

# **Reporte Etapa 3: Data Preparation**

<b>Abstract</b>	<b>3</b>
<b>Diagrama de archivos generados</b>	<b>4</b>
<b>Selección de datos</b>	<b>5</b>
Base lógica para la inclusión / exclusión de datos	5
Datos excluidos	5
Datos adicionales (internos y externos)	5
Técnicas de muestreo	7
<b>Construcción de datos</b>	<b>7</b>
Atributos derivados	7
Registros generados	7
<b>Integración de datos</b>	<b>7</b>
Datos integrados	7
<b>Formateo de datos</b>	<b>7</b>
Datos reformateados	7
<b>Limpieza y preparación de datos</b>	<b>8</b>
Reporte de limpieza de datos	8
Construcción de datos 1: Obtener ubicaciones de las antenas en base a las coordenadas usando Geopy.	8
Construcción de datos 2: Obtener las trazas de los viajes (distancia, velocidad y tiempo)	8
Limpieza de datos 1: Usuarios que no están en movimiento	9
Limpieza de datos 2: Descartar velocidades mayores a 150 km/h	10
Limpieza de datos 3: Eliminando usuarios que tienen recorren más de 100 kilómetros en el día	12
Construcción de datos 3 y limpieza 4: Reemplazar los IDs de las antenas por las coordenadas	13
Construcción de datos 4: Añadiendo las comunas al dataset	13
Construcción de datos 5: Generación de matriz de viajes	13
Construcción de datos 6: Creación de zonas universitarias	16
Construcción de datos 7: Dataset final para modelado	17
<b>Finalización de limpieza</b>	<b>19</b>
Descripción del dataset.	20

## **Abstract**

En este documento se realizaron distintos trabajos de limpieza y construcción de datos para obtener un dataset final para la preparación de los modelos.

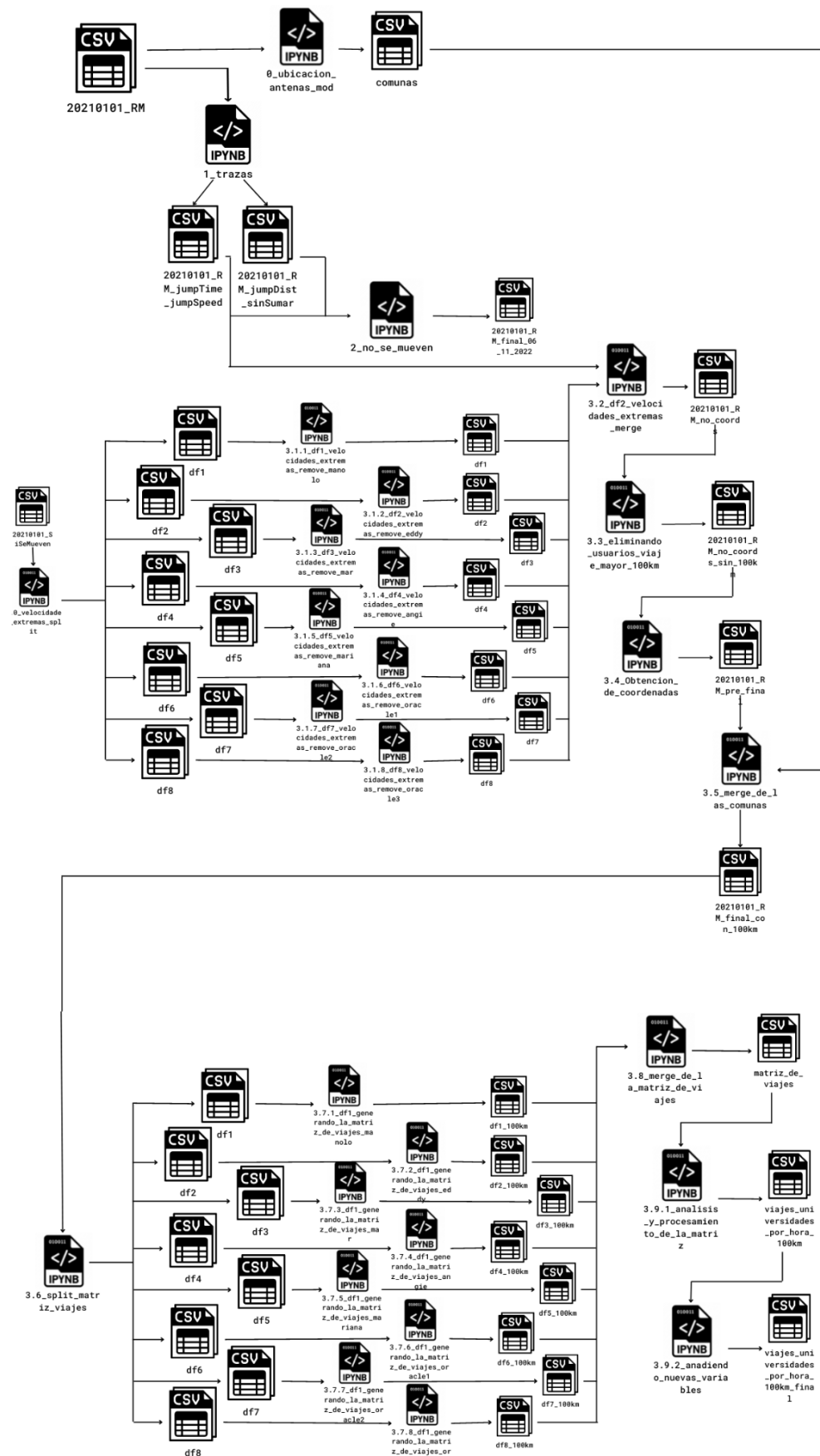
Comenzamos con un dataset de 49,618,132 datos, al analizarlos decidimos realizar operaciones de construcción y limpieza. Obtuvimos la ubicación de las antenas para saber en qué comuna se ubicaban, después nos dedicamos a obtener las trazas de los viajes mediante cálculos realizados entre los registros de los usuarios.

Después, limpiamos los usuarios sin movimiento y teletransportaciones que daban velocidades inexplicables. Reemplazamos los IDs de las antenas con una concatenación de las coordenadas para obtener valores únicos. En este punto ya teníamos 37,732,972 datos.

Integramos la información de las comunas y después pasamos a hacer la matriz de viajes donde a partir de varias condiciones generamos un dataset. Ahora sólo quedaban 1,203,010 datos. Con la información de la matriz, y las sedes universitarias creamos el dataset final con datos de zonas universitarias a 1km de radio para la parte de modelado. Así es como nos quedó un registro de la suma de los viajes por hora, universidad y comuna.

Como variables de análisis adicionales decidimos integrar datos como si la universidad es pública, privada, tiene residencias, si el viaje era dentro o fuera de la zona, el número de carreras y el tamaño en metros cuadrados. Así es como al final quedamos con 1828 datos para el modelo.

## Diagrama de archivos generados



## Selección de datos

### Base lógica para la inclusión / exclusión de datos

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.

### Datos excluidos

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.

### Datos adicionales (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 dividido por sedes
  - 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 ”
- Sedes de cada una de las universidades
  - Al tener facultades esparcidas en Santiago de Chile, consultamos todas las sedes de las universidades para hacer más específico nuestro análisis y obtener hallazgos más concretos.  
Las sedes se dividieron de la siguiente manera:

- |   |  |
|---|--|
| 1. Uchile Campus Andrés Bello                   | 17. USACH Facultad tecnológica                   |
| 2. Uchile Campus Beauchef                       | 18. USACH Facultad de arquitectura               |
| 3. Uchile North Campus (Dra. Eloisa Diaz)       | 19. USACH Facultad de ciencias médicas           |
| 4. Uchile Juan Gómez Millas,                    | 20. UDD Sede Santiago                            |
| 5. Uchile Campus Sur Uchile                     | 21. UDP Facultad de economía y empresas          |
| 6. Uchile Casa central                          | 22. UDP Facultad de derecho,                     |
| 7. UC Casa central                              | 23. UDP Facultad de arte, Arquitectura y diseño  |
| 8. UC El comendador                             | 24. UDP Facultad de ciencias sociales e historia |
| 9. UC Campus Oriente                            | 25. UDP Facultad de salud y odontología          |
| 10. UC San Joaquín                              | 26. UDP Facultad de psicología                   |
| 11. USACH Facultad de administración y economía | 27. UDP Facultad de comunicación y letras        |
| 12. USACH Facultad de ciencia                   | 28. UDP Facultad de ingeniería y ciencias        |
| 13. USACH Facultad de derecho                   | 29. UDP Facultad de medicina                     |
| 14. USACH Facultad de humanidades               |  |
| 15. USACH Facultad de ingeniería                |  |
| 16. USACH Facultad de química y biología        |  |

- 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.

### Técnicas de muestreo

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

## Construcción de datos

### Atributos derivados

Como se mencionó anteriormente, a partir de los datos originales, se obtuvieron datos derivados como lo son:

- ★ Cálculo del tiempo que tomó el viaje
- ★ Cálculo de la distancia entre una antena y otra
- ★ Cálculo de la velocidad estimada del traslado
- ★ Definiendo comuna mediante coordenadas a partir de la latitud y longitud del dataset.
- ★ Zonas universitarias a partir de agrupación de ciertas antenas.

### Registros generados

No generamos registros externos ya que no lo consideramos necesario.

## Integración de datos

### Datos integrados

- ❖ Durante la limpieza de velocidades extremas se unió una columna lógica con el dataset original para hacer la relación de los usuarios que no tienen movimiento.
- ❖ Para la generación de los viajes a las zonas universitarias, agregamos información como si la universidad es pública o privada, la sede universitaria, si la sede tiene residencias, el tamaño de la universidad y el número de carreras por sede.

## Formateo de datos

### Datos reformateados

- ❖ Se agruparon los registros a partir del user\_id para definir todos los viajes obtenidos por el mismo usuario.
- ❖ En los primeros registros de cada usuario se define en 0 la velocidad para identificar el punto inicial del usuario y generar las trazas correctas de sus viajes.
- ❖ Para el proceso de velocidades extremas, revisamos los datos siguientes para corregir correctamente la velocidad, con esto re calculamos el tiempo de traza a traza, la diferencia de distancia y al final la velocidad correcta.

## Limpieza y preparación de datos

### Lista de supuestos

1. No se toman en cuenta los usuarios que no se mueven.
2. A partir de las trazas, si la velocidad del usuario es mayor a 150 km/h se descarta el registro.
3. Se eliminarán usuarios los cuales hayan recorrido más de 100 km o se alcanzan 120 minutos y se asumirán que son taxistas. (estos no aportan información sobre la movilidad).
4. Si el usuario permanece en un estado estacionario por más de 20 minutos, se considera que acabó el viaje.
5. Si la distancia es mayor a 25 km, se corta el viaje y se inicia uno nuevo.
6. Se considera que si una antena está a 1km de la universidad, esta antena está dentro de una zona universitaria

### Reporte de limpieza de datos

#### **Construcción de datos 1: 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, auxiliándose de Geopy.

En este proceso no se hace uso del atributo Phone\_ID ni de Timestamp por lo que se eliminan ambas columnas.

1. 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.
2. 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.
3. Continuamos definiendo una función que nos permita realizar la búsqueda 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.
4. Durante nuestro proceso, observamos que obtuvimos 53 comunas en lugar de 52, por lo que consideramos que OpenStreetMap 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.
5. 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.



Todo este proceso puede encontrarse en

- 0\_ubicacion\_antenas\_mod.ipynb

### **Construcción de datos 2: 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

Todo este proceso puede encontrarse en:

- 1\_trazas.ipynb

### **Limpieza de datos 1: Usuarios que no están en movimiento**

Con el fin de optimizar el dataset se mantuvieron únicamente las columnas de Phone\_ID y borrador lógico, se cargó el dataset que contiene la distancia, velocidad y tiempo para juntar ambos y eliminar los usuarios que no registran movimiento.

En un inicio el plan era recorrer el dataset con un for del tamaño del dataset original mediante el siguiente algoritmo

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

En este punto vimos que el procesamiento era demasiado lento y sobrepasaría nuestro tiempo estimado de trabajo, por lo que consideramos una solución alternativa.

La nueva solución consistió en:

1. 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.
2. Generamos una columna lógica la cual resulta verdadera cuando la suma de la distancia es 0, lo que indica que existe movimiento.
3. En caso de que sea verdadero, se borraron los registros correspondientes.

Hecho esto, se eliminó la columna con el indicador lógico y finalmente se guardó el dataset.

Todo este proceso puede encontrarse en:

- `2_no_se_mueven.ipynb`

### **Limpieza de datos 2: Descartar 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 terminará el procesamiento sobrepasaba 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 funcionará.

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 construido 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 personal 1:
  - 21 horas, 37 minutos.
- Laptop personal 2:
  - 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 el proceso, se juntó todo el dataset, donde obtuvimos un resultado de 294666 de registros eliminados, correspondientes a las anomalías de los usuarios entre las conexiones de las antenas, dándonos un total de 49323466 registros finales.

Todo este proceso puede encontrarse en:

- 3.0\_velocidades\_extremas\_split.ipynb
- 3.1.1\_df1... - 3.1.8\_df8\_velocidades\_extremas\_remove\_manolo.ipynb
- 3.2\_df2\_velocidades\_extremas\_merge.ipynb

### **Limpieza de datos 3: Eliminando usuarios que recorren más de 100 kilómetros en el día**

Continuando con la limpieza de datos, ya que tenemos los registros de velocidades y trazas limpios, vamos a quitar a las personas que hayan recorrido más de 100 kilómetros en el día, ya que se puede tratar de algún taxista y nos estaría dando datos que realmente no son viajes de una persona individual.

Para ello sólo agrupamos a los usuarios y sumamos su distancia recorrida total, después añadimos una columna de borrado lógico e hicimos un merge con el dataset para “seleccionar” a todos los usuarios que recorrieron más de 100 kilómetros de distancia y luego eliminarlos.

En este proceso eliminamos a 887,159 usuarios, siendo en total 2,577,343 registros.

Al final guardamos una nueva copia por si teníamos que regresarnos y nuestro proceso estaba mal. Spoiler, sí.

**Nota:**

Tras realizar la generación de la matriz de viajes más adelante, nos regresamos a esta parte para omitir esto y realizar manualmente la verificación para no quitar erróneamente a algunos usuarios que legítimamente no viajaron 100km de forma constante.

Todo este proceso puede encontrarse en:

- 3.3\_eliminando\_usuarios\_viaje\_mayor\_100km.ipynb

**Construcción de datos 3 y limpieza 4: 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. Para esto, se generó una nueva columna donde se concatena las columnas de longitud y latitud. Y la columna de bts\_id se descarto.

Todo este proceso puede encontrarse en:

- 3.4\_Obtencion\_de\_coordenadas.ipynb

**Construcción de datos 4: Añadiendo las comunas al dataset**

Para este proceso, nos apoyamos del trabajo que hicimos para obtener la ubicación de las antenas y del trabajo anterior de reemplazar los IDs de las antenas por coordenadas.

En el dataset generado para obtener la ubicación de las antenas, no teníamos la concatenación de coordenadas, no fue gran problema hacerlo, sólo quitamos del proceso el eliminar esa columna y continuamos.

Ahora sí, para poner las comunas en el dataset, hicimos un merge en las coordenadas concatenadas pero sucedió un incidente. Al revisar los resultados, obtuvimos un número de datos nulos gigantesco, pues, el proceso que estábamos realizando en el primer documento para las antenas únicas con Pandas, se estaban obteniendo valores flotantes extraños y entonces así no coincidía el string de la concatenación.

Nuevamente nos regresamos y lo solucionamos eliminando datos duplicados después de realizar la columna concatenada. Verificamos que nos dieran exactamente las mismas comunas y fue así. De esta manera ya ahora sí teníamos los datos iguales y el merge ya se hacía sin generar algún dato nulo. :)

Cuando hicimos el merge, ya obtuvimos todas las comunas pertenecientes a la cada antena en todo el dataset y decidimos continuar por fin con la generación de la matriz de viajes.

Todo este proceso puede encontrarse en:

- 3.5\_merge\_de\_las\_comunas.ipynb

### **Contrucción de datos 5: Generación de matriz de viajes**

Ya teniendo el dataset limpio y con las columnas del usuario, hora, la antena y la comuna correspondiente, estábamos listos para comenzar a hacer la matriz de viajes.

Esta parte fue un poco tardada ya que debíamos cumplir condiciones específicas para que consideremos una serie de trazas como un viaje, pues las condiciones eran:

1. Si el viaje es mayor a 120 minutos...
  - a. No considerarlo como viaje
  - b. Saltar completamente a ese usuario
2. Si el usuario está en el mismo lugar por más de 20 minutos...
  - a. Terminar el viaje
  - b. Registrar uno nuevo viaje
3. Si la distancia de un viaje es mayor a 25 kilómetros...
  - a. Terminar el viaje
  - b. Registrar uno nuevo viaje
4. Si la traza ya no es del mismo usuario...
  - a. Terminar el viaje
  - b. Registrar el viaje del siguiente usuario

Al principio creamos unos algoritmos que no cumplían bien la función, de hecho eran algoritmos más complicados de lo que en realidad debíamos de hacer.

Nos tomamos un momento para pensar y realizar un algoritmo con datos imaginarios en el pizarrón, obtuvimos algo similar al algoritmo final, consideramos que fue una excelente guía y apoyo.

Ya que probamos e hicimos debug, ya estábamos listos para correr el algoritmo, pero, sorpresa, todavía teníamos una gran cantidad de datos, entonces pensamos en hacer un split nuevamente de los datos para nuevamente procesar en varias computadoras y núcleos a la vez.

Esta ocasión, aplicamos la misma técnica de contar un cierto número de usuarios entre el número de computadoras disponibles, esta vez usamos ocho.

Ahora los rangos con el dataset limpio quedaron de esta manera:

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

```
Rango 1: [      0 : 4722472 ]
Rango 2: [ 4722473 : 9442729 ]
Rango 3: [ 9442730 : 14149972 ]
Rango 4: [ 14149973 : 18881592 ]
Rango 5: [ 18881593 : 23595450 ]
Rango 6: [ 23595451 : 28307747 ]
Rango 7: [ 28307748 : 33022348 ]
Rango 8: [ 33022349 : 37732971 ]
```

Ya que obtuvimos los nuevos datasets por separado, creamos archivos dedicados para cada computadora. Del notebook que teníamos en un inicio, transferimos el algoritmo a estos y nos pusimos de acuerdo para que cada quien tomara una de las partes a procesar.

Esto fue lo que tardamos por núcleo / computadora utilizada:

- Laptop personal 1:
  - 1 hora, 28 minutos.
- Laptop personal 2:
  - 1 hora, 26 minutos.
- Laptop personal 3:
  - 4 horas, 26 minutos
- Laptop personal 4:
  - 5 horas, 12 minutos.
- Laptop personal 5:
  - 1 hora, 31 minutos.
- Oracle Instance 1:
  - 1 hora, 31 minutos.
- Oracle Instance 2:
  - 1 hora, 32 minutos.
- Oracle Instance 3:
  - 1 horas, 31 minutos.

Ya que la carga de trabajo estaba más distribuída y teníamos que procesar menos datos esta vez, los tiempos de espera fueron mucho menores y pudimos continuar con el merge de todas estas partes rápidamente, nuevamente ahorrando tiempo.

Una vez que terminamos el proceso, juntamos todos los datasets que ya se procesaron anteriormente para generar la matriz de viajes.

De repente nos percatamos de que existían viajes sin distancia y duración, por lo que revisamos el algoritmo y encontramos un número que regresaba e intentaba registrar un viaje nuevamente que no tenía nada, este problema fue arreglado rápidamente.

Luego de re procesar la información, salió otro problema donde no se estaba considerando correctamente el tiempo del viaje, entonces nos regresamos a arreglar ese detalle cambiando el valor de minutos a horas.

Para asegurarnos de que todo estuviera ya bien, hicimos bastante debugging y nos dimos cuenta que también estábamos registrando tiempo inactivo como viajes, solucionamos eso, volvimos a probar, detectamos otro problema donde no se consideraba correctamente si se trataba de un taxi la persona, pues intentábamos saltar al usuario pero el proceso no se cumplía correctamente.

Cada vez fuimos metiendo un poco más de datos y hacíamos revisión completamente manual de las salidas que estaban hasta que observamos que el algoritmo ya estaba funcionando como lo deseábamos y por fin ya volvimos a correr todo. Esto nos causó un retraso de dos días porque no fácilmente notábamos que algo estaba mal.

Después de revisar que los datasets estuvieran ya completamente correctos. los juntamos en orden para que no se revolvieran los datos y que pudiéramos generar un solo archivo.

Todo este proceso puede encontrarse en:

- 3.7.1\_df1\_generando\_la\_matriz\_de\_viajes\_manolo.ipynb
- 3.7.2\_df2\_generando\_la\_matriz\_de\_viajes\_eddy.ipynb
- 3.7.3\_df3\_generando\_la\_matriz\_de\_viajes\_mar.ipynb
- 3.7.4\_df4\_generando\_la\_matriz\_de\_viajes\_angie.ipynb
- 3.7.5\_df5\_generando\_la\_matriz\_de\_viajes\_mariana.ipynb
- 3.7.6\_df6\_generando\_la\_matriz\_de\_viajes\_oracle.ipynb
- 3.7.7\_df7\_generando\_la\_matriz\_de\_viajes\_oracle2.ipynb
- 3.7.8\_df8\_generando\_la\_matriz\_de\_viajes\_oracle3.ipynb
- 3.8\_merge\_de\_la\_matriz\_de\_viajes.ipynb

### **Construcción de datos 6: Creación de zonas universitarias**

Antes de continuar con la matriz de viajes, tuvimos que obtener las antenas dentro de las zonas universitarias, usamos la librería de Geopy donde a través de great circle, obtenemos el radio de la tierra y la distancia entre puntos, dándole un radio se obtienen las antenas que están alrededor de cierto número de kilómetros que le especificamos.

El proceso para obtenerlo fue nuevamente obteniendo las antenas únicas del dataset original, luego colocar la ubicación de cada universidad. También para este proceso consideramos que si la distancia entre la sede de la universidad y la



antena es menor de 1 km se toma como que esa antenna pertenece a la zona universitaria.

Todos los datos generados se pasan a un dataset con una columna conteniendo el nombre de la universidad, la latitud y longitud del punto de referencia y un array de las antenas situadas alrededor del punto de referencia.

Por último, guardamos el dataset generado para poder utilizarlo en el siguiente proceso.

Todo este proceso puede encontrarse en:

- 3.9.0\_Obteniendo\_antenas\_zonas\_universitarias.ipynb

### **Construcción de datos 7: Dataset final para modelado**

Ya que obtuvimos las antenas ubicadas en las zonas universitarias, nos dedicamos a hacer un procesamiento de los datos para obtener datos finales.

Primero cargamos 5000 datos de ejemplo para poder visualizarlos de mejor forma, ya revisando el dataframe vimos que habían varios datos que cuentan con una distancia recorrida de cero y de igual forma la duración de su viaje es de cero. Notamos que estos viajes se presentan mayormente cuando se llega al final, entonces esto llega a generar viajes inexistentes. Lo que realizamos para deshacernos de estos viajes fue el borrado con la columna de la distancia recorrida, ya que no hubo actividad alguna en ese viaje porque no realizó ningún movimiento.

Lo que hicimos fue tomar la distancia recorrida y creamos una columna de borrado lógico para comenzar el borrado de los datos, esto para poder re-revisar antes de tomar acción.

Después nos dimos cuenta que se iban a borrar 1,069,553 de registros que no cuentan con una distancia recorrida, al analizar la situación consideramos que el borrado de dichos registros no van a afectar los resultados del análisis.

Posteriormente cargamos el dataset creado anteriormente de las antenas que están en cada zona universitaria y extraemos los datos necesarios de las antenas y de las universidades para poder deshacernos de caracteres no necesarios y convertir a listas convencionales dichos datos.

Después, añadimos los datos para cada universidad en la lista de si es pública o privada, si tiene residencias y si el viaje se realizó dentro del kilómetro de antenas o desde fuera a la zona de la universidad.

Para obtener el dato de si la universidad es pública o privada, buscamos en internet si la universidad era pública o privada. Esto fue un poco más sencillo de investigar, ya que a pesar de que las universidades cuentan con varias sedes a lo

largo de todo Santiago de Chile, nosotros al saber si una universidad era pública o privada por consecuencia todas las sedes con las que cuenta esa universidad de igual forma serán públicas o privadas.

Para conocer si cada sede tenía residencia o no, hicimos blablabla

Para los viajes internos o externos, implementamos eso directamente en el algoritmo, este algoritmo nos genera un dataframe con la condición de que si la antena de destino está dentro de las antenas de las zonas de alguna universidad o de fuera hacia adentro en la lista de antenas dentro del kilómetro de radio establecido, se realice un registro de la hora de destino, la comuna de origen, el tipo de viaje, el tipo de universidad, y si tiene alguna residencia.

Todo este proceso se da por cada zona universitaria por variable creada con las listas de antenas hasta terminar de iterar en toda la matriz de viajes.

Notamos que de las 29 sedes que ingresamos, al final sólo nos aparecían 12 con viajes registrados, esto nos preocupó y nos regresamos repetidas veces a revisar si la matriz de viajes tenía un procedimiento correcto. Al confirmar que sí y probar detenidamente, decidimos hablar con la socia para contarle y hacerle otras preguntas.

Tras hablar con la socia de ese problema, nos comentó que estaba raro. Le hicimos preguntas de que si no se podría tratar de un día festivo o si realmente era un día entre semana, nos confirmó que no. Le contamos detalles como por ejemplo, que nuestro radio de análisis era de 1 km a partir del centro de la sede, tras pensarlo un poco nos comentó que le parecía bien ese radio de distancia y que nos enfocáramos en las 12 sedes que nos habían aparecido.

Las sedes con las que nos quedamos al final fueron las siguientes:

1. Campus Andrés Bello Uchile
2. Campus Beauchef Uchile
3. Campus Oriente UC
4. Casa central UC
5. El comendador UC
6. Juan Gómez Millas Uchile
7. North Campus (Dra. Eloisa Diaz) Uchile
8. San Joaquín UC
9. UDD Sede Santiago
10. UDP Facultad de economía y empresas
11. Campus Sur Uchile Uchile
12. Casa central Uchile


Por nuestra parte, pensamos que las posibles causas de que no aparecieran viajes eran porque durante el proceso de eliminación de velocidades extremas pudieron

haberse eliminado algunos datos por la alta concentración de antenas en la zona o porque el 50% de usuarios de Santiago de Chile cuentan con Movistar. Incluso pensamos en un error de nosotros aunque tras una extensiva revisión no pudimos encontrar algo raro en nuestros algoritmos.

De las variables que teníamos disponibles para realizar predicciones, nos comentó que también estaban muy bien, pero le preguntamos si tenía alguna idea para nosotros y nos comentó que podíamos añadir los metros cuadrados de cada sede y el número de carreras por cada sede, entonces nos pusimos manos a la obra.

Para obtener el número de metros cuadrados por sede, utilizamos Google My Maps para ubicar cada sede de las universidades, posteriormente, con otra ventana de Google Earth, estuvimos checando el perímetro correcto de las sedes para que en My Maps pudiéramos marcar correctamente lo que queríamos calcular de área en metros cuadrados.

Para obtener el número de carreras por sede, buscamos en internet con qué facultades contaba esa sede o campus, para después entrar a la página de cada facultad y empezar a buscar con qué carreras y postgrados son los que se impartían en esa facultad. Un ejemplo puede ser el campus San Joaquín de la Pontificia Universidad Católica de Chile que al ser la sede más grande de esa Universidad contaba con un total de 15 facultades, entonces fue entrar a cada página de las facultades y ver las carreras y postgrados que se ofertan para anotar un por una y luego sacar el conteo total de las facultades, para poder sacar el conteo total de la Universidad contando los resultados de cada facultad.

Conteo de las sedes:  Carreras en las Universidades

Por último, ya que procesamos nuevamente los datos observamos que todo está en orden y creamos una copia de los datos para poder seguir trabajando con dichos datos sin tener que reprocesar todo en caso de que nos equivoquemos. Para esto primero establecimos el timestamp como uno de Pandas para después poder acomodar los datos en orden del tiempo y por universidad, donde posteriormente obtendremos los viajes realizados a cada zona universitaria por hora empezando desde la hora 0 hasta la hora 23, observamos que los datos sigan correctos para poder guardar el dataset final en un archivo para luego poder trabajar con nuestros modelos.

Todo este proceso puede encontrarse en:

- 3.9.1\_analisis\_y\_procesamiento\_de\_la\_matriz.ipynb
- 3.9.2\_anadiendo\_nuevas\_variables.ipynb

## Finalización de limpieza

Al completar los procesos de limpieza, verificamos que todo estuviera en orden y unimos 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 (sin transformar aún a la matriz), el número de registros se redujo en 11,865,891 registros, por lo que el proceso de limpieza sí quitó una gran cantidad de datos que para nuestro análisis y modelo no tengan ruido.

Tras transformar el dataset a la matriz de viajes, ahora sólo contábamos con 1,203,010 de datos. Al transformarlo al dataset final para el modelado con las sedes de las universidades que registraron viajes, nos quedamos con un total de 1828 datos.

Como medida preventiva, en caso de que fuera necesario regresar y realizar correcciones a las limpiezas o generación de datos, tenemos todos los archivos almacenados para optimizar el procesamiento. Esto lo mantenemos porque sucedió varias veces que nos dábamos cuenta de alguna anomalía y rápidamente con todo el historial de archivos éramos capaces de corregir en poco tiempo los problemas.

### Descripción del dataset.

El dataset final que generamos, listo para la parte de modelado quedó con las siguientes columnas.

- **timestamp:** Es la hora en la que se realizó el viaje, formato 24 hrs.
  - Entero de 0 a 23
- **universidad:** Es el nombre de las sedes de las universidades.
  - Cadena de caracteres
- **comuna\_origen:** Es el nombre de la comuna en donde el usuario inició el viaje.
  - Cadena de caracteres
- **tipo\_de\_viaje:** Nos indica si el viaje es interno o externo.
  - Binario: 0 = Externo, 1 = Interno.
- **tipo\_de\_universidad:** Indica si las universidades son públicas o privadas.
  - Binario: 0 = Pública, 1 = Privada
- **tiene\_residencia:** Define si la sede de las universidades cuentan con residencias.
  - Binario: 0 = No tiene, 1 = Si tiene
- **carreras:** Es el número de carreras y posgrados existentes por cada sede universitaria.
  - Entero
- **tamaño\_uni:** Es el tamaño de la sede universitaria en metros cuadrados.
  - Entero
- **count:** Es la suma de viajes registrados por cada hora, comuna y universidad.
  - Entero

times tamp	universidad	comuna_origen	tipo_de_viaje	tipo_de_un iversidad	tiene_residencia	count	tamaño_uni	n_carreras
0	Campus Andrés Bello Uchile	Conchalí	0	0	1	1	45000	53
0	Campus Andrés Bello Uchile	Huechuraba	0	0	1	1	45000	53
0	Campus Andrés Bello Uchile	Lo Espejo	0	0	1	1	45000	53
0	Campus Andrés Bello Uchile	Providencia	0	0	1	7	45000	53
0	Campus Andrés Bello Uchile	Quilicura	0	0	1	1	45000	53
...	...	...	...	...	...	...	...	...
23	UDP Facultad de economía y empresas	Colina	0	1	0	1	26400	14
23	UDP Facultad de economía y empresas	Huechuraba	0	1	0	4	26400	14
23	UDP Facultad de economía y empresas	Huechuraba	1	1	0	9	26400	14
23	UDP Facultad de economía y empresas	Las Condes	0	1	0	1	26400	14
23	UDP Facultad de economía y empresas	Recoleta	0	1	0	10	26400	14

Consideramos que con estos datos ya estamos listos para realizar las predicciones del número de viajes que tiene cada zona universitaria.