Análisis Descriptivo y Predictivo Basado en Datos del Tráfico Vehicular en Antofagasta: Un Enfoque a partir de Reportes de Conductores

Richard Peña Bonifaz

10 de mayo de 2025

Resumen

El presente proyecto tiene como propósito desarrollar una herramienta efectiva para analizar y predecir, con alta confiabilidad, el comportamiento del tráfico vehicular en la ciudad de Antofagasta mediante el uso de datos provenientes de la plataforma Waze Cities. Waze, a través de su comunidad de usuarios, ofrece información en tiempo real que permite obtener una visión detallada de los eventos de tráfico que ocurren en la ciudad. Esta investigación genera información relevante para la gestión del tráfico, facilitando la toma de decisiones por parte de las autoridades locales con el fin de mejorar la seguridad vial y optimizar la eficiencia del flujo vehicular. A través del análisis y explotación de estos datos, se identifican patrones y tendencias que, integrados en la planificación urbana, contribuirán a optimizar rutas críticas, reducir la congestión y minimizar la probabilidad de accidentes.

En este estudio se emplearán técnicas de análisis de datos y métodos geoespaciales me-

diante el uso de la librería GeoPandas en Python, además de la aplicación de series temporales para proporcionar visualizaciones claras y comprensibles dirigidas al usuario final. Adicionalmente, se implementarán técnicas de aprendizaje automático para realizar predicciones de tráfico, anticipándose a eventos y contribuyendo al desarrollo urbano eficiente y seguro (Barceló & Casas, 2005).

1. Introducción

1.1. Descripción del problema

Antofagasta, una ciudad con más de 106,000 vehículos en circulación (de Seguridad de Tránsito, 2023), enfrenta desafíos significativos en la gestión de su tráfico vehicular. Durante el año 2023, se registraron 1,715 accidentes, los cuales resultaron en 31 fallecidos y 102 heridos graves (de Seguridad de Tránsito, 2023). La infraestructura vial limitada, sumada a la alta concentración de vehículos en un número reducido de arterias principales, agravó la congestión y elevó el riesgo de accidentes. A pesar de la existencia de estos problemas, no se cuenta con sistemas de monitoreo en tiempo real que permitan gestionar el tráfico de manera proactiva. Por ello, se aprovecharán fuentes de datos alternativas, como Waze, para recolectar información valiosa que facilite la toma de decisiones en materia de tráfico (Chen et al., 2015).

1.2. Objetivo general

Realizar un análisis exhaustivo del comportamiento del tráfico en la ciudad de Antofagasta basado en los eventos reportados por los conductores en la plataforma Waze. El objetivo final es generar información relevante que contribuya a la gestión eficiente del tráfico, mejorando la seguridad vial y optimizando el flujo vehicular.

1.3. Objetivos específicos

- Obtener datos suficientes y representativos sobre los eventos de tráfico en Antofagasta mediante la API de Waze Cities.
- Realizar un análisis descriptivo de los datos recolectados para identificar patrones y tendencias relevantes en el comportamiento del tráfico.
- Identificar los factores clave que influyen en la seguridad vial y la eficiencia del tráfico en la ciudad.
- Presentar información visualmente comprensible y útil para las autoridades de gestión vial, facilitando la implementación de políticas y acciones basadas en datos (Auld & Mohammadian, 2009).
- Utilizar los datos disponibles para desarrollar modelos predictivos de tráfico que permitan anticipar eventos y tomar decisiones proactivas en la gestión vial.

2. Marco teórico

El tráfico vehicular en entornos urbanos presenta un comportamiento complejo e impredecible, lo que dificulta su gestión eficiente. No obstante, el avance de las tecnologías móviles y la popularidad de aplicaciones como Waze permiten disponer de datos en tiempo real generados por los propios usuarios. Este proyecto se apoya en técnicas de análisis de datos y aprendizaje de máquinas (Machine Learning) para convertir esta información en herramientas útiles para la gestión vial. La utilización de datos geoespaciales, junto con la automatización de los procesos de recolección, análisis y visualización, constituye una solución costo-efectiva para mejorar la planificación del tráfico (Barceló & Casas, 2005).

2.1. Metodología

Una de las principales limitaciones en el análisis de fenómenos es la ausencia de datos suficientes y debidamente estructurados. Por ello, se diseñó una estrategia de recolección de datos que garantizara conclusiones con un nivel adecuado de certeza. La API de Waze Cities, que proporciona información en tiempo real sobre eventos activos, fue la fuente principal de datos. Para lograr un volumen de datos representativo, se implementó un servidor encargado de recopilar y almacenar esta información de manera continua.

Se realizó un análisis geoespacial con el objetivo de identificar puntos críticos, como vías principales, calles secundarias y zonas de alto tráfico. Este análisis se llevó a cabo utilizando GeoPandas, una herramienta de Python especializada en operaciones geoespaciales. Además, se analizaron series temporales para detectar patrones y tendencias del tráfico, identificando estacionalidades en el comportamiento. Los resultados se presentaron mediante técnicas de visualización que permitieron interpretar las tendencias y puntos de interés de manera efectiva.

El pipeline de datos incluyó una base de datos relacional para almacenar los datos, un flujo ETL (Extract, Transform, Load) para procesarlos y un servidor web para visualizarlos. Se utilizó PostgreSQL como base de datos, APScheduler para la programación de tareas y Dash como herramienta de visualización. Este enfoque permitió automatizar el flujo de datos y garantizar la actualización constante de la información.

Adicionalmente, se entrenó un modelo de clasificación para determinar la probabilidad de ocurrencia de accidentes en diferentes puntos de la ciudad. El modelo seleccionado fue XGBoost, el cual fue seleccionado utilizando técnicas de validación cruzada y optimización de hiperparámetros. Los resultados obtenidos permitieron identificar las variables más influyentes en la ocurrencia de accidentes, así como la probabilidad de ocurrencia en diferentes condiciones de tráfico. Para la selección de variables se utilizó

GridSearchCV, una técnica de búsqueda de hiperparámetros que permite encontrar la mejor combinación de variables para el modelo, se compararon modelos de regresión logística, árboles de decisión y XGBoost, siendo este último el que presentó el mejor desempeño en términos de precisión y sensibilidad.

Para la gestión del modelo, en cuanto a su implementación, mantenimiento y actualización, se programó con APSCheduler, una librería de tareas asíncronas en Python, que permite gestionar el ciclo de vida de los modelos de aprendizaje automático, desde su entrenamiento hasta su despliegue en producción y versionado.

En la parte del servidor, se desarrolló en Rust, un lenguaje de programación enfocado en la eficiencia en el uso de memoria y seguridad en el uso de la misma. Se utilizó la herramienta Memcached para generar un caché de los datos y de esta forma poder servirlos de manera más eficiente y rápida, debido al volumen de dato, se priorizó la unicidad de los datos, evitando generar copias, para no aumentar el uso de memoria.

Para el procesado de los datos en el cliente (Dash), se generan multiples workers que permiten servir la aplicación de forma eficiente, y se utiliza solo una instancia en memoria de los datos, compartidos por todos los datos, y para mantener la integridad de los datos, se implementó un Mutex para controlar el acceso de lectura a los mismos desde las tareas asíncronas.

La perspectiva general del flujo de datos se muestra en la Figura 1 y el flujo desde la API de Waze hasta el dashboard se puede observar en la Figura 2

3. Desarrollo

Para obtener los datos necesarios para el análisis, fue necesario generar un mecanismo para recolectar y almacenar la información de Waze Cities. Se implementó un servidor que se encarga de realizar peticiones a la API de Waze, recolectar los eventos de tráfico

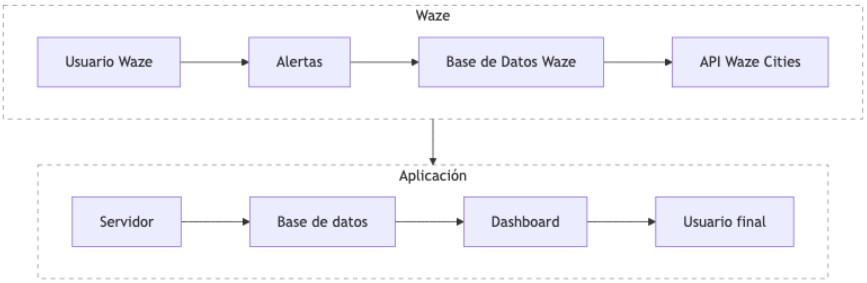


Figura 1: Pipeline general de datos

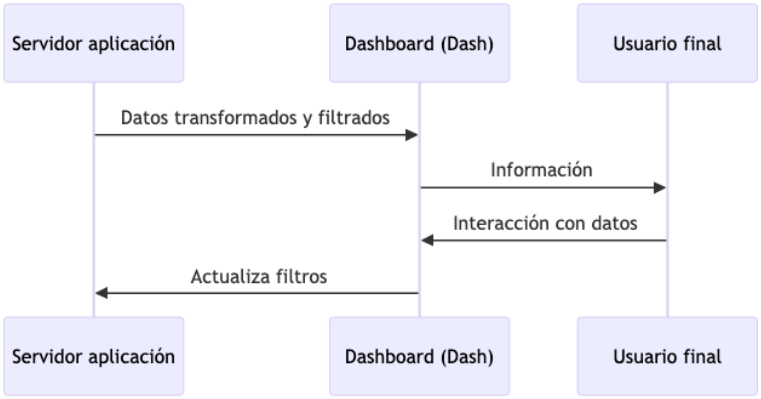


Figura 2: Flujo de información en dashboard

y almacenarlos en una base de datos PostgreSQL. Este servidor se ejecutaba de manera continua, actualizando la información cada 2 minutos. La estructura de la base de datos se muestra en la Figura 3.

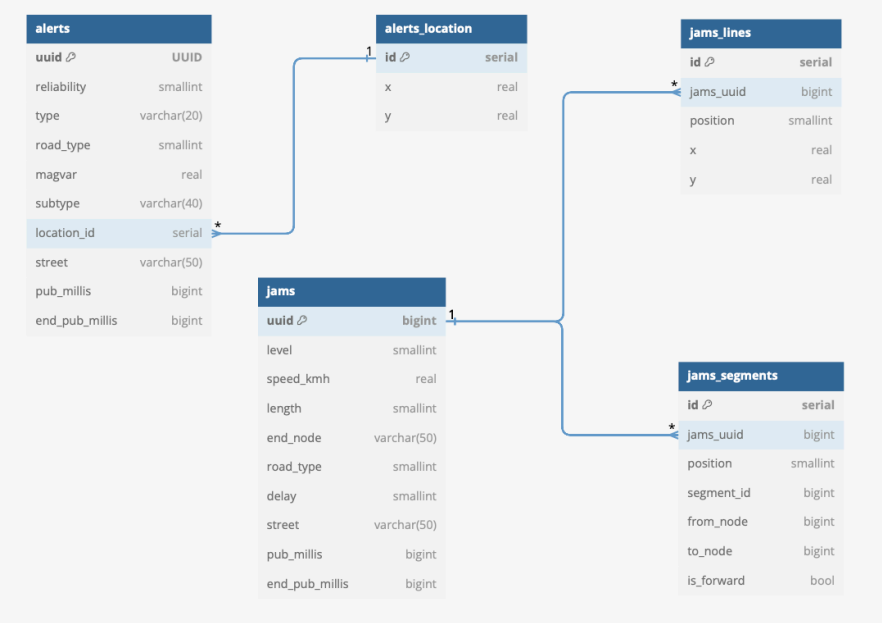


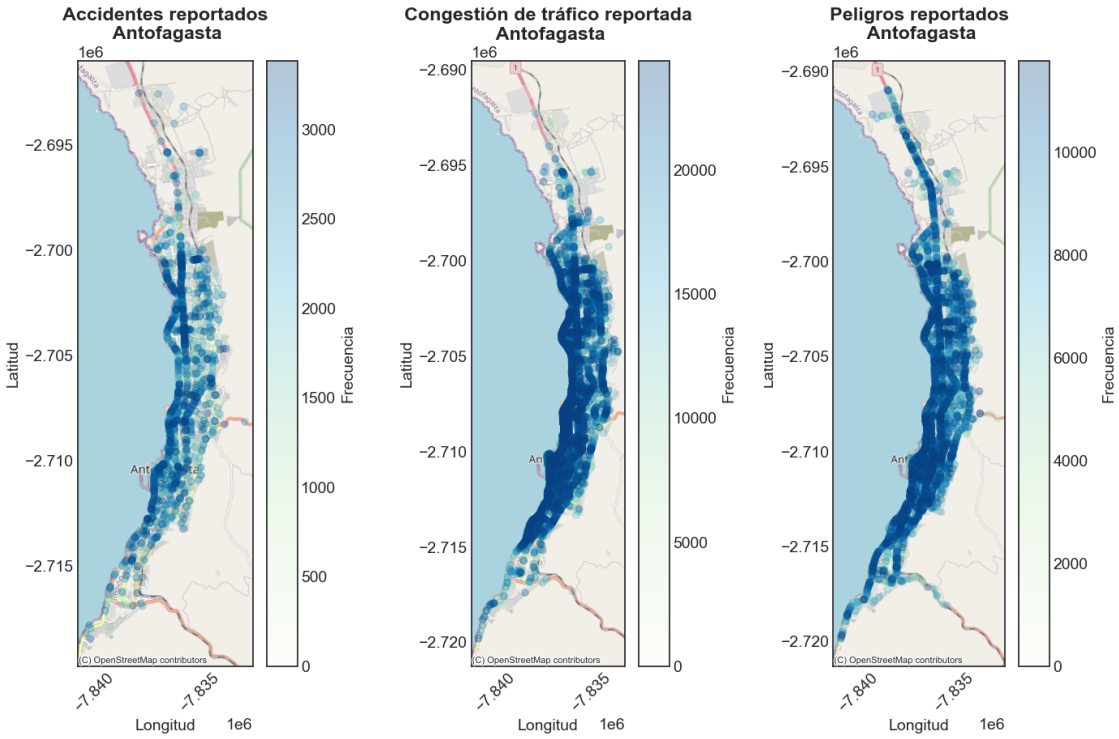
Figura 3: Diagrama de la base de datos

Existen dos estructuras principales, las alertas (alerts) y las congestiones (jams), el primero registra todos los eventos reportados por usuarios, tales como peligros, accidentes, congestión o rutas cerradas. En el caso de los eventos de congestión, son datos generados automáticamente por Waze, los cuales usando la geolocalización, estiman congestiones en diferentes puntos de la ciudad. Para este proyecto, se utilizó la información de las alertas. Sin embargo, se almacenaron los datos de congestión, para un posterior análisis, que está fuera del alcance de este estudio. Esta estructura se definió con base en los datos relevantes y completos que proporciona la API, los tipos de datos disponibles pueden ser consultados en la documentación de Waze (Waze, 2024).

Desde octubre de 2024 hasta Abril de 2025, se han acumulado 40429 alertas, las cuales se analizarán a lo largo de esde estudio.

3.1. Eventos en el espacio y tiempo

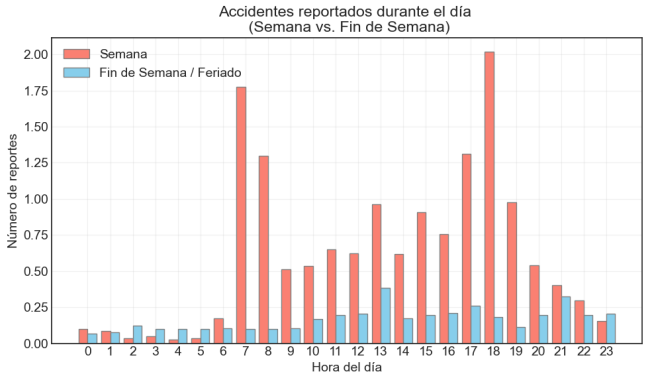
En la Figura 4 se observa la distribución espacial de los distintos tipos de eventos en la ciudad. Se aprecia una mayor concentración en las principales avenidas, especialmente en el caso de los accidentes, los cuales tienden a ocurrir con mayor frecuencia en los cruces de estas arterias.



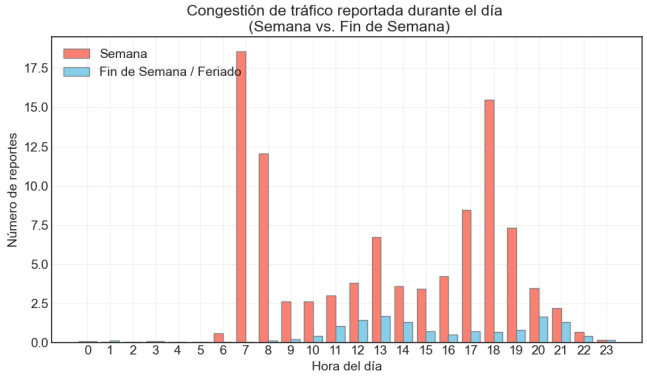
(a) Accidentes (b) Congestión (c) Peligros

Figura 4: Distribución espacial de eventos en Antofagasta

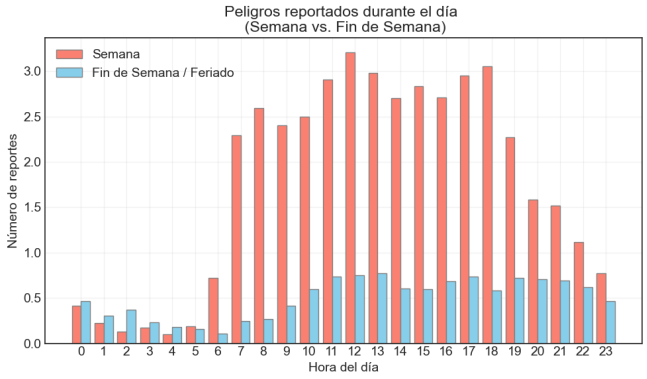
Para el análisis temporal, los eventos se agruparon por hora del día, calculando el promedio de eventos ocurridos por jornada. En la Figura 5 se evidencia una concentración de accidentes y congestiones durante las horas punta. En contraste, los eventos de tipo peligro presentan una distribución más uniforme a lo largo del día.



(a) Accidentes

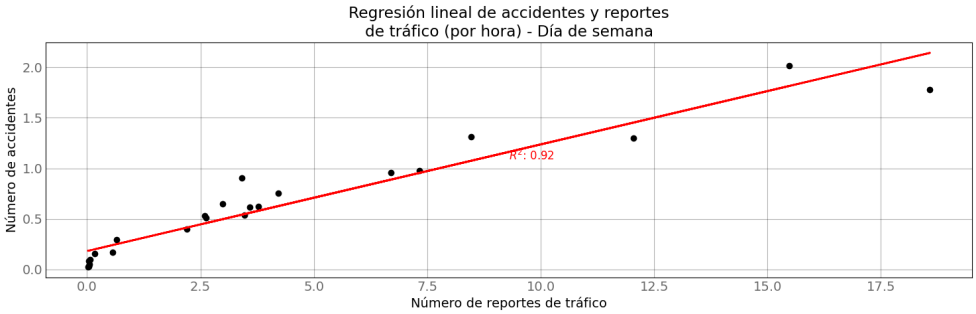


(b) Congestión

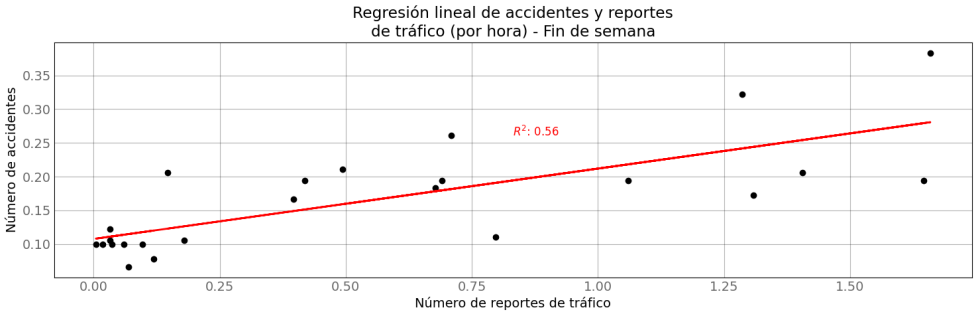


(c) Peligros

Figura 5: Eventos según hora del día en Antofagasta

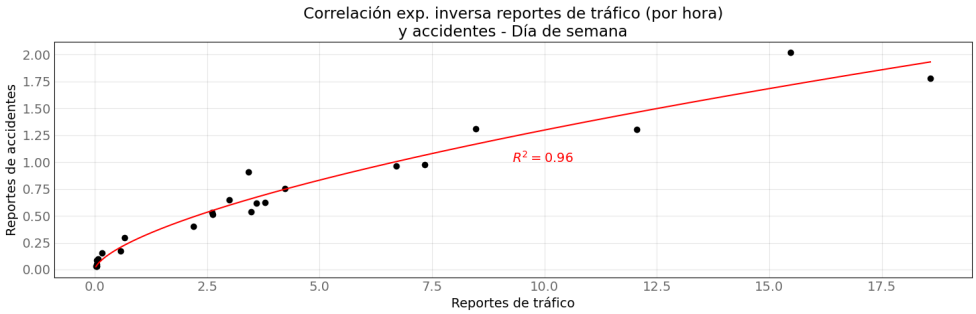


(a) Día de semana

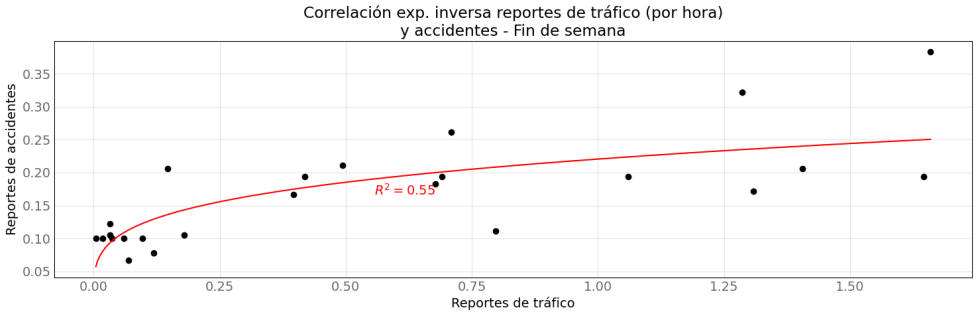


(b) Fin de semana

Figura 6: Correlación lineal entre accidentes y congestión



(a) Día de semana



(b) Fin de semana

Figura 7: Correlación exponencial inversa entre accidentes y congestión

3.2. Frecuencia por zona

Para representar conjuntos estáticos de puntos espaciales, se utiliza la estructura cKd-tree, una variante compacta del Kd-tree que codifica distancias relativas mediante spiral codes y las almacena usando Directly Addressable Codes (DACs), optimizando el uso de memoria sin sacrificar eficiencia en consultas espaciales (Gutiérrez et al., 2023). A cada zona la llamaremos cuadrante o segmento.

En la Figura 8 puede verse como se distribuyen los segmentos númericamente, partiendo desde el segmento 1 en la esquina inferior izquierda. La opacidad del color rojo va en relación con la cantidad de accidentes por segmento.

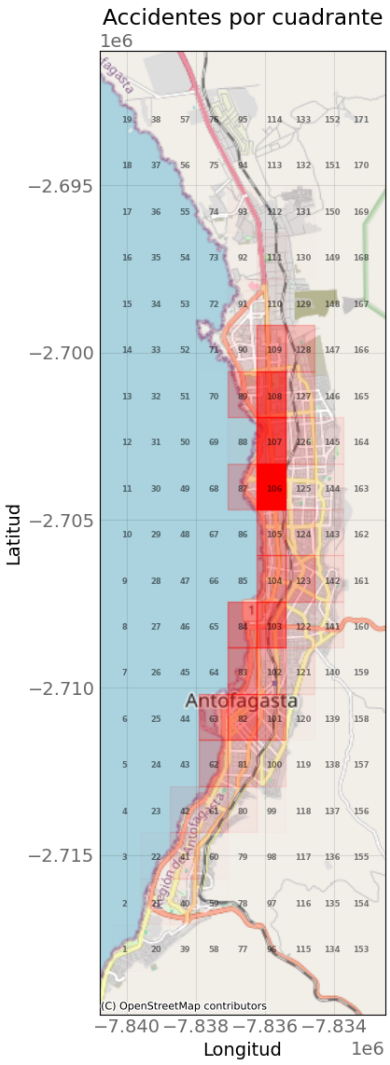
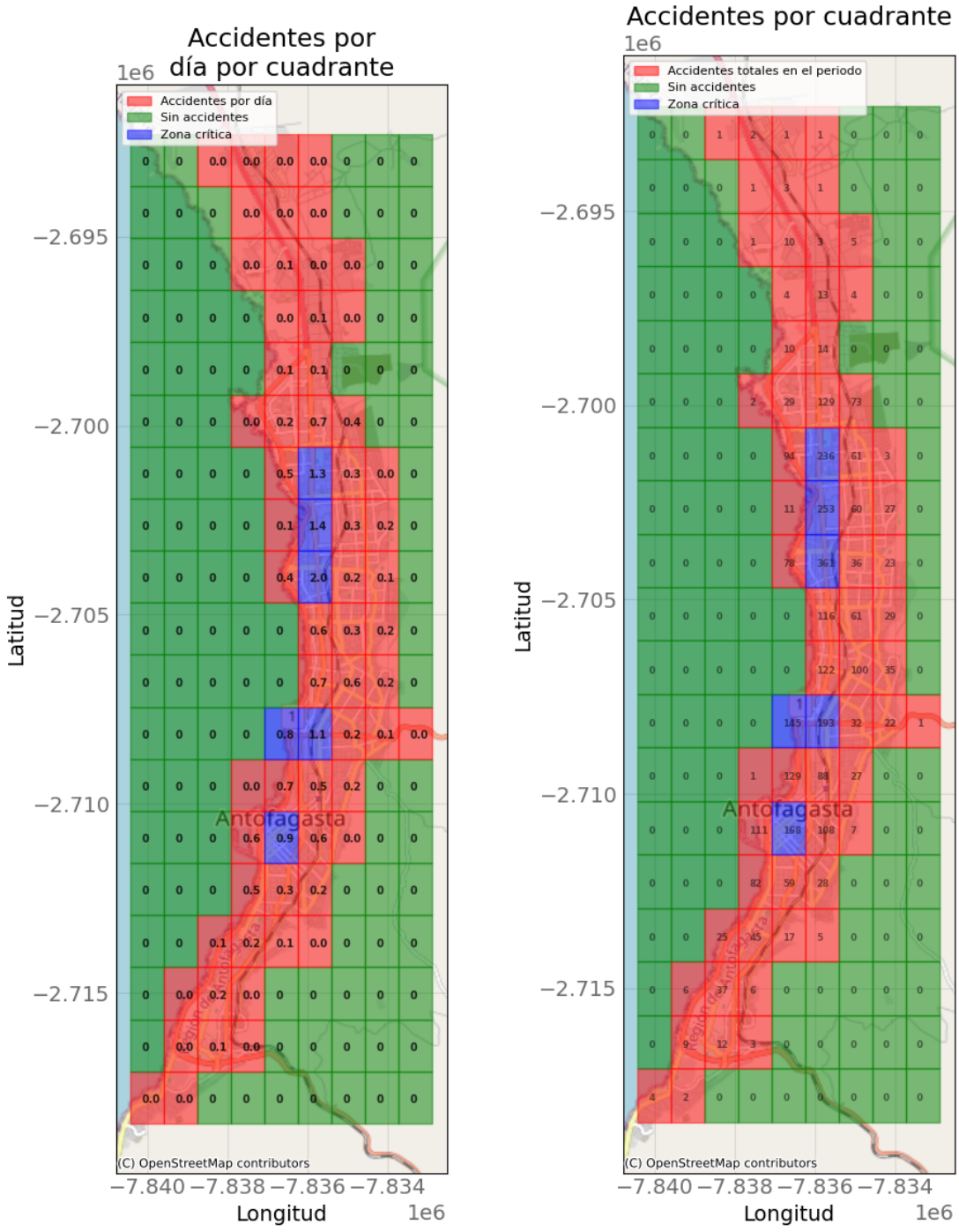


Figura 8: Identificación de cuadrantes en mapa de Antofagasta

Tomamos la media de accidentes diarios, los agrupamos por segmento, definimos que las zonas mayores a 0.8 accidentes por día son considerados críticos, construimos un mapa según la criticidad (9a), si tomamos la suma de los accidentes, tenemos la 9b que muestra la cantidad de accidentes reportados en el segmento en el periodo de estudio, que a su vez se clasifican como críticos si poseen más de 140 accidentes.



(a) Media de accidentes por día

(b) Total accidentes

Figura 9: Accidentes por segmento

3.3. Selección del modelo

Se evaluaron distintos algoritmos de clasificación para abordar el problema planteado. Inicialmente se probaron modelos clásicos como *Logistic Regression*, *K-Nearest Neighbors*, *Decision Tree*, entre otros. Sin embargo, estos no mostraron mejoras relevantes en las métricas evaluadas, por lo que se descartaron para la fase de ajuste de hiperparámetros.

Se procedió entonces a aplicar una búsqueda exhaustiva (*GridSearchCV*) sobre dos modelos con buen desempeño preliminar: **Random Forest** y **XGBoost**.

Random Forest. Se evaluaron los siguientes hiperparámetros:

- `max_depth`: [2, 5, 10]
- `n_estimators`: [20, 80, 100, 200]
- `min_samples_leaf`: [2, 4, 8]
- `max_leaf_nodes`: [2, 8]
- `max_features`: [2, 8]
- `class_weight`: [None, 'balanced']

Los mejores hiperparámetros obtenidos fueron:

- `class_weight`: None
- `max_depth`: 10
- `max_features`: 2
- `max_leaf_nodes`: 8
- `min_samples_leaf`: 2
- `n_estimators`: 100

Con estos valores, el modelo alcanzó una puntuación máxima de **0.7957** en validación cruzada.

XGBoost. Se aplicaron múltiples rondas de *grid search* con distintos rangos de hiperparámetros, refinando progresivamente los valores. Los mejores resultados se obtuvieron con:

- `colsample_bytree`: 0.7
- `gamma`: 1
- `learning_rate`: 0.1
- `max_depth`: 30
- `n_estimators`: 80

Este modelo logró una métrica de validación cruzada de **0.8665**, superando significativamente al resto de las alternativas evaluadas. Por este motivo, **XGBoost fue seleccionado como modelo final**.

En el Tabla 1 se puede observar la comparación de ambos modelos.

Modelo	Mejores Hiperparámetros	Mejor Score
Random Forest	`class_weight`: None `max_depth`: 10 `max_features`: 2 `max_leaf_nodes`: 8 `min_samples_leaf`: 2 `n_estimators`: 100	0.7957
XGBoost	`colsample_bytree`: 0.7 `gamma`: 1 `learning_rate`: 0.1 `max_depth`: 30 `n_estimators`: 80	**0.8665**

Cuadro 1: Comparativa de modelos ajustados con GridSearchCV

3.4. Generación de eventos negativos simulados

Dado que el conjunto original contenía únicamente eventos positivos (ocurridos), fue necesario generar instancias negativas (no ocurridos) para entrenar un modelo de clasificación binaria. Para ello, se implementó una estrategia inspirada en el enfoque de generación de eventos negativos artificiales propuesto por Goedertier et al. (2009), que permite representar problemas secuenciales como tareas de clasificación supervisada.

En nuestro caso, la generación se basa en una grilla temporal con intervalos de 5 minutos, sobre la cual se combinan todas las posibles ubicaciones (`group`) y tipos de evento. Luego, se identifican aquellas combinaciones que no se encuentran en los datos originales (es decir, donde no ocurrió un evento) y se etiquetan como eventos negativos (`happen` = 0). Estas instancias se muestrean de forma aleatoria para igualar la cantidad de eventos positivos, logrando así un conjunto balanceado con un 50 % de eventos

positivos y 50 % de negativos.

Este enfoque permite mantener la estructura temporal y geoespacial de los datos reales, evitando introducir ruido aleatorio, y facilitando la evaluación robusta de métricas como `recall` y `precision` (Goedertier et al., 2009).

3.5. Resultados del modelo seleccionado

La versión final del modelo (`v6`) se entrenó utilizando los hiperparámetros obtenidos durante la etapa de validación cruzada. El modelo fue registrado y versionado mediante *MLflow*, permitiendo un seguimiento detallado de sus parámetros y métricas.

Los parámetros más relevantes del modelo fueron:

- `max_depth` = 20
- `n_estimators` = 80
- `learning_rate` = 0.1
- `gamma` = 0.8
- `colsample_bytree` = 0.7

El modelo utiliza codificación *one-hot* y un esquema de clasificación binaria (`binary:logistic`), con una semilla aleatoria (`random_state = 42`) para garantizar la reproducibilidad.

En cuanto al rendimiento, se obtuvieron los siguientes resultados para una muestra de 64.708 eventos, con un 50 % de eventos ocurridos y un 50 % de no ocurridos. Esta proporción se logró mediante la simulación de eventos negativos sobre una grilla temporal de 5 minutos, generando combinaciones posibles de tiempo, localización y tipo de evento que no están presentes en los datos originales. Esto permite evaluar el modelo en condiciones balanceadas, facilitando la interpretación de métricas como `precision`, `recall` y `F1-score`.

- **Accuracy**: 0.8795
- **Recall**: 0.9076
- **Precision**: 0.8583
- **F1-score**: 0.8823
- **Promedio validación cruzada (10 folds)**: 0.7819

Respecto a la matriz de confusión, se registraron los siguientes valores:

- Verdaderos positivos: 5,539
- Verdaderos negativos: 5,843
- Falsos positivos: 965
- Falsos negativos: 595

Estas métricas evidencian un modelo con excelente capacidad para identificar correctamente los casos positivos, sin sacrificar precisión ni balance. La validación cruzada muestra estabilidad a lo largo de los distintos folds, lo que refuerza la robustez del modelo final.

3.6. Curva de aprendizaje del modelo

La Figura 10 muestra la curva de aprendizaje del modelo XGBoost entrenado. Se observa un rendimiento alto y consistente tanto en el conjunto de entrenamiento como en el de validación (aproximadamente 0.88–0.90), con una brecha mínima entre ambas curvas. Esto indica que el modelo generaliza adecuadamente, sin presentar sobreajuste ni subajuste. Además, el comportamiento plano de ambas curvas sugiere que incrementar el tamaño del conjunto de entrenamiento no tendría un impacto significativo en el rendimiento, lo cual evidencia una saturación en la capacidad de aprendizaje del modelo con las características actuales.

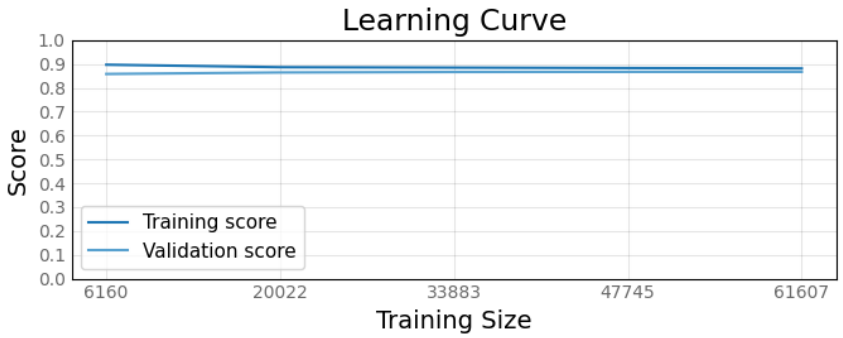


Figura 10: Identificación de cuadrantes en mapa de Antofagasta

Referencias

Auld, J., & Mohammadian, A. (2009). Framework for modeling activity planning as a sequence of planned activities. *Transportation Research Part B: Methodological*, *43*(6), 924-937.

Barceló, J., & Casas, J. (2005). Dynamic network simulation with AIMSUN. *Simulation approaches in transportation analysis*, 57-98.

Chen, C., Zhang, J., & Antoniou, C. (2015). Data-driven approaches to traffic prediction and management. *Transportation Research Procedia*, *10*, 12-24.

de Seguridad de Tránsito, C. N. (2023). Informe de accidentes de tránsito [Antofagasta, Chile.]. https://www.conaset.cl

Goedertier, S., Martens, D., Vanthienen, J., & Baesens, B. (2009). Robust Process Discovery with Artificial Negative Events. *Journal of Machine Learning Research*, *10*, 1305-1340.

Gutiérrez, G., Torres-Avilés, R., & Caniupán, M. (2023). cKd-tree: A Compact Kd-tree. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3365054

Waze. (2024). Waze API documentation. https://support.google.com/waze/partners/answer/13458165