Metodología de desarrollo

Descripción de objetivos:

El objetivo de este proyecto es ***desarrollar un modelo de clasificación binaria capaz de predecir la severidad de las lesiones en accidentes de tráfico*** *en función de variables relacionadas con las condiciones del entorno, características del siniestro y factores contextuales.*

La variable objetivo distingue entre:

- ***0 = Lesión leve***
- ***1 = Lesión grave***

👉🏼*Con ello se busca identificar de manera temprana los factores de riesgo asociados a la ocurrencia de lesiones graves, facilitando la toma de decisiones preventivas y estratégicas en seguridad vial.*

El modelo será evaluado mediante diferentes algoritmos de machine learning, incluyendo tanto técnicas tradicionales (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost) como métodos especializados para manejo de desbalanceo (Balanced Random Forest y Easy Ensemble).

La finalidad última es construir una herramienta predictiva robusta, que además pueda ser desplegada en un entorno de producción mediante una API (FastAPI), garantizando su utilidad práctica para la detección y prevención de accidentes con alto riesgo de lesiones graves.

1. Análisis exploratorio:

Objetivos:

1. ***Comprender la estructura del dataset:*** *Analizar la cantidad de registros y variables, así como sus tipos de datos, para planificar correctamente las etapas de limpieza y modelado.*
2. ***Detectar posibles problemas de calidad de datos:*** *Identificar valores faltantes, duplicados o inconsistencias que puedan afectar el rendimiento del modelo.*
3. ***Explorar la distribución de las variables numéricas y categóricas:*** *Reconocer patrones, sesgos y posibles relaciones con la variable objetivo.*
4. ***Identificar valores atípicos (outliers):*** *Localizar y evaluar su impacto en las variables numéricas para decidir su tratamiento.*
5. ***Analizar las categorías poco frecuentes:*** *Determinar aquellas etiquetas con baja representación para agruparlas y mejorar la capacidad de generalización.*

6. ***Evaluar correlaciones y asociaciones:*** *Entender cómo se relacionan las variables entre sí y con la variable objetivo, lo que servirá de base para la selección de características.*

Contenido:

1. ***Dimensiones y tipos de datos:*** *Se realizó un análisis preliminar del dataset para identificar la cantidad de registros y variables, así como los tipos de datos de cada columna (numéricas, categóricas y de fecha). Esto permitió planificar adecuadamente el preprocesamiento y la codificación de las variables.*
2. ***Valores faltantes:*** *Tras la revisión, no se identificaron valores faltantes en ninguna de las columnas, lo que simplifica el flujo de preparación de datos y evita la necesidad de imputaciones o eliminación de registros.*
3. ***Valores extremos (outliers):*** *Se detectaron valores extremos en algunas variables numéricas, especialmente en métricas de conteo como* `num_units` *e* `injuries_total`*. Estos outliers serán gestionados mediante técnicas según convenga, para evitar que distorsionen la construcción del modelo.*
4. ***Duplicados:*** *Se identificaron registros duplicados, pero representan un porcentaje muy bajo respecto al total de datos, por lo que no se espera que impacten significativamente en los resultados del modelo. Se mantuvieron en el dataset considerando que podrían reflejar eventos válidos del sistema de registro.*
5. ***Categorías poco frecuentes:*** *se identifican etiquetas con muy baja frecuencia en variables categóricas, que serán agrupadas en "OTHERS" para evitar problemas de codificación y mejorar la generalización del modelo.*

---

2. Preprocesado

Objetivos:

1. ***Garantizar la calidad de los datos:*** *Eliminar duplicados e inconsistencias para obtener un dataset limpio y confiable.*
2. ***Construir la variable objetivo (target):*** *A partir de las métricas disponibles, definir claramente la clase a predecir.*
3. ***Reducir el impacto de valores extremos:*** *Aplicar técnicas de tratamiento de outliers para mejorar la estabilidad de los modelos.*
4. ***Transformar variables categóricas:*** *Codificarlas adecuadamente (one-hot, label encoding u otros) para que los algoritmos puedan utilizarlas.*
5. ***Procesar variables temporales:*** *Extraer características relevantes (año, mes, día, hora, día de la semana) que aporten al modelo.*
6. ***Aplicar escalado y normalización:*** *Homogeneizar las magnitudes de las variables numéricas para facilitar el aprendizaje de los algoritmos sensibles a la escala.*

7. ***Optimizar la representatividad de las clases:*** *Garantizar un balance adecuado entre clases minoritarias y mayoritarias mediante técnicas como* `class_weight`*, sobremuestreo o ensambles específicos.*

Contenido:

- *Eliminación de duplicados*
- *Creación de variable objetivo*
- *Gestión de valore extremos (outliers)*
- *tratamiento de variables categóricas: Codificación*
- *tratamiento de variables de tiempo*
- *transformaciones necesarias (escalado, normalizacion, etc)*

Variable objetivo: `["injury_severity"]`

Creada a partir de:
`injuries_fatal` , `injuries_incapacitating` , `injuries_non_incapacitating` , `injuries_reported_not_evident` , `injuries_no_indication` , `"injuries_total"`

con los objetivos:

- ***Grave (1)*** *→ si* `injuries_incapacitating > 0` *o* `injuries_non_incapacitating > 0` *o* `injuries_fatal > 0` *o* `"injuries_total" = 2`
- ***Leve (0)*** *→ si* `injuries_reported_not_evident > 0` *o* `injuries_no_indication > 0` *o* `"injuries_total" = 0`

Esto convierte el problema en una clasificación binaria (0/1) donde:

- `0 = Leve`
- `1 = Grave`

⚠️ *Desbalanceo de clases*

significado para el modelo: *Si entrenamos sin ajustes, el modelo va a predecir 0 casi siempre, ignorando "grave" y "fatal". Una métrica como* `accuracy` *dará valores altos, pero será engañoso (porque basta con predecir "Leve" en el 82% de los casos).*

tecnica de oversampling→ ***SMOTE*** *para aumentar las muestras de clases pequeñas (grave/fatal)*

Eliminación de duplicados

Se detecta un número muy pequeño de registros duplicados, aproximadamente ***0,009% del total de los datos.*** *Dado que representan una fracción mínima, se procede a su eliminación para garantizar la consistencia del dataset, evitando que afecten de manera marginal la distribución de las variables y el entrenamiento del modelo. Esta decisión no impacta significativamente la representatividad de la muestra.*

Gestion de outliers ⚠️

*En el presente estudio **no se aplicó eliminación de outliers** debido a que, tras un análisis preliminar, se observó que gran parte de los registros catalogados como atípicos coincidían con la **clase minoritaria (accidentes graves)**, la cual representa aproximadamente menos del 20% del dataset.*

Eliminar estos valores hubiera supuesto:

- ***Pérdida de información crítica**: los accidentes graves, aunque poco frecuentes, son precisamente los casos de mayor interés para la predicción.*
- ***Aumento del desbalance de clases**: la supresión de estos registros habría reducido aún más la proporción de la clase minoritaria, dificultando la capacidad del modelo para aprender sus patrones.*
- ***Riesgo de sesgo en el modelo**: al eliminar los casos más extremos, el modelo tendería a centrarse únicamente en accidentes leves, reduciendo su utilidad en escenarios reales donde lo importante es detectar los eventos graves.*

*Además, dado que se emplearon algoritmos basados en árboles (Random Forest, Gradient Boosting, XGBoost, Balanced Random Forest y EasyEnsemble), que son **intrínsecamente robustos a valores extremos**, la decisión de mantener los outliers no compromete el rendimiento del modelo.*

*En este contexto, los denominados "outliers" no deben ser tratados como errores de medición, sino como observaciones raras pero válidas que reflejan situaciones de alta gravedad. Por ello, se optó por conservarlos y aplicar técnicas de **balanceo de clases** (SMOTE, class_weight, ensembles balanceados) que permiten mitigar el impacto del desbalance sin sacrificar la representatividad de los casos graves.*

*Agrupación **de categorías raras en "OTHERS"***

Objetivo:** **reducir la complejidad del modelo, mejorar la generalización y asegurar estabilidad en la codificación.

En la preparación de datos categóricos para modelos de Machine Learning, es frecuente encontrarse con categorías muy poco representadas. En nuestro dataset de accidentes de tráfico, varias variables categóricas presentan decenas de categorías con muy pocos registros, por ejemplo causas del accidente, tipos de control de tráfico o condiciones meteorológicas extremas.

Agrupar estas categorías raras en una categoría común, "OTHERS", tiene varias ventajas metodológicas:

1. ***Reducción de la cardinalidad:** Variables con muchas categorías pueden generar matrices de características muy grandes cuando se usa one-hot encoding. Esto incrementa el consumo de memoria y la complejidad del modelo sin aportar información significativa.*
2. ***Prevención de overfitting:** Categorías con pocos ejemplos no permiten al modelo aprender patrones confiables y no será capaz de generalizar*

3. ***Mejora de la estabilidad del modelo:*** *La consolidación de categorías raras permite que el modelo se enfoque en patrones frecuentes y representativos, mejorando la capacidad predictiva y la robustez ante nuevos datos.*
4. ***Facilidad de codificación:*** *Para técnicas como* ***one-hot encoding o target encoding****, tener categorías con muy pocos casos puede generar columnas con datos escasos o inestables.*
5. ***Mantenimiento y escalabilidad:*** *Facilita la actualización del modelo con nuevos datos, evitando la proliferación de categorías raras inesperadas en futuros datasets.*

Agrupar categorías raras en "OTHERS" es una práctica estándar de preprocesamiento que . Esto es especialmente relevante en datasets con alta cardinalidad y distribuciones de frecuencia muy desbalanceadas, como es el caso de los accidentes de tráfico.

Codificación de variables categóricas

La elección de la codificación de variables categóricas depende de dos factores clave:

1. ***Cardinalidad de la variable*** *(número de categorías distintas).*
2. ***Orden o jerarquía inherente*** *(si hay un orden lógico entre categorías).*

Tecnicas a utilizar:

1. Target Encoding

Variables: `prim_contributory_cause` , `trafficway_type` , `traffic_control_device` , `first_crash_type`

Justificación:

- *Estas variables tienen* ***alta cardinalidad*** *y muchas categorías raras.*
- *Target Encoding reemplaza cada categoría por la media de la variable target (* `injury_severity` *) para esa categoría.*
- *Permite al modelo capturar la relación directa entre la categoría y la gravedad del accidente.*
- *Evita crear demasiadas columnas como en One-hot encoding, reduciendo complejidad y consumo de memoria.*
- *Mejora la capacidad predictiva sobre categorías frecuentes y raras, especialmente útil en variables como causas de accidente o tipo de vía.*

2. One-hot Encoding

Variables: `road_defect` , `weather_condition`

Justificación:

- *Estas variables son nominales con pocas categorías significativas.*

- *One-hot encoding crea una columna binaria por cada categoría, evitando asumir relación ordinal inexistente.*
- *Permite al modelo aprender patrones específicos de cada categoría sin introducir jerarquía artificial.*
- *Adecuado para características que afectan el riesgo de accidente pero no tienen orden natural, como defectos en la vía o condiciones climáticas.*

3. Label Encoding

Variable: `crash_type` , `intersection_related_i`

Justificación:

- *Variable binaria (* `NO INJURY` *vs* `INJURY/TOW` *).*
- *Label Encoding convierte la variable en 0/1, suficiente para que el modelo la interprete correctamente.*
- *Mantiene la simplicidad y compatibilidad con modelos lineales o basados en árboles.*

4. Ordinal Encoding

Variables: `roadway_surface_cond` , `lighting_condition`

Justificación:

- *Estas variables presentan una* ***jerarquía natural o nivel de riesgo****.*
  - *Ejemplo:* `roadway_surface_cond` *va de "DRY" → "ICE" según peligrosidad de resbalón.*
- *Ordinal Encoding asigna números enteros respetando el orden, permitiendo que modelos lineales o árboles interpreten la gravedad o riesgo relativo.*
- *Captura la relación entre categorías y probabilidad de lesiones graves, algo que One-hot no puede representar directamente.*

Resumen metodológico:

Tipo de codificación	Variables	Razón principal
Target Encoding	`prim_contributory_cause, trafficway_type, traffic_control_device` , `first_crash_type`	Alta cardinalidad, captura relación con target, reduce complejidad
One-hot Encoding	`road_defect, weather_condition`	Nominales, pocas categorías, evita jerarquía artificial
Label Encoding	`crash_type` , `intersection_related_i`	Binaria, simple y eficiente
Ordinal Encoding	`roadway_surface_cond, lighting_condition`	Variables con orden/jerarquía, captura relación de riesgo/gravedad

⚠️ ***No usar Target Encoding en la muestra de test directamente; aplicar la media calculada en el train.***

Codificacion de variables de tiempo: `crash_date`

1. `crash_hour` ***(0–23) : Codificación cíclica****:*

- *Tiene naturaleza circular (23h y 0h son muy cercanas). El modelo aprende mejor patrones horarios (ejemplo: más accidentes fatales de madrugada).*

2. `crash_day_of_week` ***(1–7)***

- *Útil porque los fines de semana suelen tener perfiles de accidentes diferentes (ej: alcohol).*

Escalado/ Transformaciones

Justificación de no escalar los datos:

1. ***Naturaleza del modelo:*** *Algunos modelos de machine learning* ***no son sensibles a la escala de las variables****.*

- *Ejemplos claros:*
  - *Árboles de decisión (* `DecisionTreeClassifier` *)*
  - `Random Forest`
  - `Gradient Boosting / LightGBM / XGBoost`
- ***Estos modelos hacen divisiones basadas en umbrales de las características, no en distancias, por lo que la magnitud de los valores no afecta la construcción del árbol ni las decisiones de partición.***

2. ***Simplicidad y eficiencia:***
   - *Evitamos pasos adicionales de preprocesamiento que incrementan la complejidad y el tiempo de ejecución.*
   - *No necesitamos pipelines adicionales para escalar, lo que hace el workflow más directo y reproducible.*
3. ***Mantenimiento del valor de variables categóricas codificadas:*** *Algunas codificaciones, como* ***t****arget encoding o ordinal encoding, generan valores que ya contienen significado en su magnitud (media de la target o ranking de severidad). Escalar podría diluir esta información útil.*

Balanceo con `SMOTE` *(Synthetic Minority Over-sampling Technique)*

Descripción: SMOTE es una técnica de sobremuestreo utilizada para balancear datasets donde existe un ***desbalance de clases****, es decir, cuando algunas clases tienen muchas menos muestras que otras.*

En lugar de simplemente duplicar las instancias de la clase minoritaria, SMOTE ***genera nuevas muestras sintéticas*** *interpolando entre ejemplos existentes de esa clase. Esto se hace seleccionando un ejemplo minoritario y un vecino más cercano de la misma clase, y creando un nuevo punto en el espacio de características entre ellos.*

Cómo funciona:

- ***Balance de clases:*** *Al equilibrar el número de instancias entre clases, SMOTE evita que los modelos se sesguen hacia la clase mayoritaria. Esto es especialmente crítico en problemas donde la clase minoritaria es la más relevante (predicción de lesiones graves).*
- ***Mejor generalización:*** *Al generar ejemplos sintéticos en lugar de duplicar datos, se reduce el riesgo de overfitting comparado con el sobremuestreo simple.*
- ***Compatibilidad con modelos:*** *SMOTE puede aplicarse a cualquier modelo supervisado que se beneficie de un dataset balanceado, incluyendo regresión logística, árboles de decisión, Random Forest, Gradient Boosting y XGBoost.*

Limitaciones a tener en cuenta:

- *Si se aplica en exceso, puede generar overfitting si el modelo aprende características muy específicas de los puntos sintéticos.*

⚠️ *Es importante aplicar SMOTE* ***solo sobre el conjunto de entrenamiento*** *y nunca sobre el conjunto de prueba, para evitar "filtrar" información del test al entrenamiento.*

---

3. Entrenamiento y Evaluación

Contenido:

- ***Diccionario de modelos:*** *Este apartado proporciona un marco conceptual sólido que facilite la comprensión de los algoritmos utilizados en el proyecto.*

La diversidad de técnicas de machine learning y estadística puede resultar compleja para la interpretación de resultados y la selección de modelos adecuados. Al documentar cada algoritmo con detalle, se busca:

1. ***Claridad conceptual:*** *Explicar cómo funciona cada modelo, sus supuestos, ventajas y limitaciones, evitando malentendidos o interpretaciones incorrectas de los resultados.*
2. ***Transparencia metodológica:*** *Justificar la elección de cada algoritmo en función del problema planteado, de la naturaleza de los datos y de los objetivos del estudio.*
3. ***Reproducibilidad:*** *Facilitar que otros investigadores o profesionales puedan replicar el análisis siguiendo la misma lógica metodológica.*
4. ***Soporte en la toma de decisiones:*** *Permitir comparar algoritmos y seleccionar aquellos que ofrezcan un balance óptimo entre precisión, interpretabilidad y eficiencia.*

- ***Métricas de evaluación:*** *Este apartado es fundamental para garantizar que los modelos desarrollados sean validados de manera rigurosa y objetiva.*

La evaluación no solo permite medir el desempeño de cada algoritmo, sino también identificar posibles errores, sesgos o limitaciones. Al documentar las técnicas y métricas utilizadas, se asegura:

1. ***Rigurosidad científica:*** *Se adoptan métodos reconocidos que permiten evaluar la efectividad de los modelos de manera cuantitativa y reproducible.*
2. ***Comparabilidad de modelos:*** *Facilita la comparación entre diferentes algoritmos utilizando criterios uniformes, lo que respalda la selección del modelo más adecuado.*
3. ***Transparencia y confiabilidad:*** *Permite a otros investigadores o profesionales comprender cómo se midió el rendimiento y cómo se interpretan los resultados.*
4. ***Optimización del desempeño:*** *Identifica áreas de mejora en los modelos mediante métricas que reflejan precisión, sensibilidad, estabilidad y generalización.*

Diccionario de modelos

Objetivos:

1. ***Describir detalladamente cada algoritmo*** *empleado en el proyecto, incluyendo su principio de funcionamiento y la naturaleza de los datos para los que es más adecuado.*
2. ***Explicar los parámetros clave*** *de cada modelo y cómo influyen en el comportamiento y desempeño del mismo.*
3. ***Comparar enfoques*** *(por ejemplo, regresión vs. árboles vs. ensambles) para evidenciar las diferencias en rendimiento y aplicabilidad según los distintos escenarios del análisis.*
4. ***Generar una guía de referencia rápida*** *para que cualquier miembro del equipo o lector pueda entender los modelos sin necesidad de consultar fuentes externas.*
5. ***Establecer criterios de evaluación*** *que permitan justificar la selección del modelo final en base a métricas objetivas y coherentes con los objetivos del proyecto.*

- ***Regresión logística*** `(LogisticRegression)`***:*** *Es un Modelo estadístico lineal para los problemas de clasificación.*

Cómo funciona: Calcula una combinación lineal de las variables y aplica ***la función logística sigmoide*** *para convertir Z en una probabilidad entre 0 y 1*

Ventajas: Simple, interpretable.

Limitaciones: No captura relaciones complejas/no lineales sin transformación de variables.

- ***Árboles de decisión (*** `DecisionTreeClassifier`***) :*** *Es un Modelo basado en estructura de árbol, basado en reglas jerárquicas tipo "si-entonces".*

Cómo funciona: Selecciona la mejor característica en cada nodo según un criterio ***(Gini, Entropía).*** *Divide el dataset en subconjuntos homogéneos de clases y se repite recursivamente hasta un límite de profundidad o pureza.*

Ventajas: Interpretación gráfica, fácil de entender y captura relaciones no lineales.

Limitaciones: Propenso al sobreajuste (sensible al ruido.) *e inestable frente a pequeños cambios en los datos.*

- ***Random Forest (*** `RandomForestClassifier` ***):*** *Es un algoritmo de ensamblaje basado en árboles de decisión mediante* ***bagging (bootstrap aggregating)****.*

Cómo funciona: Crea ***muchos árboles independientes*** *con submuestras aleatorias de datos y features; las predicciones se promedian (clasificación por votación).*

Ventajas: Reduce sobreajuste, Robusto a ruido y datos desbalanceados (moderadamente).

Limitaciones: Menos interpretable, más lento.

💡 ***Bagging:*** *Es una técnica de* ***ensamble*** *diseñada para mejorar la* ***estabilidad y precisión*** *de los modelos de predicción, especialmente los que son inestables como los árboles de decisión*

Cómo funciona:

1. ***Muestreo con reemplazo (bootstrap):*** *Se generan múltiples subconjuntos de datos a partir del dataset original, seleccionando aleatoriamente ejemplos con reemplazo.*
2. ***Agregación de predicciones (aggregating):***

- *Para* ***problemas de regresión:*** *se hace el promedio de las predicciones de todos los modelos.*
- *Para* ***problemas de clasificación:*** *se realiza un voto mayoritario (la clase más frecuente entre los modelos).*

Ventajas: Reduce la ***varianza*** *del modelo, haciendo que sea más estable frente a pequeñas variaciones en los datos. Mejora la* ***generalización*** *y disminuye el riesgo de* ***overfitting*** *comparado con un solo modelo complejo.*

Limitaciones:

- *Aunque son eficaces para modelos con alta varianza* ***no mejora modelos que ya son sesgados, s****i el modelo base tiene un sesgo alto, el ensamble seguirá siendo sesgado.*
- *Aunque cada subconjunto tiene el mismo tamaño que el dataset original, puede contener duplicados y omitir algunos datos.*

- ***Gradient Boosting (*** `GradientBoostingClassifier` ***):*** *Ensamble secuencial de árboles basado en* ***boosting****, donde cada árbol corrige errores del anterior y va optimizando gradualmente la función de pérdida.*

Cómo funciona: Cada nuevo árbol corrige los errores de los anteriores:

1. *Entrena un árbol simple.*
2. *Calcula los* ***residuos*** *(errores).*
3. *Entrena un nuevo árbol para predecir los residuos.*
4. *Suma los árboles ponderados por un* ***learning rate****.*

Ventajas: Muy potente, alto rendimiento, captura relaciones no lineales complejas.

Limitaciones: Más lento que Random Forest y sensible a hiperparámetros.

- ***Gradient Boosting (*** `XGBoost` ***):*** *Implementación optimizada y regularizada del Gradient Boosting.*

Cómo funciona: Usa mejoras en regularización, paralelización y manejo de valores faltantes para mayor eficiencia y precisión.

Ventajas: Muy rápido y preciso.

Limitaciones: Complejo de tunear.

> 💡 ***Boosting:*** *es otra técnica de* ***ensamble en machine learning****, pero a diferencia de* ***bagging****, se centra en* ***reducir el sesgo y mejorar la precisión*** *al combinar modelos débiles de manera secuencial.*
>
> *Es un método que construye árboles de decisión secuenciales. Cada árbol nuevo intenta corregir los errores de los árboles anteriores, optimizando una función de pérdida (loss function) usando gradientes.*
>
> *Cómo funciona: Después de cada modelo, se aumenta el peso de los ejemplos mal predichos para que el siguiente modelo los enfoque mejor.*
>
> *Combinación de predicciones:*
>
> - *Las predicciones finales son una combinación ponderada de todos los modelos.*
> - *En clasificación, se hace un voto ponderado; en regresión, un promedio ponderado.*
>
> *Ventajas: Mejora significativamente la precisión de modelos simples (árboles pequeños) y reduce tanto sesgo como varianza en muchos casos.*
>
> *Limitaciones:*
>
> 1. ***Mayor riesgo de overfitting*** *si se permite que los modelos sean muy complejos o hay mucho ruido.*
> 2. ***Mayor costo computacional*** *debido a la secuencia de entrenamientos.*
> 3. ***Más sensible a outliers****, porque los errores se ponderan más fuertemente.*

Comparacion resumida de Boosting y Bagging

Aspecto	Bagging	Boosting
Objetivo	Reducir varianza	Reducir sesgo
Modelo base	Complejo o inestable	Simple o débil
Entrenamiento	Paralelo	Secuencial
Errores previos	No se consideran	Se corrigen ponderando los errores
Ejemplo	Random Forest	XGBoost, AdaBoost, GradientBoosting
Riesgo de overfitting	Bajo	Medio-alto si hay ruido

- ***Balanced Random Forest (*** `BalancedRandomForestClassifier)` ***:*** *Variante de Random Forest diseñada para* ***clases desbalanceadas****.*

Cómo funciona: Cada árbol se entrena sobre una muestra balanceada (submuestreo de la clase mayoritaria + todos los casos minoritarios).

Ventajas: Mantiene más información que un simple undersampling, bueno para desbalance extremo y buen recall de la clase minoritaria.

Limitaciones: Tiende a producir más falsos positivos.

- ***EasyEnsemble (*** `EasyEnsembleClassifier)` ***:****Técnica de ensamble para* ***desbalance de clases****, basada en* ***bagging*** *y* ***undersampling múltiple****.*

Cómo funciona: Genera varios subconjuntos balanceados de los datos (undersampling) y entrena un AdaBoost en cada uno; los resultados se combinan.

Ventajas: Estable, aprovecha el balance sin perder información.

Limitaciones: Más costoso en cómputo.

Resumen de diferencias principales:

Algoritmo	Tipo	Cómo actúa	Pros	Contras
LogisticRegression	Lineal	Probabilidades con sigmoide	Simple, interpretable	No captura relaciones no lineales
DecisionTreeClassifier	Árbol	Divide recursivamente según criterio de pureza	Interpretación clara, no lineal	Overfitting
RandomForestClassifier	Ensamble (bagging)	Varios árboles + votación	Robusto, reduce overfitting	Menos interpretable
GradientBoostingClassifier	Ensamble (boosting)	Árboles secuenciales corrigiendo errores	Muy potente, no lineal	Sensible a hiperparámetros

Algoritmo	Tipo	Cómo actúa	Pros	Contras
XGBoost	Ensamble (boosting optimizado)	Gradient Boosting con paralelismo y regularización	Rápido, preciso, manejo NA	Complejo de configurar
BalancedRandomForestClassifier	Ensamble balanceado	RF con submuestras balanceadas	Maneja desbalance	Pierde info clase mayoritaria
EasyEnsembleClassifier	Ensamble balanceado múltiple	Varios submodelos sobre subsets balanceados	Maneja desbalance extremo	Computacionalmente caro

Métricas

Objetivos

1. ***Definir las métricas de evaluación*** *seleccionadas, indicando qué mide cada una (precisión, recall, F1-score, AUC, etc.) y por qué son relevantes según la naturaleza del problema y la distribución de los datos.*
2. ***Establecer criterios objetivos de comparación*** *entre modelos, permitiendo seleccionar aquel que ofrezca el mejor equilibrio entre rendimiento y generalización.*
3. ***Facilitar la interpretación de resultados****, asegurando que los valores de las métricas puedan ser comprendidos y utilizados para decisiones posteriores.*
4. ***Promover la reproducibilidad*** *del análisis, dejando claro cómo se calcularon las métricas y cómo se aplicaron las técnicas de evaluación en cada modelo.*

- ***Precisión:*** *Qué porcentaje de predicciones positivas fueron correctas. Importante si los falsos positivos tienen costo.*
- ***Recall:*** *Qué porcentaje de casos reales positivos fueron detectados. Crítico si los falsos negativos son costosos (no detectar una lesión grave).*
- ***F1-score:*** *Combina precisión y recall,* ***útil para datasets desbalanceados.***
- ***Accuracy:*** *Útil como referencia global, pero* ***menos confiable en problemas desbalanceados****.*
- ***ROC-AUC:*** El área bajo la curva ROC mide la capacidad del modelo para **diferenciar entre leve y grave**. *Un valor cercano a 1 indica buen poder discriminativo*

Elementos clave para clasificación binaria:

- ***TP (True Positive):*** *Leve correctamente identificado como leve*
- ***TN (True Negative):*** *Grave correctamente identificado como grave*
- ***FP (False Positive):*** *Leve clasificado erróneamente como grave*
- ***FN (False Negative):*** *Grave clasificado erróneamente como leve*

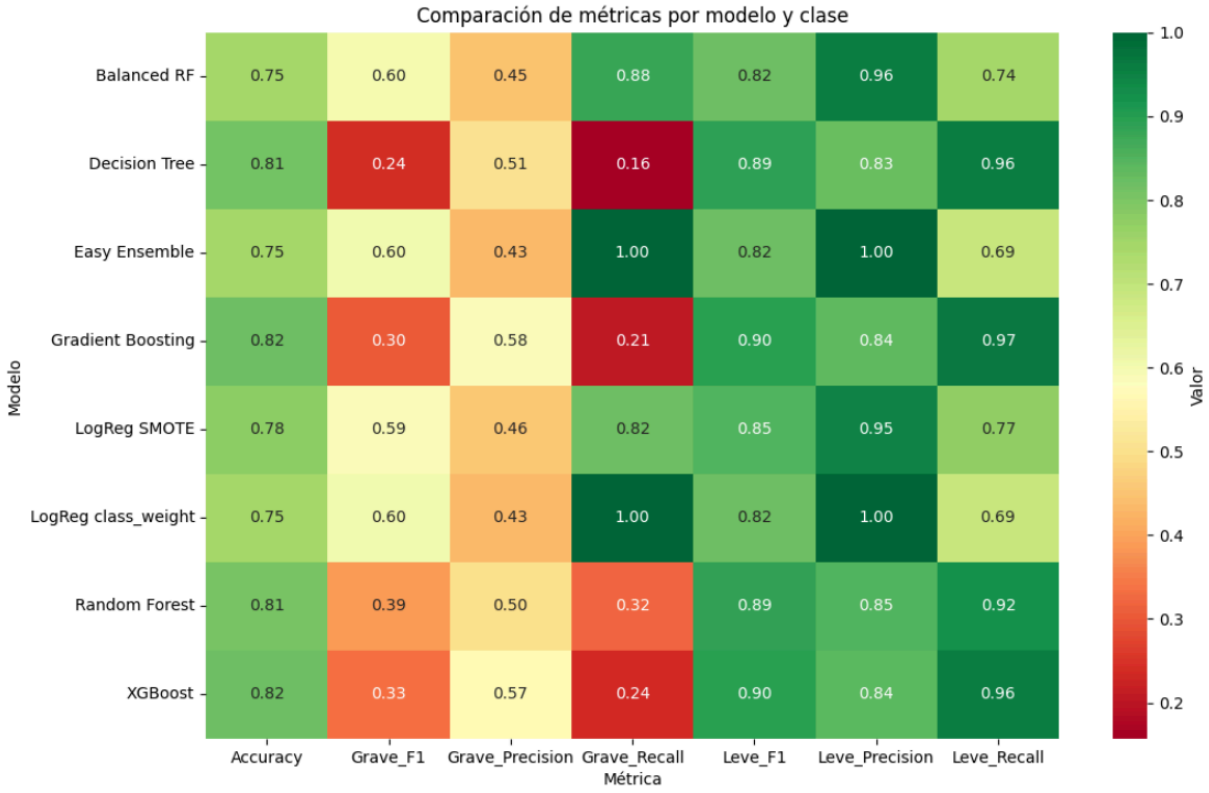
Conclusiones individuales por modelo

- ***Clase 0 = Leve***
- ***Clase 1 = Grave***

Observaciones generales sobre las métricas

Modelo	F1 Leve	F1 Grave	Recall Leve	Recall Grave	Precision Leve	Precision Grave	Accuracy
Decision Tree	0.8911	0.2402	0.9637	0.1575	0.8286	0.5064	0.8095
Random Forest	0.8863	0.3901	0.9238	0.3203	0.8518	0.4986	0.8084
Gradient Boosting	0.8966	0.3045	0.9652	0.2060	0.8371	0.5835	0.8200
XGBoost	0.8956	0.3336	0.9574	0.2363	0.8413	0.5675	0.8195
LogReg class_weight	0.8165	0.6039	0.6899	0.9998	0.9999	0.4326	0.7492
LogReg SMOTE	0.8495	0.5857	0.7706	0.8158	0.9465	0.4569	0.7792
Balanced RF	0.8181	0.6017	0.7417	0.8795	0.9630	0.4461	0.7490
Easy Ensemble	0.8163	0.6038	0.6897	0.9999	1.0000	0.4325	0.7490

Modelo	Observación	Interpretación
Decision Tree	recall muy bajo, lo que indica un sesgo muy fuerte hacia la clase mayoritaria.	El árbol simple sobreajusta la clase mayoritaria y es insuficiente para capturar la clase grave. *(Algoritmo propenso al sobreajuste)*
Random Forest	recall muy bajo, lo que indica un sesgo muy fuerte hacia la clase mayoritaria.	bagging ayuda a mejorar la detección de la clase minoritaria, pero todavía limitada
Gradient Boosting	sesgo hacia la clase mayoritaria.	boosting prioriza la clase mayoritaria, pero clase minoritaria sigue siendo un reto.
XGBoost	El modelo predice muy bien los casos leves y muchos casos graves no se detectan.	Es muy sensible a la clase mayoritaria; detecta bien los casos leves, pero falla en la minoritaria.
LogReg class_weight	Excelente para clase grave en recall	`class_weight='balanced'` **aumenta detección de la clase minoritaria**, sacrificando leve.
LogReg SMOTE	F1 y recall de clase grave mejoradas	Sobremuestreo clase minoritaria + modelo lineal. Mejora recall clase grave.
Balanced RF	Muy buen recall para clase grave	**especialmente diseñado para datasets desbalanceados**, detectan clase minoritaria mucho mejor, aunque clase mayoritaria pierde algo de precisión.
Easy Ensemble	Muy buen recall para clase grave	**especialmente diseñado para datasets desbalanceados**, detectan clase minoritaria mucho mejor, aunque clase mayoritaria pierde algo de precisión.



✅ Significado del heatmap:
- Verde: valores altos → buen rendimiento.
- Rojo: valores bajos → áreas a mejorar (principalmente clase minoritaria “Grave”).
- Las columnas de Recall Grave y F1 Grave muestran los puntos débiles de todos los modelos.
- Permite comparar rápidamente cómo cada modelo maneja la clase mayoritaria y la minoritaria, así como la métrica global.

Conclusiones globales

- *Todos los modelos predicen muy bien la clase **leve**, que es la mayoritaria.*
- *La clase **grave**, minoritaria, sigue siendo difícil de predecir; incluso modelos con ensambles (Random Forest, Gradient Boosting) solo logran mejorar parcialmente el recall.*
- *Random Forest es el modelo que **mejor equilibrio** muestra entre la detección de leves y graves, gracias a su enfoque ensemble.*
- ***La accuracy global no es representativa del desempeño real, debido al desbalance de clases.***

1. ***Desempeño por clase**:*
   - ***Clase leve (mayoritaria)**: todos los árboles y boosting logran F1 alto (~0.89-0.90).*
   - ***Clase grave (minoritaria)**: LogReg con class_weight/SMOTE y ensambles balanceados logran F1 ~0.60, mientras que árboles estándar y boosting obtienen F1 <0.40.*
2. ***Trade-off precisión-recall**:*
   - *Los modelos con balanceo (LogReg con class_weight, SMOTE, Balanced RF, Easy Ensemble) priorizan **recall de la clase minoritaria**, a costa de precisión en clase mayoritaria.*
   - *Los modelos tradicionales (Decision Tree, RF, Boosting) priorizan la clase mayoritaria, maximizando accuracy global, pero ignorando la minoritaria.*
3. ***Métricas globales**:*
   - *Accuracy más alto: Gradient Boosting (~0.82).*
   - ***Mejor detección clase grave: Easy Ensemble y LogReg class_weight (recall ~0.999).***
   - *F1 balanceado entre clases: LogReg SMOTE y Balanced RF.*

4. Validación cruzada

¿Qué es la validación cruzada?

La validación cruzada (Cross-Validation, CV) es una técnica utilizada en Machine Learning para evaluar el rendimiento de un modelo de manera más robusta y confiable.

En lugar de entrenar y evaluar el modelo con una única división del dataset (train/test), la validación cruzada divide los datos en varias submuestras (folds) y entrena/evalúa el modelo múltiples veces.

Esto permite:

- *Reducir el riesgo de overfitting.*
- *Obtener una estimación más estable y generalizable del rendimiento del modelo.*
- *Comparar modelos de forma más justa.*

💡 ***StratifiedKFold** = responde **"qué tan estable y confiable es mi modelo con mis datos"**.*

💡 ***GridSearchCV =** responde **"qué configuración de hiperparámetros es la mejor"**.*

1. ***StratifiedKFold CV (para Balanced RF y Easy Ensemble):***

- *Se utilizó la validación cruzada con **estratificación**, es decir, asegurando que la proporción de clases **(leve vs grave)** se mantuviera en cada fold.*
- *Esto es crítico en datasets desbalanceados como el nuestro, ya que evita que algunas divisiones de entrenamiento o validación queden sin suficientes ejemplos de la clase minoritaria.*
- *Configuración aplicada:*
  - *n_splits = 5, dividiendo los datos en 5 subconjuntos.*
  - *Cada modelo se entrena en 4 folds y se valida en el fold restante, repitiéndose el proceso hasta cubrir todos los folds.*
  - *Se calcularon las métricas (Accuracy, F1, AUC, etc.) en cada iteración y se promedió el resultado.*

🔎 *El* `BalancedRandomForest` *y el* `EasyEnsemble` *son modelos diseñados para manejar clases desbalanceadas, por lo que era importante evaluar su rendimiento en particiones* ***estratificadas***

para reflejar correctamente la proporción de accidentes leves y graves.

2. GridSearchCV (para Logistic Regression y Random Forest):

En estos modelos clásicos se aplicó Grid Search con validación cruzada, que consiste en:

1. *Definir un espacio de hiperparámetros a evaluar (profundidad de árboles, número de estimadores, regularización, etc.).*
2. *Entrenar el modelo con todas las combinaciones posibles de hiperparámetros.*
3. *Evaluar cada configuración con validación cruzada.*
4. *Seleccionar la combinación que maximiza una métrica objetivo (ej. F1-score).*

- *Configuración aplicada:*
  - *GridSearchCV con cv = 5 folds.*
  - *Métrica principal de optimización: F1-score, priorizando el equilibrio entre precisión y recall en la clase minoritaria.*

🔎*La* `LogisticRegression` *y el* `RandomForest` *son más sensibles a la elección de hiperparámetros. El uso de GridSearchCV permitió optimizar su rendimiento y compararlos de forma justa con los modelos balanceados.*

Conclusiones

- *La validación cruzada fue clave para evaluar de manera justa y robusta los modelos, evitando depender de una sola partición de datos.*
- *Los modelos tradicionales* `(Logistic Regression y Random Forest)` *mejoraron con* `GridSearchCV,` *pero siguen siendo menos efectivos en la predicción de la clase minoritaria (accidentes graves).*
- *Los modelos adaptados al desbalance* `(Balanced RF y Easy Ensemble)` *mostraron un rendimiento más equilibrado, con mejor recall y F1 para la clase grave, lo que es crítico en este caso de estudio.*
- *El* `Easy Ensemble` *destacó con el ROC-AUC más alto (0.846 en validación cruzada), mientras que el Balanced RF mostró un mejor equilibrio general entre métricas y estabilidad en los folds.*
- *En conclusión, la validación cruzada permitió confirmar que los modelos balanceados son más adecuados para este problema, y que la elección del* `Balanced Random Forest` *como modelo final está justificada por su consistencia, interpretabilidad y rendimiento competitivo*

5. Despliegue

Objetivos

El objetivo del despliegue es poner en producción el modelo de predicción de severidad de accidentes de tráfico mediante una API accesible y escalable. Esto permite que aplicaciones externas, dashboards o servicios de terceros consuman las predicciones en tiempo real a través de peticiones HTTP.

En otras palabras: ***llevar el modelo desde un entorno experimental (notebooks) a un servicio utilizable en producción.***

Workflow de Despliegue

1. ***Entrenamiento del modelo***
   - *El modelo de Machine Learning (Balanced Random Forest) se entrenó en Jupyter Notebook.*
   - *Se seleccionaron las mejores métricas (F1-score, ROC-AUC, Average Precision) para validar el rendimiento.*
   - *Se exportó el modelo final como archivo serializado (`best_model.pkl`) utilizando `joblib`.*
2. ***Preparación de la API***
   - *Se implementó un servicio con* ***FastAPI*** *en Python.*
   - *Se definió un esquema de entrada con `pydantic` (`CrashInput`) para garantizar que las variables ingresadas respeten el formato esperado.*
   - *Se cargó el modelo previamente entrenado en memoria al iniciar el servidor.*
3. ***Definición de Endpoints***
   - `/` ***(GET):*** *endpoint de bienvenida y verificación de estado de la API.*
   - `/predict` ***(POST):*** *endpoint principal que recibe las variables del accidente en formato JSON y devuelve la predicción de severidad (`0 = leve`, `1 = grave`) junto con la probabilidad asociada.*

Prueba de inferencia

Objetivo

Verificar el correcto funcionamiento del modelo desplegado mediante FastAPI, asegurando que recibe entradas en el formato esperado, procesa los datos y devuelve predicciones junto con probabilidades asociadas.

1. ***Pruebas de la API***
   - *Se utilizó la documentación automática de FastAPI (Swagger UI en `/docs`) para enviar ejemplos de entrada y verificar las predicciones.*
   - *El modelo `BalancedRandomForestClassifier` previamente entrenado y guardado fue cargado al iniciar la aplicación.*
   - *También se probaron peticiones manuales mediante herramientas como **Postman** o `curl`.*

2. ***Despliegue Local***
   - *Se levantó el servidor con `uvicorn`:*

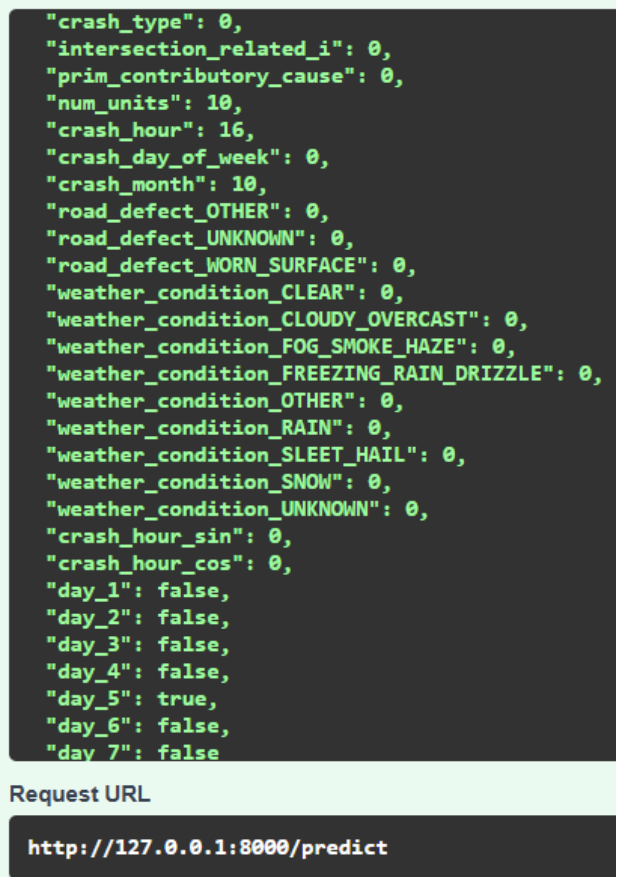
```
uvicorn API_Accident:app --reload
```

👉🏻***Conéctate a las API a través del siguiente enlace →http://127.0.0.1:8000/docs***

```
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [22348] using WatchFiles
INFO:     Started server process [35776]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

- *Se envió un ejemplo representativo al endpoint*

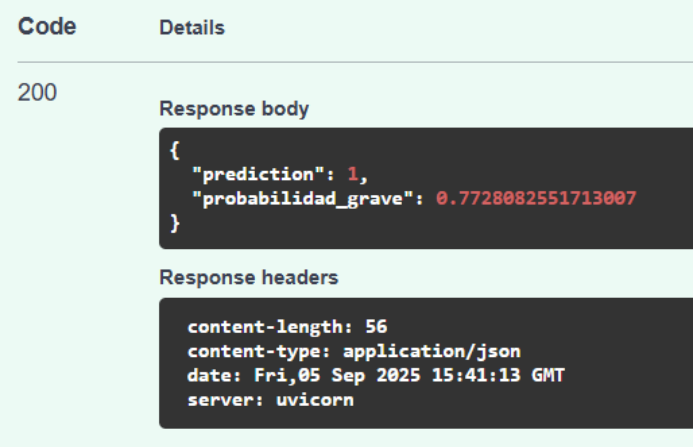
```
"day_of_week": 5, #"viernes"
"month": 10, #"Octubre"
"hour": 16,
"num_units": 10 # Nº de unidades involucradas en la colisión
```



- ***Procesamiento interno**:*
  - *La API transformó los datos de entrada en el formato requerido por el modelo.*
  - *El modelo calculó la predicción de severidad (`leve = 0`, `grave = 1`) y las probabilidades de pertenecer a cada clase.*

- ***Respuesta de la API:***

```
"prediction": "grave",
"probability": 0.77
```



Conclusiónes

- ***Predicción:*** *El modelo clasificó el caso como* ***grave*** *(clase 1).*
- ***Probabilidad asociada (0.77):*** *Existe un* ***77% de confianza*** *en que el accidente se clasifique como severo bajo las condiciones dadas (viernes de octubre, 16h, 10 unidades involucradas).*
- *Esto sugiere que los factores de* ***hora punta (tarde en día laboral)*** *y* ***alto número de unidades implicadas*** *son determinantes en la severidad de los accidentes.*
- *La prueba confirma que la API está funcionando correctamente y devuelve predicciones coherentes con el entrenamiento del modelo.*
- *El sistema puede ahora integrarse en un entorno de producción o extenderse con un frontend que permita a usuarios introducir datos y visualizar la predicción en tiempo real.*