**Documentación:**

Al comenzar el proyecto descargamos los dataset ofrecidos por Soy Henry (más de nueve), los mismos fueron repartidos entre los integrantes del grupo y cada uno analizó los datos que contenían. Una vez realizada la puesta en común de la información relevada, comenzamos a proponer propuestas de productos que se adecuarán a los datos analizados; la primer idea que surgió era crear un análisis para el ayuntamiento, del cual se decantaron recomendaciones para mejorar la situación actual en cuanto al tráfico, pudimos a través de un análisis exploratorio en diversos medios periodísticos informarnos de la creciente ola de accidentes acaecida en la zona durante el último año, tras encontrarnos con la dificultad de comercializar este producto, decidimos en base a los datos buscar nuevas alternativas. A continuación pensamos en realizar una app (al estilo Waze) que recomendará las rutas más seguras, pero allí nos topamos con que los datos se actualizan mensualmente en el dataset principal, por lo cual sería irrelevante ver la información sino es a tiempo real existiendo otras apps que ofrecen este servicio de manera muy eficiente. Finalmente surgió la idea de realizar una página que determine el riesgo de un conductor a la hora de adquirir una póliza para un vehículo, con el fin de optimizar las pérdidas de las aseguradoras . Una vez establecido nuestro producto, nos percatamos que había información relevante que nos parecía pertinente agregar ya que era determinante a la hora de establecer el riesgo en la póliza, y por ello se decidió investigar y agregar información desde la base de datos del departamento de policía de New York donde figura la cantidad y tipo de robos acontecidos en cada barrio, allí filtramos solo los que incluyen robo de autos. Junto con este escogimos el que contenia información sobre el volumen de tráfico y otro que aporta informacion adicional sobre los choques, el cual contenia el mismo numero identificador de choque(Id de choque), de alli pudimos agregar informacion faltante tal como rango etario, sexo, tipo de auto en los cuales habia faltantes.

Una vez establecido el producto y los datasets relevantes, utilizamos Socrata para consumir los datos desde las APIS que contienen nuestra información a utilizar, salvo en el dataset de volumen de tráfico que es consumido desde el CSV puesto observamos que la API se actualiza anualmente, por lo cual es innecesario e irrelevante realizarlo desde la API. A partir de este momento se comenzó con la ardua tarea de normalización y transformación. Aquí descubrimos que en el dataset principal los datos no vienen ordenados por fecha por cual debimos descargar todos los datos que contenía y luego formatearlos como fecha (ya esta columna venía establecida como cadena de texto) y recién allí pudimos ordenarlos por fecha de manera descendente. Una vez realizado este proceso filtramos la información de los últimos cuatro años, ya que los cambios acaecidos en una metrópolis de la magnitud de la ciudad de New York son enormes respecto a la implementacion de camaras con radar, patrulleros, etc; por lo cual llamar datos de más antigüedad no reflejaba el estado actual de la situación e influenciaría en nuestras proyecciones posteriores. Luego se procedió a traducir las columnas al español debido a que nos parecía pertinente para el correcto análisis, debido a que nuestro público receptor es de habla hispana. A continuación se procedió a borrar duplicados, debido a que hallamos que en el dataset principal venían duplicados porque contenían los datos de cada vehículo interviniente en el choque en una fila separada. A continuación normalizamos la escritura de los dataset, ya que los mismos contenían diferentes tipos de tipeados, en diversos estilos de escritura. El paso siguiente fue filtrar el tipo de auto debido a que allí descubrimos que había demasiadas categorías que no eran representativas del vehículo en si, los vehículos que no responden a ninguno de los tipos establecidos se asignaron de manera random a los tipos establecidos para que no se produzca un desbalance en el dataset principal, de trece quedaron siete. Con el fin de completar información y para posteriormente poder plasmar adecuadamente los datos, decidimos realizar una función que complete la latitud y longitud en las filas faltantes, y en los dataset que no la traían (volumen de tráfico), y también en base a la latitud y longitud se creó otra función que complete las localidades allí donde había faltantes de esta información. Se procedió a eliminar las columnas irrelevantes y se separó la fecha en tres campos (día, mes, año) puesto que en el modelo relacional nos exige este tipo de formato. Tras ese proceso, se desarrolló un sistema de puntaje de letalidad para ver qué tipo de auto y localidad eran más letales, para eso, cada herido por choque sumaba un punto mientras que los muertos, al ser más graves y caros sumaban 2 y así ya tener el dataset principal listo para AWS.

Una vez transcurrido todo este tedioso proceso se recopiló y optimizo el código, allí indento por grupo y se separe por secciones para mejor legibilidad.

Para montar nuestra base de datos en la nube elegimos los servicios ofrecidos por AWS. Para ello se realizó la creación de un bucket llamado ‘proyecto-henry’ con las políticas adecuadas, el cual recibe archivos de forma externa, en este caso desde Airflow para poder mantener actualizado nuestro Data Warehouse y tener almacenados los CSV con nuestra información original. A continuación para que la información se actualice automáticamente cada determinada cantidad de tiempo (él cual fue establecido de acuerdo a la frecuencia en que la API actualiza la información) se crearon los Crawler a través de Glue, los mismos se dividen en dos, uno de ellos recibe la información directamente desde Airflow y mientras que el otro envía esa información recibida hacia Redshift, nuestra base de datos. Antes de enviarlos a Redshift a los nuevos archivos recibidos les da formato de tablas, las cuales fueron construidas de acuerdo a nuestro modelo relacional y las mismas deben coincidir con el formato que existe en el modelo, esto evita duplicados además actualizar la base de datos.

En Redshift se aloja nuestra Base de Datos, para lograr esto primero creamos un clúster donde se almacena nuestra información y posteriormente se crean las tablas vacías para poblarlas con los datos de los CSV que se encuentran en el apartado de S3, escogimos como nuestra tabla de hechos principal la que contenía la cantidad de accidentes, la misma fue relacionada con los demás dataset a través del Id de choque en el caso del dataset que contiene las causas, a través de localidad en los casos de los dataset de Volumen de tráfico y Robos. Los datos tales como rango etario, tipo de vehiculo, localidad y sexo fueron establecidas como tablas de dimensiones. La ventaja de utilizar esta herramienta es que nos permite tener conexión a múltiples herramientas como lo es Power Bi, Qlik que son utilizadas para realizar las visualizaciones de nuestros datos y de esta manera están fácilmente actualizadas, solo con un clic. Redshift posee una interfaz de consulta, como cualquier otro gestor de SQL, en cual se pueden realizar querys para acceder a la información.

Al momento de realizar la automatización nuestra primera opción fue Github Actions, pero a las pocas horas de estar funcionando en la nube AWS nos bloqueó el acceso a la cuenta por dejar expuesto públicamente las credenciales de seguridad, luego de volver a darnos acceso y con una penalización pudimos ingresar nuevamente en la cuenta. Finalmente elegimos Airflow como la herramienta para conectar los datos desde la API con S3, para ello debimos a través de Windows subsystem crear un ambiente de trabajo Linux, puesto que Airflow solo corre en este servidor, una vez que pusimos en marcha el servidor root, pudimos acceder a la interfaz del navegador como localhost, allí se pueden ver los servicios activos y se pueden pausar, simplemente con correr un botón con el mouse. Airflow es de fácil mantenimiento, si hay que realizar cambios en el server lo puede realizar una persona sin conocimientos profundos y se puede realizar de manera remota. Airflow guarda los scripts de las funciones que ejecutan tareas, estas se denominan DAGs, las cuales se alojan en la carpeta con el mismo nombre(Dags), previamente creada, allí te dice si existe algún error que tengas haya que corregir en el código. Luego levantamos el Scheduler, que es lo que programa las tareas a partir del script de DAG que se escoja. Una vez establecido todos estos procesos volvemos a la interfaz web y escogemos qué tarea(DAG) queremos que se ejecute, esto va a seguir andando hasta que se pause manualmente desde el navegador.

Establecimos un flujo de trabajo, para ello se normalizo el código debido a que hay partes en que la varios integrantes intervinieron y cada integrante tiene su forma de programar y nombrar variables, se estableció el lenguaje común de Python como el parámetro normalizador a seguir ya sea en las variables, funciones y espaciamiento; se transformó funciones for a while(ya que descubrimos que while es más performante), debido a que para poder pasar variables entre tareas a Airflow es con xcom y tiene un límite de 2mb, por ello no se podían pasar dataframes de gran tamaño, por lo que se tuvo que se crear archivos y después levantarlos en la tarea siguiente. Una vez establecido esto dimos orden a las tareas para determinar cual se ejecuta en qué orden al ser subidas a S3.

En primer momento para realizar las visualizaciones habíamos elegido Qlik, por su interfaz amigable y ya que ofrece conectores con Redshift de fácil configuración , pero al momento de realizar el dashboard,nos dimos cuenta que esta herramienta no ofrece muchas opciones de personalización en cuanto al fondo y gama de colores en los gráficos, por ello terminamos escogiendo Power Bi, como la herramienta para realizar el dashboard, debido a que también tiene conector con Redshift y es fácilmente personalizable. Se escogió el nombre y logo de nuestro producto, así como también la paleta de colores. Para realizar el dashboard tuvimos que ponderar cuales eran las variables más relevantes para realizar los gráficos (Rango etario, Localidad, Sexo-que luego fue descartado-, Robo, Volumen de tránsito, Cantidad de choques, Tipo de Vehículo), se establecieron dos filtros, uno de Localidad y otro de Año, para que se pueda ver la incidencia de las distintas variables de acuerdo a estos dos parámetros. Escogimos los gráficos en función de la mejor visualización de la variable a representar, por ejemplo, en las variables de pocas dimensiones se escogió el gráfico de torta o anillo, en las variables donde quisimos dar relevancia al paso del tiempo se escogieron gráficos de línea, para poder hacer gráficos de comparaciones elegimos gráficos de barras -apiladas o rectas. Finalmente, para poder plasmar mejor la relevancia que tiene la localidad en cuanto a los choques y el volumen de tráfico decidimos utilizar un mapa de densidad, creado a través de Qlik, también habíamos realizado mapas en Plotly pero los mismos fueron descartados ya que no se apreciaba bien la incidencia de estas variables.

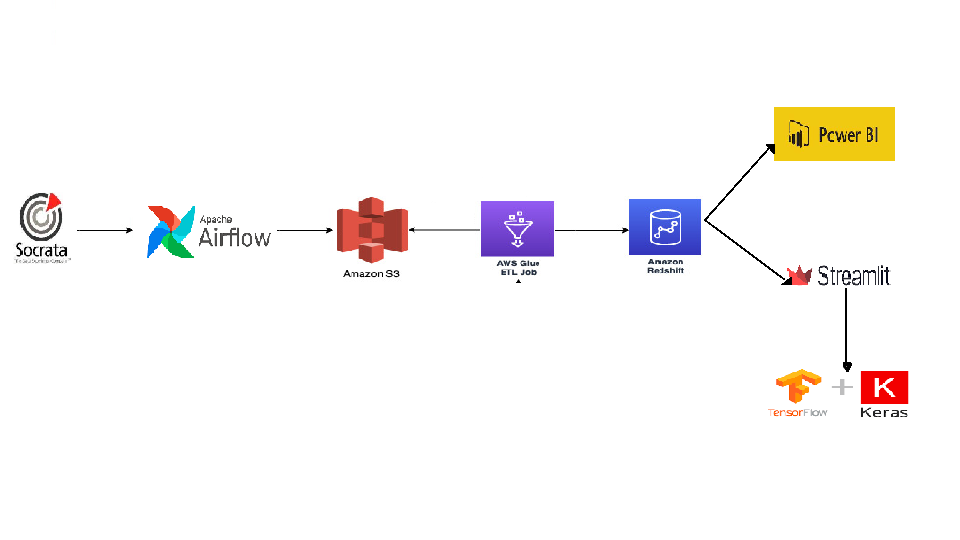
En el análisis pudimos observar datos que pudimos complementar con información adicional recopilada de diversos medios periodísticos. Observamos que pese a que Manhattan es el lugar con mayor volumen de tráfico, no es el lugar con mayor cantidad de choques como sesgadamente creíamos, al averiguar el por que de ello descubrimos que luego del once de septiembre se establecieron rigurosas medidas de seguridad para circular (tales como cámara de reconocimiento y radar, peaje, permisos especiales para aquellos vehículos que no eran transporte público, etc). Luego también se notó que las mujeres tenían muchos menos choques que los hombres, pero pudimos averiguar que es proporcionalmente menor la cantidad de mujeres al volante que de hombres, por lo cual decidimos descartar esta variable. Otro dato que nos pareció relevante es que los jóvenes suelen intervenir en menor cantidad de accidentes pero su letalidad es proporcionalmente mayor, respecto a las demás edades, aquí pudimos investigar y observar que el consumo de sustancias (ya sea alcohol o estupefacientes) es de vital incidencia, se reporta que pese a que son minoría las personas que dan positivos en este tipo de test, la letalidad cuando da positiva es muy alta, respecto a quienes daban negativo. Otra cuestión que notamos es que el barrio del Bronx es el posee mayor cantidad de robos, al averiguar el porqué de ello, pudimos identificar que este tipo de localidad es un lugar donde históricamente hay problemas con pandillas, narcotráfico y consumo de estupefacientes.

Consideramos que estos datos pueden optimizar el gasto en las aseguradoras puesto que achican la brecha de otorgarle una póliza a una persona que puede ser muy costosa respecto a su prima.

A continuación decidimos plasmar en una interfaz de Streamlit la información relevada y realizar un modelo probabilístico, para lograr dar el formato buscado a la página incorporamos lenguaje HTML y CSS, puesto que en las funciones básicas de Streamlit esto se ve muy limitado. La aplicación se divide en dos funciones principales una es accediendo como cliente y nos ayuda a cotizar una póliza de seguro, y la otra es ingresando como aseguradora y nos indica la probabilidad de que el conductor se involucre en un siniestro y que tan alto es el costo del choque, en este caso se debe acceder con un usuario y contraseña previamente asignadas por nuestro operador y accediendo con esta opción se despliegan dos gráficos de gauge que plasman las dos probabilidades analizadas, ambos fueron realizados a través de Plotly

Ambas funcionan conforme se van seleccionando los campos (que son las variables que consideramos relevantes), van filtrando la información del CSV para poder determinar los porcentajes de probabilidad de que un conductor con determinadas características se vea involucrado o no en un incidente. Las variables fueron ponderadas de acuerdo a la cantidad de hechos en las cuales se ven involucradas, esto se va acumulando a través de un algoritmo que determina qué tan alta o baja es la probabilidad y costo del accidente, si el vehículo es más nuevo.

Adicional a estas dos herramientas decidimos agregar una herramienta de clasificación de imágenes donde el usuario podrá cargar una foto de su vehículo y determinar en qué categoría se halla, en caso de no tenerlo muy claro, y sepan que opción elegir, para poder ver el valor del seguro. Para lograr esto se estableció un modelo de machine learning, usando Tensor Flow y Keras, que son librerías que intervienen en el reconocimiento de imágenes, se entrenó el modelo con los tipos de autos que queríamos que reconozca, para que luego se realice la correcta clasificación, aquí como los modelos de autos fueron establecidos arbitrariamente por el equipo de trabajo hubo que hacer el entrenamiento manual modelo por modelo.



Medios de los cuales obtuvimos información:

<https://okdiario.com/curiosidades/cuales-son-4-barrios-mas-peligrosos-nueva-york-4815430>

<https://lipsigabogadosdenuevayork.com/accidentes-vehiculares/estad%C3%ADsticas/>

<https://elpais.com/sociedad/2019/03/31/actualidad/1554036405_982184.html>

<https://elpais.com/internacional/2022-07-26/una-epidemia-de-violencia-vial-recorre-las-calles-de-nueva-york-con-cifras-record-de-siniestralidad.html>

<https://www.telemundo47.com/historias-destacadas/nueva-york-aumenta-el-patrullaje-para-combatir-el-exceso-de-velocidad-en-las-carreteras/2314307/>

<https://eldiariony.com/2022/07/26/sangre-en-el-asfalto-29-mas-muertes-en-accidentes-viales-en-nueva-york-el-bronx-es-la-zona-mas-peligrosa/>