólo nos queda reemplazar los IDs de las antenas por las coordenadas, más adelante explicaremos la razón de recuperar el dato que usamos para obtener la ubicación de las antenas.

Reemplazar los IDs de las antenas por las coordenadas
El dataset presentaba datos inconsistentes con los IDs de las antenas y las ubicaciones registradas, por lo que generaba ruido al tener identificadores con diferente latitud y longitud; generar las ubicaciones a partir de las coordenadas nos permite llevar un mejor orden y claridad en el dataset.

Finalización de limpieza
Al completar todos los pasos de limpieza anteriores, revisamos que todo estuviera en orden y unimos todo el set de datos usando los distintos archivos generados y procesados a través de un merge con Pandas.

Comparando las características del dataset original y el final, el número de registros se redujo en 11,865,891 renglones, por lo que el proceso de limpieza sí quitó una buena cantidad de registros que para nuestro análisis y modelo nos iban a estar metiendo mucho ruido.

Este nuevo dataset lo guardamos en la misma carpeta del dataset original pero con un nombre nuevo para saber que es el que está limpio.

Como medida preventiva, en caso de que sea necesario regresar y realizar correcciones a las limpiezas, tenemos los archivos generados todavía para evitarnos mucho tiempo de espera en procesamiento.