

TRABAJO DE FIN DE MÁSTER

**AGENTE IA PREDICTIVO PARA LA
DETECCIÓN DE FRAUDE EN RECLAMOS
DE SEGUROS DE VEHÍCULOS**



UNIVERSIDAD COMPLUTENSE DE MADRID

Máster Data Science, Big Data & Business Analytics 2024-2025

Emma Arenas Villaverde
MADRID, 2025

ÍNDICE

1. CONTEXTO Y JUSTIFICACIÓN DEL PROYECTO.....	5
2. OBJETIVOS DEL PROYECTO.....	5
3. METODOLOGÍA.....	6
4. DESARROLLO DE LA SOLUCIÓN.....	7
4.1. SELECCIÓN Y ANÁLISIS DESCRIPTIVO DE LOS DATOS.....	7
4.2. PREPROCESAMIENTO.....	9
4.3. MAPEO SEMÁNTICO DE VARIABLES.....	11
4.4. MODELOS PREDICTIVOS.....	12
4.4.1. COMPARATIVA DE ALGORITMOS Y TÉCNICAS DE OPTIMIZACIÓN.....	12
4.4.2. ENSEMBLE FINAL DE XGBOOST, RANDOM FOREST, LOGISTIC REGRESSION Y LIGHTGBM.....	14
4.5. DESARROLLO DEL BACKEND INTELIGENTE.....	15
4.5.1. PIPELINE DE INGENIERÍA DE CARACTERÍSTICAS.....	15
4.5.2. CLASIFICADOR DE RIESGO Y GENERADOR DE RECOMENDACIONES.....	17
4.5.3. CAPA DE IA CONTEXTUAL (LLM + REGLAS).....	18
4.5.4. FUNCIÓN PRINCIPAL DE ORQUESTACIÓN.....	19
4.6. INTERFAZ DE USUARIO Y APP FRONTEND CON STREAMLIT.....	19
4.6.1. FORMULARIO DE ENTRADA Y VALIDACIÓN DE ERRORES.....	20
4.6.2. VISUALIZACIÓN DE RESULTADOS.....	21
4.6.3. IMPLEMENTACIÓN DEL HISTORIAL DE RECLAMOS.....	21
4.6.4. DISEÑO Y CONFIGURACIÓN GENERAL.....	22
4.7. DESPLIEGUE Y PUBLICACIÓN DE LA SOLUCIÓN EN ENTORNO PRODUCTIVO.....	22
5. EVALUACIÓN DEL SISTEMA.....	23
6. CONCLUSIONES.....	24
6.1. VALOR AGREGADO DEL PROYECTO.....	24
6.2. LIMITACIONES DEL MODELO Y POSIBLES MEJORAS.....	25
REFERENCIAS BIBLIOGRÁFICAS Y FUENTES.....	27
ANEXOS.....	28

1. CONTEXTO Y JUSTIFICACIÓN DEL PROYECTO

La industria de los seguros de vehículos desempeña un papel fundamental en la economía global, ya que proporciona una red de seguridad para los propietarios de vehículos y garantiza la cobertura de los daños derivados de accidentes, robos y otros imprevistos. Sin embargo, ha surgido un fenómeno preocupante que afecta tanto a las aseguradoras como a los clientes: **el fraude en los seguros de vehículos**. El fraude en los seguros de vehículos es una práctica que implica la **manipulación o tergiversación de la información relacionada con un reclamo para obtener una indemnización que no corresponde a la situación real**. Según estudios recientes, se estima que el fraude en el sector representa un porcentaje significativo de las pérdidas anuales de las compañías de seguros; algo que impacta directamente en sus resultados financieros y en sus márgenes de rentabilidad. Las aseguradoras deben dedicar mayores recursos a la investigación de los reclamos, lo que **incrementa los costos operativos y reduce la eficiencia de los procesos**. Además, el fraude afecta a la relación de confianza entre la aseguradora y el cliente, ya que el aumento en los costos derivados del fraude puede traducirse en **primas más altas** para los asegurados “honestos”. En consecuencia, la problemática representa una amenaza tanto para las aseguradoras como para los consumidores.

A lo largo del tiempo, las compañías han utilizado sistemas tradicionales basados en reglas y procesos manuales para identificar reclamaciones fraudulentas. Estos métodos, que dependen de la experiencia y el juicio de los analistas humanos, presentan varias limitaciones que los hacen ineficaces frente a la creciente sofisticación de las tácticas utilizadas por los defraudadores. En contraposición, **la inteligencia artificial y el *machine learning* son especialmente eficaces en la detección de fraudes** debido a su capacidad para aprender de los datos y adaptarse al contexto sin intervención humana directa. A través de modelos predictivos, se pueden anticipar los reclamos de fraude en base a patrones históricos y características que indican un alto riesgo de fraude. En este marco, el proyecto propuesto se justifica en primer lugar por la **necesidad de las aseguradoras de mejorar sus capacidades de detección de fraude**. Con los métodos tradicionales quedando obsoletos frente a las tácticas fraudulentas, la adopción de **modelos predictivos y de IA** es una clave para mantener la competitividad y la sostenibilidad del negocio. Por otro lado, la implementación de este tipo de soluciones contribuirá significativamente a **reducir los costos operativos asociados con el fraude**, mejorando al mismo tiempo la **eficiencia de los procesos de reclamación**. Al basarse en datos objetivos y patrones, la solución busca lograr una clasificación más precisa evitando que los clientes legítimos sean penalizados.

2. OBJETIVOS DEL PROYECTO

El objetivo principal de este Trabajo Fin de Máster es el desarrollo de una **herramienta inteligente que permita a las aseguradoras detectar posibles fraudes en los reclamos de seguros de vehículos** de forma rápida, eficaz y comprensible.

Los objetivos específicos que han guiado el desarrollo del proyecto son los siguientes:

1. **Diseñar una arquitectura modular y extensible** que integre componentes de análisis predictivo, lógica explicativa y presentación visual en una única aplicación cohesiva.
2. **Desarrollar un modelo predictivo supervisado** entrenado con datos históricos de siniestros, capaz de asignar un *score* de fraude a cada reclamo con base en múltiples variables relevantes
3. **Incorporar un sistema de clasificación de riesgo**, proporcionando un nivel de riesgo cualitativo (bajo, medio, alto) para cada caso, a fin de ofrecer una evaluación del grado de sospecha asociado a cada reclamo.
4. **Generar recomendaciones prácticas basadas en reglas de negocio para cada caso**, ayudando a los usuarios a tomar decisiones informadas y eficaces.
5. **Incorporar una capa de inteligencia artificial explicativa (LLM)** que genere descripciones contextuales del porqué de cada predicción, mejorando la transparencia y la capacidad de decisión del usuario final.
6. **Implementar una interfaz de usuario clara y accesible** con un diseño moderno, estéticamente cuidado y amigable, pensado para facilitar la interacción incluso a usuarios sin conocimientos técnicos.
7. **Desarrollar un sistema de validación de entradas**, con mensajes de error claros, validaciones lógicas, y control de datos inconsistentes o incompletos.
8. **Incorporar un historial de reclamos analizados**, con identificadores únicos, filtros por nombre y fecha, y capacidad de revisión posterior, contribuyendo a la trazabilidad de los casos (*audit trail*).
9. **Publicar la solución en la nube**, asegurando su accesibilidad desde cualquier dispositivo y eliminando barreras técnicas para su uso inmediato.

3. METODOLOGÍA

El desarrollo de este proyecto se basa en una **metodología iterativa, modular y centrada en el usuario final**, con el objetivo de construir una solución funcional y adaptada al entorno real de una aseguradora. Desde el inicio, se ha buscado combinar un enfoque técnico con una experiencia de usuario intuitiva asegurando que la herramienta sea accesible para los empleados.

El proceso comienza con el **análisis del contexto actual** en la detección de fraudes en las aseguradoras; identificando puntos críticos. A partir de este diagnóstico, el **desarrollo técnico** se lleva a cabo por fases. Primero, se entrena y valida el **modelo predictivo** utilizando un conjunto de datos históricos previamente analizado y preprocesado. Paralelamente, se diseña el **esquema del formulario** mediante un archivo `io_schema.json`, desde el cual se construye dinámicamente la entrada de datos en la interfaz. Posteriormente, se integra la **lógica del sistema** para procesar las predicciones y visualizar los resultados de forma clara. Además, se incorpora la posibilidad de generar una **explicación de IA** mediante el uso de una API externa, todo ello gestionado desde un mismo entorno visual, accesible y autoexplicativo.

La interfaz se desarrolla utilizando **Streamlit**, un *framework* ideal para este tipo de soluciones por su rapidez de desarrollo, su capacidad de **personalización** y su **accesibilidad** desde cualquier navegador. A nivel estético, se busca cuidar el **diseño visual**, integrando estilos CSS personalizados, iconografía clara y una estructura que guía al usuario en todo momento. Todo el flujo se diseña con foco en la **experiencia del usuario**, asegurando que cualquier empleado de una aseguradora pueda utilizar la herramienta sin conocimientos técnicos y obtener resultados comprensibles.

Una vez completado el desarrollo funcional, se pasa a una fase de **validación del sistema** mediante casos de prueba simulados, diseñados específicamente para representar distintos niveles de riesgo. Esta fase permite comprobar que el sistema reacciona adecuadamente ante entradas conflictivas, datos incoherentes o predicciones límite, demostrando su robustez y fiabilidad.

Por último, se procede al **despliegue del sistema en la nube** a través de **Streamlit Cloud**, lo que permite su **acceso universal sin necesidad de instalaciones locales**. Esto garantiza que cualquier usuario pueda utilizar la herramienta desde cualquier dispositivo, con tan solo acceder a una URL, favoreciendo la **escalabilidad y adoptabilidad de la solución**.

4. DESARROLLO DE LA SOLUCIÓN

4.1. SELECCIÓN Y ANÁLISIS DESCRIPTIVO DE LOS DATOS

Para el desarrollo de la solución se optó por utilizar el conjunto de datos público “**carclaims.csv**” ([véase Anexo 1](#)), el cual cuenta con licencia **CC0: Public Domain**. Este dataset es proporcionado por **Angoss Knowledge Seeker Software**, una herramienta ampliamente reconocida en el ámbito de la analítica avanzada, el análisis de patrones y el aprendizaje automático. La selección del conjunto de datos “**carclaims**” responde a criterios tanto de calidad como de relevancia para el análisis de fraude en seguros de vehículos. Como primer paso en el desarrollo del proyecto, se realizó un [análisis exploratorio y descriptivo](#) con el fin de comprender la estructura general del dataset, evaluar la coherencia y completitud de los datos, y detectar posibles anomalías o inconsistencias. Este proceso permitió validar su adecuación para los objetivos del Trabajo Fin de Máster, así como identificar tendencias iniciales y relaciones entre variables que resultan prometedoras para la detección automatizada de fraude.

El dataset no solo incluye un volumen significativo de registros -un total de **15.420 casos de reclamaciones**- sino que también presenta una gran diversidad de variables que abarcan múltiples dimensiones relevantes para este tipo de análisis. Concretamente, el conjunto consta de **33 columnas**, de las cuales 32 representan **variables predictoras** y 1 corresponde a la **variable objetivo**; denominada “**FraudFound_P**”. Esta última indica si la reclamación fue **fraudulenta (1)** o **legítima (0)**. La variedad de variables disponibles y sus tipos ([véase Anexo 2](#)) facilita el entrenamiento de modelos predictivos, pero también la identificación de patrones de comportamiento que podrían estar relacionados con el fraude.

Se procedió a analizar el número de **valores únicos** presentes en cada variable, así como la naturaleza de dichos valores ([véase Anexo 3](#)), con el objetivo de detectar posibles inconsistencias, errores de codificación, redundancias o problemas que pudieran afectar negativamente al modelado. A partir de este análisis exploratorio preliminar, se extrajeron las siguientes observaciones:

Variables categóricas simples

- **AccidentArea, Sex, Fault, PoliceReportFiled, WitnessPresent, AgentType**

Variables con valores erróneos

- **DayOfWeekClaimed** → Se detectaron 8 valores únicos, entre los que aparece un 0 inválido. Se opta por eliminar en el preprocesado, ya que solo puede haber 7 días en la semana.
- **MonthClaimed** → Se detectaron 13 valores únicos, entre los que aparece un 0 inválido. Se opta por eliminar en el preprocesado, ya que solo puede haber 12 meses.
- **Age** → Se detectaron múltiples registros con valor 0, lo cual no es válido como edad. Se opta por imputar en el preprocesado, ya que eliminarlos implicaría una pérdida de información.

Variables redundantes o poco informativas

- **PolicyType** → Combinación de **VehicleCategory** y **BasePolicy**. Se opta por evaluar su redundancia.
- **Age** y **AgeOfPolicyHolder** → Representan el mismo concepto. Dado que **Age** tiene numerosos ceros, se opta por eliminar en el preprocesado y conservar **AgeOfPolicyHolder**.
- **PolicyNumber** y **RepNumber** → Son identificadores únicos sin utilidad predictiva; se opta por eliminar en el preprocesado.
- **Year** → Variable temporal que no se puede generalizar a futuros ejercicios. Se opta por eliminar en el preprocesado, ya que no se podrá usar como predictor.

Variables con rangos categóricos a transformar

Estas variables contienen valores en forma de rangos textuales (por ejemplo: 15 to 30) y deben ser transformadas en el preprocesado en valores numéricos utilizando el valor medio del rango:

- **VehiclePrice, Days_Policy_Accident, Days_Policy_Claim, AgeOfVehicle, NumberOfCars, PastNumberOfClaims, AgeOfPolicyHolder, NumberOfSupplements, AddressChange_Claim**

Variables sin clarificación contextual

- **DriverRating** → No se especifica el criterio de puntuación ni su escala. Dada la falta de información contextual, se opta por eliminar en el preprocesado.

Variables categóricas a codificar

Estas variables categóricas contienen múltiples niveles y requieren codificación antes de ser utilizadas en modelos de aprendizaje automático:

- **Month, DayOfWeek, Make, DayOfWeekClaimed, MonthClaimed, MaritalStatus, VehicleCategory, BasePolicy, AddressChange_Claim**

Problema de desbalance en la variable objetivo

- **FraudFound_P** → Se observa un importante desbalance en la variable objetivo, con únicamente un ~6% de casos etiquetados como fraude. Se prevé la aplicación de técnicas de sobremuestreo durante la fase de modelado para corregir este desequilibrio.

A continuación, se procedió a realizar un análisis de los **valores atípicos y la distribución estadística** de las variables. Para ello, se empleó la función `describe()` aplicada tanto a las variables numéricas como categóricas. En el caso de las **variables numéricas** ([véase Anexo 4](#)), se extrajeron estadísticas clave como la media, desviación estándar (*std*), mínimo, máximo, y los percentiles 25%, 50% y 75%. Además, se calculó el **coeficiente de asimetría (skewness)** con el fin de identificar posibles distribuciones sesgadas que requieran transformaciones para mejorar la eficiencia de los algoritmos de aprendizaje automático. Por su parte, en el caso de las **variables categóricas** ([véase Anexo 5](#)), se analizaron los valores más frecuentes (*top*) y la frecuencia absoluta del valor más común (*freq*), con el objetivo de detectar distribuciones extremadamente desbalanceadas o niveles poco informativos.

Posteriormente, se llevó a cabo un análisis exhaustivo de los **valores faltantes** (valores NaN explícitos) en el conjunto de datos. No se detectaron valores faltantes en ninguna de las variables, lo cual constituye un aspecto positivo, ya que evita la necesidad de imputaciones adicionales.

Por otro lado, se realizó un análisis de **colinealidad y correlación entre variables**. El objetivo fue identificar tanto redundancias como relaciones significativas con la variable objetivo que pudieran influir en la selección de características y en la estrategia de modelado. En el caso de las **variables numéricas** ([véase Anexo 6](#)), se utilizó un **mapa de calor de correlaciones (heatmap)** con degradado de color para identificar visualmente relaciones destacadas entre pares de variables, así como una **clasificación de correlaciones absolutas respecto a la variable objetivo**, ordenadas de mayor a menor. Ninguna variable numérica presentó una correlación superior a 0.03 con la variable objetivo, por lo que el poder predictivo del modelo deberá apoyarse en **combinaciones de variables y patrones no lineales**. Para las **variables categóricas** ([véase Anexo 7](#)), se aplicó la **V de Cramer corregida por sesgo**, utilizando una función personalizada basada en `chi2_contingency` de `scipy.stats`. Este coeficiente permite cuantificar la asociación entre dos variables categóricas, en este caso entre cada predictor categórico y la variable objetivo. Además, se construyó una **matriz de redundancia** aplicando la V de Cramér a cada par de variables categóricas, lo que permitió detectar colinealidades excesivas entre ellas. Las variables categóricas presentaron coeficientes V de Cramér más altos que las numéricas, lo que las posiciona como candidatas más relevantes para el modelo. Por otro lado, “PolicyType” reflejó una **colinealidad casi perfecta** con “BasePolicy” y “VehicleCategory”, por lo que se decidió eliminar para evitar duplicidad de información.

Finalmente, se realizó una **descomposición de la variable objetivo por subgrupos predictivos** con el objetivo de explorar visualmente la distribución del fraude en función de distintas variables ([véase Anexo 8](#)). Esta técnica sirvió como una primera aproximación visual para identificar posibles **segmentos de alto riesgo** de fraude, así como para evaluar el **poder discriminativo preliminar** de cada variable. Para ello, se aplicó una función personalizada que calcula, para cada variable, la **tasa de fraude** como el porcentaje de registros etiquetados como fraudulentos, y crea una gráfica de barras con cada una.

4.2. PREPROCESAMIENTO

Tras el análisis descriptivo de los datos, se pasó a la realización de varias [transformaciones y limpiezas en los datos](#) para mejorar la calidad del conjunto y prepararlo adecuadamente para la modelización.

El primer paso del preprocesado consistió en la **limpieza de los datos**, específicamente en la eliminación de filas con valores erróneos que podrían distorsionar el análisis y los resultados del modelo. Durante el análisis descriptivo, se observó que las columnas “**DayOfWeekClaimed**” y “**MonthClaimed**” contenían valores de “0”. Estos valores no son coherentes con la naturaleza de los datos, y dado que solo había un caso en cada columna, se decidió que imputarlos podría introducir ruido innecesario y distorsionar las relaciones entre las variables. Por lo tanto, se optó por eliminar estas filas del conjunto de datos.

Una vez realizada la limpieza de los datos, se procedió a eliminar **variables que no aportan valor significativo para el modelo** y que podrían contribuir a la sobrecarga computacional, además de incrementar el riesgo de sobreajuste:

- **PolicyNumber** y **RepNumber**: ya que contienen identificadores únicos para las pólizas y los representantes, respectivamente. Aunque pueden ser útiles en otros contextos, en este caso no aportan información significativa para la predicción.
- **PolicyType** y **Age**: por redundancia extrema frente a otras variables.
- **DriverRating**: debido a la falta de información contextual clara sobre su escala.
- **Year**: al ser una variable vinculada a un año específico, no tiene aplicabilidad futura, lo que limita su capacidad para generalizar en ejercicios posteriores.

En el siguiente paso del preprocesado, se abordaron las **variables categóricas que toman solo dos valores posibles**, las cuales fueron transformadas en variables binarias (0 y 1). Esta transformación es esencial para permitir que el modelo de *machine learning* trabaje de manera eficiente, ya que la mayoría de los algoritmos de aprendizaje automático no pueden manejar directamente variables categóricas no numéricas. Las variables fueron convertidas en variables booleanas ([véase Anexo 9](#)) utilizando un proceso de codificación con **LabelEncoder** de la librería scikit-learn, que asigna valores binarios de acuerdo con las categorías presentes en cada variable.

Por otro lado, se abordaron las **variables que presentan rangos en lugar de valores numéricos continuos**. Este tipo de variables no es adecuado para el modelado directo en muchos algoritmos de *machine learning*, ya que no proporcionan un valor numérico exacto que pueda ser utilizado en cálculos. Para resolver este problema, se decidió convertir las variables categóricas con rangos en valores medios que representarán el centro de cada intervalo ([véase Anexo 10](#)). Esta transformación permite que las variables sean tratadas como valores numéricos continuos.

Finalmente, se pasó a la **codificación de las variables categóricas**; que deben ser convertidas en un formato que los modelos de *machine learning* puedan procesar. En este caso, se decidió utilizar el **OrdinalEncoder** en lugar del tradicional One-Hot Encoding por varias razones:

1. **Número limitado de categorías:** estas variables tienen un número limitado de categorías que no son necesariamente independientes entre sí. Por ejemplo, los meses del año (Month), o el día de la semana (DayOfWeek), no deberían ser tratadas como variables independientes, ya que el orden tiene un impacto en la relación entre las categorías.
2. **Evitar la expansión de la dimensionalidad:** utilizar One-Hot Encoding en variables con muchas categorías (como Make, que tiene muchas marcas de vehículos diferentes) puede aumentar de manera considerable el número de características del conjunto de datos. Esto podría resultar en un modelo más complejo y un mayor riesgo de sobreajuste.

El proceso de codificación se realizó utilizando un **diccionario** manualmente creado para cada variable categórica ([véase Anexo 11](#)), el cual especifica el orden de las categorías en función de su relevancia o secuencia natural.

4.3. MAPEO SEMÁNTICO DE VARIABLES

Para complementar el preprocesado, se realizó un **mapeo semántico de variables** con el objetivo de preparar los datos para su integración en una aplicación que permita a los usuarios interactuar mediante un formulario. El mapeo semántico de variables consistió en la creación de un [esquema de entrada](#) (*IO schema*), que define cómo cada campo de entrada se vincula a los valores que ahora han sido transformados en números. Este esquema facilita la **conversión de la información entre la interfaz de usuario y el modelo de machine learning**, permitiendo que los datos introducidos sean correctamente procesados y utilizados para hacer predicciones. Las decisiones tomadas durante el diseño del esquema incluyen:

- Como las variables originales estaban en inglés, se realizó la **traducción al español** para facilitar la comprensión de los usuarios que interactuarán con la aplicación. Este proceso se llevó a cabo utilizando el parámetro **“original”**, que indica cuál es el nombre original de la variable en inglés.
- Además de la traducción, se especificaron los **tipos de variables** utilizando el parámetro **“type”**. Esta información es crucial para que la aplicación maneje correctamente los datos introducidos por el usuario.
 - Para variables numéricas como la edad del asegurado o el precio del vehículo, se asignó el **tipo *numeric***.
 - Para variables categóricas como la marca del vehículo o la zona del accidente, se asignó el **tipo *categorical***. Este tipo indica que las opciones disponibles son categorías predeterminadas, y cada una de ellas se asigna a su número correspondiente para que el modelo lo pueda entender.
 - En algunos casos, fue necesario agregar **opciones especiales** que permiten manejar situaciones en las que los usuarios no pueden proporcionar una respuesta exacta o cuando el valor es desconocido. Esto se logró utilizando el **valor “-1”** en ciertas variables; el modelo, al encontrar ese valor, trata este caso de manera especial, asumiendo que el dato no es conocido y ajustando el proceso de predicción en consecuencia.

- En el caso de la variable "Make", se añadió la opción "OTRA".
- Similarmente, en la variable "Deductible", se incluyó la opción "SIN FRANQUICIA".
- Para facilitar la interacción con el usuario, se crearon dos variables de tipo *date* que permiten extraer de forma sencilla las fechas del accidente y de la reclamación. Estas fechas se almacenan en las variables "**FECHA DEL ACCIDENTE**" y "**FECHA DE LA RECLAMACION**". A partir de estas, se generan automáticamente en el *backend* otras variables, como el mes, la semana del mes y el día de la semana, tanto para el accidente como para la reclamación.
- Además, se introdujo una nueva variable llamada "**FECHA EN LA QUE SE EMITIO LA POLIZA**" de tipo *date*, que es crucial para calcular el tiempo transcurrido entre la emisión de la póliza y el accidente, así como entre la emisión de la póliza y la reclamación. Las variables "Days_Policy_Accident" y "Days_Policy_Claim" se derivan automáticamente en el *backend* restando la fecha de emisión a las fechas del accidente y la reclamación, respectivamente.

4.4. MODELOS PREDICTIVOS

Una vez completado el preprocesado de los datos, el siguiente paso fue la construcción de los [modelos predictivos](#). Para ello, primero se separaron las *features* de la variable objetivo, lo que permitió preparar los datos para ser utilizados en el entrenamiento del modelo. A continuación, se realizó una **división estratificada de los datos en conjuntos de entrenamiento y prueba** (*train/test*), con una proporción de **80% para entrenamiento y 20% para prueba**. Esta división estratificada asegura que la distribución de las clases en ambos conjuntos sea representativa de la distribución original de las clases en el conjunto de datos completo.

Una de las principales dificultades que se enfrenta en problemas de clasificación es el **desequilibrio en las clases**, donde una clase está subrepresentada en relación con la otra. Este desbalance puede afectar el rendimiento del modelo, ya que el modelo podría predecir de manera desproporcionada la clase mayoritaria. En este caso, para abordar el problema, se aplicó la técnica **SMOTE** (*Synthetic Minority Over-sampling Technique*), que genera **muestras sintéticas de la clase minoritaria**. El objetivo es mejorar la capacidad del modelo para aprender patrones representativos de la clase minoritaria, al proporcionar más ejemplos de esa clase en el conjunto de entrenamiento.

4.4.1. COMPARATIVA DE ALGORITMOS Y TÉCNICAS DE OPTIMIZACIÓN

En el análisis de clasificación, es fundamental realizar una **comparación inicial entre modelos** para obtener una visión general de cuál de ellos tiene el mejor rendimiento en términos de las métricas de evaluación más relevantes. Por ello, se comenzó con una **evaluación preliminar de diferentes algoritmos**, probando varios modelos y comparando sus desempeños.

Modelos probados:

- **DecisionTreeClassifier**: modelo basado en árboles de decisión, que es fácil de interpretar pero puede ser propenso al sobreajuste cuando los datos son complejos.

- **RandomForest**: un conjunto de árboles de decisión que mejora la precisión y reduce el riesgo de sobreajuste en comparación con un solo árbol de decisión.
- **BalancedRandomForestClassifier**: variante de RandomForest diseñada específicamente para manejar clases desbalanceadas, equilibrando las clases.
- **LogisticRegression**: modelo comúnmente utilizado para clasificación binaria, ideal para obtener resultados interpretables y simples.
- **XGBoost**: potente algoritmo de *boosting* conocido por su rendimiento sobresaliente en problemas de clasificación, especialmente en datos desbalanceados.
- **LightGBM**: un algoritmo basado en árboles de decisión optimizado para eficiencia y velocidad, adecuado para conjuntos de datos con muchas variables categóricas.

Métricas de evaluación:

- **Accuracy**: proporción de predicciones correctas sobre el total de predicciones. Es importante destacar que esta métrica puede ser engañosa en datasets desbalanceados.
- **Recall (Sensibilidad)**: capacidad del modelo para identificar correctamente las instancias de la clase positiva, fundamental en problemas donde la clase minoritaria es de interés (como la detección de fraudes).
- **Precision**: indica cuántas de las predicciones positivas son realmente correctas. Es importante en contextos donde los falsos positivos tienen un alto coste.
- **F1-Score**: la media armónica entre recall y precision, proporcionando un balance entre detectar instancias positivas y evitar falsos positivos.
- **MCC (Matthews Correlation Coefficient)**: métrica robusta que considera todos los aspectos del rendimiento del modelo, especialmente útil en problemas con clases desbalanceadas.

Además, se utilizó **validación cruzada** para obtener una evaluación más fiable del rendimiento, dividiendo los datos en múltiples subconjuntos y entrenando y validando el modelo varias veces.

Los resultados de esta comparación inicial ([véase Anexo 12](#)) presentaron hallazgos relevantes. A pesar de que modelos como **Random Forest** o **LightGBM** tuvieron un mayor accuracy en el conjunto de test (**> 0.93**), su **recall es extremadamente bajo**, lo cual es problemático en un contexto de detección de fraude. En este tipo de escenarios, el recall (capacidad del modelo para identificar correctamente los casos fraudulentos) es una métrica crítica, ya que los falsos negativos (casos de fraude no detectados) tienen un **coste significativo para las aseguradoras**. En cambio, **Logistic Regression**, aunque con menor accuracy, logró el **recall más alto del conjunto (55,1%)**, mostrando un mejor equilibrio entre sensibilidad y precisión frente a los casos minoritarios de fraude. Además, presentó un rendimiento competitivo en **test_f1** y **test_roc_auc**, lo que sugiere que tiene una mejor capacidad de generalización en un conjunto de datos desbalanceado.

Se optó por optimizar el modelo base de **Logistic Regression** utilizando varias técnicas. En primer lugar, se utilizó **GridSearchCV** para encontrar los **mejores parámetros del modelo**:

- **C**: controla el nivel de regularización.
- **penalty**: tipo de regularización a utilizar. Los valores posibles son l1, l2, y elasticnet.
- **solver**: el algoritmo utilizado para la optimización interna del modelo.
- **class_weight**: ajusta el impacto relativo de cada clase durante el entrenamiento.

Una vez optimizados los parámetros ([véase Anexo 13](#)), el siguiente paso fue ajustar el **umbral de decisión**. El modelo de Logistic Regression por defecto utiliza un umbral de 0.5 sobre la probabilidad estimada. Sin embargo, este umbral no siempre es el más adecuado, especialmente en problemas con clases desbalanceadas. En el proceso de ajuste, se utilizaron las probabilidades predichas por el modelo y se calcularon las métricas de **precision**, **recall** y **F1-Score** para diferentes umbrales. Posteriormente, se seleccionó el umbral ([véase Anexo 14](#)) que maximiza el **F1-Score**, ya que proporciona un buen equilibrio entre precisión y recall. Los resultados del modelo con el nuevo umbral ([véase Anexo 15](#)) reflejaron una mejora significativa en la capacidad del modelo para detectar fraudes; el **recall** del modelo optimizado aumentó a **62.7%**, comparado con el **55.1%** sin ajuste. Después, se pasó a la fase de **identificación y selección de las variables más importantes para la predicción del fraude**. Esto se realizó utilizando el modelo previamente ajustado, mediante la técnica **SelectFromModel**. Esta técnica permite seleccionar únicamente aquellas características cuyo peso (coeficiente) es superior al umbral definido. En este caso, el umbral se fijó en la mediana de los coeficientes, lo que asegura que se conserven las variables más relevantes. Aunque en los resultados del modelo con menos variables seleccionadas ([véase Anexo 16](#)) mejoró el recall de la clase minoritaria (fraude), el modelo con el conjunto completo de variables presentó un mejor rendimiento global en términos de F1-Score y accuracy para ambas clases. Este deterioro en el rendimiento al reducir las variables puede explicarse porque, en problemas complejos como la detección de fraude, **la señal predictiva no está concentrada en pocas variables**, sino que depende de la combinación de muchas características. En conclusión, mantener el conjunto completo de variables resultó ser la estrategia más efectiva para preservar la capacidad predictiva del modelo.

4.4.2. ENSEMBLE FINAL DE XGBOOST, RANDOM FOREST, LOGISTIC REGRESSION Y LIGHTGBM

Para mejorar aún más la capacidad predictiva del sistema, se implementó una **técnica de ensemble**, combinando las salidas de cuatro modelos: **Logistic Regression** (optimizada), **Random Forest**, **XGBoost** y **LightGBM**. El enfoque consistió en entrenar cada modelo individualmente sobre el conjunto de entrenamiento y luego calcular la probabilidad de clase para cada observación en el conjunto de test. Para ello, se asignaron **pesos** de la siguiente manera:

- $w_{lr} = 0.1$ (Logistic Regression)
- $w_{rf} = 0.1$ (Random Forest)
- $w_{xgb} = 0.3$ (XGBoost)
- $w_{lgbm} = 0.4$ (LightGBM)

Además, se utilizó un umbral de clasificación ajustado de **threshold = 0.18**, calibrado previamente para maximizar el recall de la clase minoritaria (fraudes). Este umbral fue elegido específicamente para **aumentar la sensibilidad del modelo hacia los casos de fraude** sin comprometer excesivamente la precisión.

Los resultados obtenidos del **ensemble** ([véase Anexo 17](#)) fueron los mejores en comparación con los demás modelos, por lo que se eligió como **modelo final**. Esta decisión se fundamenta en:

1. El modelo final logró detectar un gran número de reclamos fraudulentos, reduciendo significativamente los **falsos negativos a solo 51 casos**. En sistemas antifraude, esto es crucial, ya que el objetivo principal es minimizar los fraudes no detectados.
2. Aunque la precisión en fraudes es baja (**15.02%**), este comportamiento es común en la industria de seguros. Los sistemas antifraude tienden a priorizar el recall, **aceptando un volumen elevado de falsos positivos que luego se depuran en capas posteriores**.
3. El propósito del sistema no es emitir una decisión final automática, sino actuar como un **filtro inicial que apoye a los empleados en la identificación de reclamos sospechosos**. De este modo, el modelo sirve como una herramienta de priorización.
4. Aunque el número de falsos positivos es alto (**758**), estos representan casos que simplemente serán sometidos a revisión adicional. **El coste de revisar casos legítimos es menor que el riesgo económico de dejar escapar fraudes**.
5. En contextos de detección de fraude, el uso de accuracy como métrica principal puede ser engañoso. Dado que los fraudes representan un porcentaje muy pequeño del total de reclamos, un modelo que simplemente prediga “no fraude” para todos los casos obtendría una accuracy artificialmente alta. Por tanto, en este proyecto se han priorizado métricas más adecuadas, como **Recall, Precision y F1-score**.

A partir de aquí, se procedió a guardar los [artifacts](#) necesarios para el *backend*:

- **Modelos individuales:** Logistic Regression, Random Forest, XGBoost y LightGBM en archivos .pkl para facilitar su carga y uso en el *backend*.
- **Configuración del ensemble:** los pesos asignados a cada modelo y el *threshold*.
- **Explicaciones SHAP:** para interpretar cómo las variables influyen en las predicciones.
- **Lista de columnas esperadas:** que el modelo requiere para hacer predicciones.

4.5. DESARROLLO DEL BACKEND INTELIGENTE

Una vez que se completó el desarrollo del modelo predictivo y se aseguraron los *artifacts* necesarios, el siguiente paso consistió en la creación del [backend](#) que permita hacer funcionar la aplicación de manera eficiente. Este *backend* inteligente es responsable de gestionar tanto la entrada de datos como la ejecución de los modelos de predicción, todo ello de forma automatizada y escalable. Para ello, se diseñaron y programaron varias **funciones** que abordan distintos aspectos del proceso. Estas funciones se centralizan en una **función principal de orquestación**, que gestiona el flujo de trabajo de forma integral, integrando cada una de las tareas para ofrecer una solución operativa.

4.5.1. PIPELINE DE INGENIERÍA DE CARACTERÍSTICAS

Una de las partes más cruciales del *backend* es la programación del **pipeline de preprocesamiento de datos** ([véase Anexo 18](#)), que asegura que la entrada de información se ajuste al formato requerido por los modelos predictivos. Este *pipeline* se diseñó para recibir datos en bruto y transformarlos en un **formato limpio y estructurado**, listo para la posterior inferencia. Para lograr

esto, se utilizó el archivo **io_schema**, que define cómo debe estar estructurada la información de entrada y qué transformaciones son necesarias para adaptar los datos a las expectativas del modelo.

Una de las tareas clave dentro de este pipeline fue el **renombrado de columnas** para asegurar que las columnas de los datos de entrada coinciden con las que los modelos esperan. Para ello, se cargó el *artifact* **feature_cols**, que contiene las columnas requeridas por el modelo, y se diseñó un código para transformar los nombres de las columnas, basándose en la información del **io_schema**. El proceso se realizó de la siguiente manera: se creó un **mapa de correspondencias entre los nombres originales y los nuevos nombres**, y luego se aplicó este mapa a los datos para renombrar las columnas de manera consistente.

Después, se procedió a crear la iteración sobre cada columna del esquema para aplicar las **transformaciones necesarias según el tipo de datos**. A continuación, se explican los detalles de cada tipo de transformación:

- **Fechas:** las columnas de tipo *date* se convierten al formato *datetime* de Pandas para asegurar que sean interpretadas y puedan ser manipuladas en el *pipeline*.
- **Catóricas:** las columnas de tipo *categorical* son mapeadas a sus respectivas opciones definidas en el **io_schema**, asegurando que los valores de las variables catóricas coincidan con las opciones predefinidas.
- **Núéricas:** para las columnas de tipo *numeric*, se aplica una conversión a formato numérico. Además, se introdujeron varios ajustes adicionales:
 - Algunas de las columnas numéricas originalmente eran **catóricas de rangos**. Para este tipo de columnas, los valores numéricos se **aproximan a la media de los rangos definidos**. Por ejemplo: si el número de documentos relacionados con el accidente introducido por el usuario es de 9, pero el rango más alto con el que se entrenó el modelo es 7, el valor se redondea a 7. Este ajuste de redondeo al valor más cercano se realizó mediante una función que calcula el valor más cercano a un conjunto predefinido de valores. Para ello, se utilizó una **lista de posibles valores de referencia** y se aplicó un algoritmo que selecciona el valor más cercano.
 - Otro caso particular fue el "PRECIO DEL VEHICULO". Como el modelo se entrenó con datos en dólares, pero el usuario ingresará el precio en euros, se implementó una **conversión automática de euros a dólares** antes de continuar con el procesamiento. Esta conversión se hizo utilizando un tipo de cambio fijo.
- **Derivadas:** es decir, aquellas variables que se generan a partir de otras. En el **io_schema**, aquellas características que se marcaron como derivadas se calculan automáticamente en el *pipeline*, utilizando las **columnas originales como base** para su creación.
 - Para facilitar la interacción con el usuario, se definieron las columnas de tipo *date* "FECHA DEL ACCIDENTE" y "FECHA DE LA RECLAMACION". A partir de estas dos fechas, el *backend* genera automáticamente las variables que necesita el modelo, tales como el mes, el día de la semana y la semana del mes, tanto para el accidente como para la reclamación.

- Además, se incorporó una nueva columna llamada "FECHA EN LA QUE SE EMITIO LA POLIZA" que permite calcular las variables "Days_Policy_Accident" y "Days_Policy_Claim", las cuales representan el tiempo transcurrido entre la emisión de la póliza y el accidente, así como entre la emisión de la póliza y la reclamación. Estas variables derivadas se generan mediante la resta de las fechas correspondientes.

4.5.2. CLASIFICADOR DE RIESGO Y GENERADOR DE RECOMENDACIONES

El **clasificador de riesgo** tiene como objetivo proporcionar al usuario, en la futura aplicación, una clasificación clara y comprensible del nivel de riesgo asociado a cada reclamación. Esta clasificación se presenta como "**Bajo riesgo**", "**Riesgo medio**" o "**Alto riesgo**", entonces, en lugar de simplemente mostrar un *score* numérico sin contexto, la clasificación de riesgo permite al empleado saber a qué casos debe prestar más atención. Para determinar el nivel de riesgo, se estableció el siguiente criterio de clasificación mediante código ([véase Anexo 19](#)):

- **Bajo riesgo**: si el *score* de la reclamación es inferior al *threshold*.
- **Riesgo medio**: si el *score* se encuentra entre el umbral y el *threshold* + 0.1.
- **Alto riesgo**: si el *score* supera el umbral + 0.1.

La razón de utilizar el umbral (***threshold***) como punto de referencia es que permite establecer una línea base objetiva para determinar qué tan riesgoso es un caso en función de su puntuación (*score*). Así, no es necesario que los empleados interpreten una puntuación en términos absolutos, sino que se tiene un **marco de referencia claro**. Por otra parte, la decisión de usar un **margen de +0.1** sobre el umbral para diferenciar el riesgo medio del alto riesgo fue tomada con el objetivo de crear una transición suave entre los niveles de riesgo, pero sin que los casos se solapen. El rango de 0.1 puntos se estableció porque es un intervalo suficientemente grande como para indicar un cambio significativo en la probabilidad de riesgo sin ser tan amplio como para perder sensibilidad en la clasificación.

Además del clasificador de riesgo, se desarrolló un **generador de recomendaciones** que proporciona sugerencias automáticas a los empleados sobre cómo proceder con cada reclamación. Este sistema fue diseñado no solo para facilitar la toma de decisiones, sino también para asegurar que los empleados tengan en cuenta aspectos críticos y posibles indicios de fraude o situaciones que requieran una mayor investigación. Las recomendaciones se basan en un conjunto de **reglas de negocio** establecidas tras un análisis exhaustivo del mercado de seguros de vehículos. Esto se implementó mediante una función de código ([véase Anexo 20](#)) en la que se programaron las siguientes recomendaciones automáticas, basadas en las características de la reclamación:

- Cuando el usuario marca que **existen documentos suplementarios**, aparece la recomendación: "**Consultar los documentos suplementarios adjuntos al reclamo.**"
- Cuando el usuario indica que **existe un testigo**, se muestra la recomendación: "**Solicitar testimonio o contacto del testigo.**"
- Cuando se presenta un **informe policial**, se recomienda: "**Revisar el informe policial relacionado con el accidente.**"

- En todos los casos, el sistema genera la recomendación de: **"Confirmar la responsabilidad declarada por el asegurado."**
- Si el asegurado ha presentado más de **6 reclamaciones previas**, aparece la recomendación: **"Historial de reclamos elevado: revisar patrones o recurrencia."**
- Si el accidente ocurrió en los **primeros 30 días** después de contratar la póliza, la recomendación es: **"El accidente ocurrió poco después de contratar la póliza: revisar con atención."**
- Si el asegurado ha cambiado de domicilio recientemente, se muestra la recomendación: **"El asegurado cambió de domicilio recientemente: validar veracidad del cambio."**
- Si el **valor del vehículo** es superior a 65,500, el sistema recomienda: **"Vehículo de alto valor: considerar inspección más exhaustiva."**
- Si el **agente externo** gestionó la póliza, la recomendación es: **"Agente externo involucrado: revisar consistencia de la documentación."**
- Si el asegurado tiene **menos de 21 años**, se sugiere: **"Corroborar historial del asegurado por edad especialmente joven."**
- Si el vehículo involucrado es de la **categoría deportiva**, aparece la recomendación: **"Evaluar el contexto del accidente por tratarse de un vehículo deportivo."**
- Si la póliza es **básica o de cobertura limitada**, el sistema recomienda: **"Evaluar nivel de cobertura total de la póliza por posible incentivo a fraude."**
- Si **no hay respaldo documental**, el sistema sugiere: **"Falta total de respaldo documental: enviar perito o iniciar investigación formal."**
- Si no se identifican recomendaciones claras, se indica: **"No se identificaron recomendaciones automáticas. Evaluar manualmente."**

4.5.3. CAPA DE IA CONTEXTUAL (LLM + REGLAS)

Además del **score**, la clasificación de riesgo y las recomendaciones, se implementó una **capa adicional de IA generativa** para proporcionar una explicación más detallada y comprensible sobre el riesgo de cada reclamación. Esta capa está basada en un **modelo de lenguaje de gran escala (LLM)** combinado con **reglas específicas de negocio**, lo que permite generar explicaciones contextuales y fáciles de entender para el empleado del área de siniestros. El propósito de esta capa es transformar los resultados en una explicación más cercana y amigable, utilizando lenguaje natural para que el empleado pueda comprender rápidamente las razones detrás de cada recomendación.

La función que lo gestiona trabaja de la siguiente manera ([véase Anexo 21](#)):

1. **Inicialización de la API:** comienza utilizando la **API de OpenAI** para comunicarse con el **modelo GPT-4o-mini**. Para ello, verifica que la **API key** proporcionada por el usuario es correcta.
2. **Preparación de la entrada:** la información del reclamo que se quiere procesar se convierte en un formato legible para el modelo. Esto incluye los resultados del análisis, las recomendaciones generadas previamente y los detalles específicos del caso.

3. **Construcción del *prompt*:** el primer mensaje define el **rol del sistema**, es decir, el comportamiento que debe tener el modelo: un asistente experto en detección de fraudes en seguros de vehículos. Se estableció que el modelo debe explicar de manera sencilla la clasificación de riesgo y las recomendaciones, sin entrar en detalles técnicos sobre puntuaciones o algoritmos. Además, el modelo debe basarse en patrones comunes de riesgo y proporcionar una justificación comprensible.
4. **Solicitud de generación de la explicación:** el segundo mensaje es el **rol del usuario**, que le proporciona al modelo la información relevante del reclamo, las recomendaciones generadas y el nivel de riesgo. El modelo utiliza esta información para redactar una explicación clara y profesional.
5. **Manejo de la respuesta:** si el modelo genera una respuesta válida, esta es extraída y devuelta como la explicación final. Si no se obtiene contenido o si hay algún error durante la generación, se devuelve un mensaje de error apropiado.

4.5.4. FUNCIÓN PRINCIPAL DE ORQUESTACIÓN

Por último, la **función de orquestación** ([véase Anexo 22](#)) juega un papel fundamental en la integración de todos los componentes del sistema, pues realiza varias tareas clave de manera secuencial para procesar y evaluar la reclamación:

1. **Carga de *artifacts*:** el primer paso en la función de orquestación es la carga de los **modelos de predicción**, la **configuración del *ensemble***, el ***schema* de entrada** y las **columnas de características**. Esto se logra llamando a las funciones respectivamente.
2. **Preprocesamiento de datos:** después de cargar los *artifacts* se llama al ***pipeline* de preprocesado**. Este paso es crucial, ya que convierte los datos crudos en un formato adecuado para que los modelos de predicción puedan utilizarlos.
3. **Generación de predicciones:** con los datos preprocesados, se generan las **predicciones** utilizando los modelos cargados. El código itera sobre ellos y calcula las probabilidades de cada uno. Posteriormente, las predicciones de todos se combinan utilizando un **peso ponderado** definido en la configuración del *ensemble*. El resultado es un **score de fraude**.
4. **Clasificación de riesgo y recomendaciones:** una vez calculados los *scores*, se utiliza el **clasificador de riesgo** para categorizar el nivel de riesgo de la reclamación en **bajo**, **medio** o **alto**. El umbral de clasificación es determinado por el valor ***threshold*** cargado previamente en la configuración. Además, las **recomendaciones** se generan y proporcionan sugerencias específicas para el empleado.
5. **Explicación de la IA:** el proceso para generar la explicación en lenguaje natural se realiza en una función que se encarga de verificar la **API key** introducida, y luego pasa los resultados al **modelo GPT-4o-mini** para generar una explicación.
6. **Devolución de resultados:** finalmente, la función de orquestación devuelve un **diccionario** con los resultados, que incluyen el **score**, el **riesgo** y las **recomendaciones** para la reclamación. Además, si se tiene la API key, la explicación generada por la IA también se muestra.

4.6. INTERFAZ DE USUARIO Y APP FRONTEND CON STREAMLIT

Con el backend ya configurado y operativo, se procedió al desarrollo de la [interfaz de usuario \(UI\) y aplicación frontend](#) utilizando **Streamlit**, una tecnología que permite construir interfaces web de forma rápida y eficiente desde código Python. La elección de Streamlit se fundamenta en varias razones. En primer lugar, su enfoque está especialmente orientado a soluciones de análisis de datos y *machine learning*, lo que lo hace ideal para **proyectos donde se combinan modelos predictivos con interacción visual**. En segundo lugar, su **facilidad de uso y despliegue** permite crear aplicaciones funcionales, limpias y visualmente estéticas.

En esta línea, el objetivo principal de la interfaz es que funcione como una **herramienta práctica y útil** para los empleados de una aseguradora, permitiéndoles analizar un reclamo de forma rápida, objetiva y estandarizada. La experiencia de uso se basa en un flujo muy sencillo:

1. El empleado **introduce los datos del reclamante y del reclamo** a través de un formulario estructurado.
2. El sistema **devuelve en cuestión de segundos un análisis automático** que incluye:
 - a. Un score de fraude numérico.
 - b. Un nivel de riesgo cualitativo.
 - c. Recomendaciones prácticas.
 - d. Una explicación generada por IA contextual (si se dispone de la clave de API).

Este primer análisis actúa como un **filtro preliminar** que ayuda al empleado a ahorrar tiempo, identificar rápidamente casos sospechosos y enfocar mejor sus recursos en los reclamos más dudosos.

4.6.1. FORMULARIO DE ENTRADA Y VALIDACIÓN DE ERRORES

El primer paso dentro de la aplicación consiste en la introducción de los datos del reclamo por parte del usuario a través de un **formulario** ([véase Anexo 23](#)). Esta funcionalidad constituye el corazón de la interfaz, ya que es aquí donde el empleado de la aseguradora recoge toda la información necesaria sobre el incidente que desea analizar.

Al principio, la aplicación solicita al usuario que introduzca el **nombre del reclamador**. A partir de este dato, se genera automáticamente un **identificador único** para cada reclamo, combinando el nombre ingresado con la fecha y hora exacta del análisis (formato YYYYMMDDHHMMSS). Esto garantiza la trazabilidad de cada evaluación y permite su recuperación posterior desde el historial.

Luego, para el formulario, los campos se definieron en base al archivo **io_schema** en el *backend*:

- **Campos de tipo fecha:** se muestran como calendarios interactivos, donde el usuario puede seleccionar visualmente la fecha deseada.
- **Campos categóricos con dos opciones** (como “Hombre / Mujer”): se presentan mediante botones tipo radio horizontales.

- **Campos categóricos con múltiples opciones:** como menús desplegables. De forma predeterminada, estas listas muestran la opción “Selecciona una opción”, obligando al usuario a realizar una selección activa y evitando así errores por omisión.
- **Campos numéricos:**
 - El campo “PRECIO DEL VEHÍCULO” se representa con un **slider interactivo** con valores entre 500 y 100.000 euros, con incrementos de 500.
 - El resto de campos numéricos utilizan un **campo de entrada directa**, con restricciones mínimas (por ejemplo, la edad mínima permitida es 16 años), y permiten tanto la escritura manual como el uso de flechas incrementales).

Además, se incorporó un **sistema de validación de errores** en tiempo real, que detecta y muestra al usuario cualquier inconsistencia o falta de información antes de permitir el envío del formulario. Las validaciones implementadas incluyen:

- **Incongruencias temporales:**
 - La fecha del accidente no puede ser posterior a la fecha de la reclamación.
 - La fecha de emisión de la póliza no puede ser posterior ni a la fecha del accidente ni a la fecha de la reclamación.
- **Errores en campos categóricos:** si el usuario no selecciona ninguna opción válida en un campo desplegable (dejando la opción por defecto “Selecciona una opción”), se muestra un mensaje de error indicando: “Debe seleccionar una opción válida para [nombre del campo]”.
- **Restricciones de tipo de datos:** los campos numéricos no permiten el ingreso de letras ni valores negativos.

Para pasar a visualizar los resultados del reclamo una vez completado, se desplegó el botón “**Analizar reclamo**”, que lanza el proceso de análisis y cambia la aplicación a este modo.

4.6.2. VISUALIZACIÓN DE RESULTADOS

Una vez que el usuario completa y envía correctamente el formulario de entrada, la aplicación pasa automáticamente al **modo de análisis** ([véase Anexo 24](#)). En primer lugar, se muestra un **resumen de los datos del reclamo** introducidos previamente. Esta recapitulación cumple una doble función: por un lado, ofrece al empleado una confirmación visual de la información introducida y, por otro, garantiza que siempre quede constancia visible de los valores que fueron utilizados como base para el análisis. Además, la barra lateral incluye un **campo para introducir la clave de API** necesaria para activar la explicación generada por IA ([véase Anexo 25](#)). Este campo se presenta con formato cifrado (como una contraseña), aunque permite alternar su visualización si el usuario lo desea. En caso de que la clave introducida sea válida, la aplicación muestra la **explicación detallada mediante la capa de IA contextual**. Esta explicación ofrece mayor transparencia y ayuda al empleado a entender por qué el modelo había clasificado el reclamo de una determinada forma.

Finalmente, se incluyó un botón que permite al usuario **volver al formulario inicial**, facilitando la navegación en caso de que desee registrar un nuevo reclamo o corregir el anterior.

4.6.3. IMPLEMENTACIÓN DEL HISTORIAL DE RECLAMOS

Una de las funcionalidades clave para garantizar la **trazabilidad y el seguimiento de cada caso** es el **historial de reclamos** ([véase Anexo 26](#)), accesible en todo momento desde la barra lateral de la aplicación. Cada vez que se completa un análisis, el sistema guarda automáticamente la información del reclamo junto con sus resultados¹, utilizando el identificador único previamente generado. Esta funcionalidad cumple un papel como **mecanismo de *audit trail*** para facilitar la revisión posterior de los casos. Además, se implementó un sistema de **búsqueda avanzada dentro del historial**, con dos filtros clave:

- **Filtro por nombre del reclamador:** permite encontrar fácilmente reclamos relacionados con una persona específica.
- **Filtro por rango de fechas:** útil para acotar la búsqueda a reclamos realizados dentro de un período determinado.

Por último, se incorporó un botón que permite al usuario **eliminar todo el historial**. Este botón está protegido: solo aparece activo si hay reclamos registrados, y al hacer uso de él se restablece el sistema al estado inicial, eliminando todos los análisis previos.

4.6.4. DISEÑO Y CONFIGURACIÓN GENERAL

Para ofrecer una experiencia de usuario clara y profesional, se cuidó el **diseño visual de la aplicación** y su **configuración general**. Esta capa estética no solo mejora la usabilidad, sino que también refuerza la identidad visual de la herramienta. Para ello, el diseño general se especificó mediante un archivo de configuración ubicado en `.streamlit/config.toml`. Y, adicionalmente, se utilizó la función `st.set_page_config()` para personalizar aún más la interfaz; incluyendo el **título de la aplicación** (CarClaim AI), su **disposición en pantalla** y un **favicon** personalizado (un icono de un coche-robot diseñado específicamente para la herramienta). Además de todo esto, se añadió una **capa de personalización CSS** que modifica aspectos clave de la aplicación:

- Estilo de **inputs y campos**, adaptando el color del texto y los *placeholders*.
- Apariencia de **sliders, botones y menús desplegables**, armonizados con la paleta de colores de la aplicación.
- Modificación del diseño de la **barra lateral** para resaltar las secciones más importantes.

Por otro lado, en la parte superior de la interfaz y justo encima del formulario ([véase Anexo 27](#)), se incluyó una breve **presentación de la herramienta**, que explica en qué consiste, cómo funciona y qué pasos debe seguir el usuario para analizar un reclamo. Esta introducción está acompañada del coche-robot virtual, que actúa como presentador.

¹ Actualmente, este historial se implementa en modo demostrativo; funcionando únicamente durante la **sesión activa** del usuario. Aunque los datos se guardan de forma local en un archivo temporal, en un entorno real de producción esta funcionalidad estaría integrada con una **base de datos centralizada** vinculada a usuarios. En ese escenario, sería necesario implementar un sistema de credenciales de acceso y almacenamiento persistente, garantizando la trazabilidad a largo plazo.

4.7. DESPLIEGUE Y PUBLICACIÓN DE LA SOLUCIÓN EN ENTORNO PRODUCTIVO

Una vez desarrollada la herramienta, el siguiente paso fue su despliegue en un **entorno accesible y funcional para cualquier usuario**, sin necesidad de instalaciones locales ni conocimientos técnicos. Para ello, se utilizó **Streamlit Cloud**, una plataforma que permite alojar aplicaciones desarrolladas en Streamlit de forma gratuita y accesible desde cualquier navegador web. Este tipo de despliegue garantiza una **experiencia de uso inmediata y universal**, ya que los usuarios no necesitan instalar Python, gestionar entornos virtuales ni preocuparse por dependencias técnicas. Basta con acceder a la **URL pública** (<https://carclaim.streamlit.app/>) para empezar a utilizar la herramienta.

El proceso de publicación fue sencillo y se compuso de los siguientes pasos:

1. **Subida del repositorio a GitHub:** todo el código fuente, junto con los archivos necesarios para la ejecución de la aplicación, fue organizado y alojado en un repositorio público disponible en: https://github.com/emmaarenas/TFM_FraudeSeguros
2. **Enlace con Streamlit Cloud:** una vez en GitHub, se configuró el proyecto dentro de la plataforma de Streamlit Cloud, seleccionando el repositorio y rama deseados. La plataforma genera una **máquina virtual específica para la ejecución de la aplicación**, por lo que fue esencial incluir un archivo **requirements.txt** con todas las dependencias necesarias.
3. **Configuración automática y despliegue:** Streamlit Cloud se encarga de instalar las dependencias y lanzar la aplicación de forma automática. Cualquier cambio futuro en el código del repositorio puede ser sincronizado fácilmente.

Gracias a este enfoque, se consiguió una solución **completamente funcional y pública**, lista para ser utilizada por cualquier persona, desde cualquier dispositivo con acceso a Internet.

5. EVALUACIÓN DEL SISTEMA

Para garantizar la fiabilidad y robustez de la herramienta, se llevó a cabo una **evaluación exhaustiva del sistema** a través de numerosas pruebas, cubriendo tanto los posibles errores de uso como el comportamiento esperado en condiciones normales de operación. A continuación, se resumen los principales puntos validados durante esta fase:

- El sistema **detecta y muestra correctamente los mensajes de error** cuando se introduce información incompleta o inconsistente en el formulario:
 - Se muestran advertencias claras si las fechas introducidas no tienen coherencia temporal (por ejemplo, un accidente posterior a la reclamación), o si no se ha seleccionado una opción válida en campos categóricos.
 - Se impide la ejecución del análisis si no se ha introducido el nombre del reclamador.
- La **validación de la API key** funciona correctamente; si se introduce una clave no válida, el sistema no genera una explicación IA.
- El **tiempo de procesamiento** tras enviar el formulario es muy rápido, y la transición al modo análisis ocurre sin retardos perceptibles para el usuario.
- El **historial de reclamos** funciona correctamente, garantizando que:

- Cada reclamo tiene un **identificador único**, por lo que no pueden existir duplicados.
- El botón de **borrado de historial** funciona adecuadamente.
- La **búsqueda por nombre** filtra correctamente los resultados del historial.
- El **rango de fechas** permite localizar fácilmente reclamos según la fecha.

Además de estas validaciones técnicas, se ha incorporado una pequeña muestra de reclamos predefinidos para ilustrar el funcionamiento del sistema. Aunque en esta versión de la herramienta el historial funciona por sesión, se ha creado un archivo llamado [historial.json](#) que contiene **seis reclamos simulados**. Estos casos de prueba han sido diseñados a partir del **análisis exploratorio de datos (EDA)**, donde se identificaron patrones relevantes en la relación entre variables y probabilidad de fraude. Gracias a estos conocimientos, se generaron perfiles de usuarios ficticios con casuísticas variadas. Los resultados obtenidos en estos análisis confirman que:

- Las **recomendaciones** responden de forma coherente a los datos proporcionados.
- La **clasificación de riesgo** se realiza correctamente según los umbrales de decisión.
- La **IA contextual** ofrece explicaciones precisas, bien argumentadas y lógicas.

Casos de prueba incluidos:

- **Carlos García:** caso con múltiples factores sospechosos (accidente muy próximo a la contratación, ausencia de documentos, póliza a todo riesgo, sin testigos) → El sistema lo clasifica correctamente como *Alto riesgo*, y la IA detalla todos los aspectos críticos.
- **Fernando Martín:** accidente el mismo día de la emisión de la póliza, coche de alta gama e historial con múltiples reclamaciones → Otro caso de *Alto riesgo* correctamente detectado.
- **Pedro Pereira:** joven con varios reclamos previos, mudanza reciente, poca documentación y ausencia de informe policial → La herramienta lo identifica correctamente como *Alto riesgo*.
- **Carmen Sánchez:** reclamo con testigos, con solo una reclamación previa, aunque con poca documentación → se clasifica como *Bajo riesgo*, aunque con recomendaciones prudentes.
- **Patricia López:** perfil estable, sin reclamos previos, documentación completa y póliza gestionada internamente → Clasificada como *Bajo riesgo*, con una explicación clara y lógica.
- **Miriam Valero:** caso con accidente múltiple, informe policial, testigos y documentación completa → El sistema lo trata adecuadamente como *Bajo riesgo*.

6. CONCLUSIONES

6.1. VALOR AGREGADO DEL PROYECTO

Este proyecto representa una **aportación innovadora y de alto valor** en el ámbito de la **gestión de fraudes en seguros de vehículos**; combina técnicas avanzadas de inteligencia artificial, y lo hace a través de una interfaz accesible, comprensible y aplicable en entornos reales de trabajo. Uno de los aspectos más destacables es su capacidad para aprovechar la potencia de **modelos predictivos y de lenguaje (LLMs)** en una única plataforma integrada. Por un lado, el modelo de detección de fraude aporta un **análisis estadístico y automatizado de cada reclamo**, fundamentado en datos históricos y relaciones cuantificables. Por otro, el uso de modelos de lenguaje (LLM) permite generar **explicaciones contextuales**, transformando resultados complejos en narrativas útiles para la toma

de decisiones. Este enfoque híbrido resuelve uno de los principales retos en proyectos basados en IA: la **interpretabilidad**. La herramienta no se limita a clasificar reclamos como fraudulentos o legítimos, sino que explica el porqué.

Desde el punto de vista funcional, el proyecto ofrece una **solución de extremo a extremo**: desde la recogida de datos a través de un formulario, pasando por el análisis automatizado, hasta la generación de recomendaciones y la posibilidad de revisar el historial de reclamaciones procesadas. Todo ello se presenta en una **interfaz accesible desde cualquier dispositivo**, sin necesidad de instalaciones complejas. Este nivel de accesibilidad convierte a la herramienta en una solución especialmente valiosa para las empresas, obteniendo beneficios inmediatos:

- **Reducción del tiempo invertido por empleado** en el análisis inicial de cada reclamo.
- **Detección temprana de fraudes**, lo que se traduce en importantes **ahorros económicos**.
- Mayor **uniformidad en las evaluaciones**, reduciendo la subjetividad.
- **Centralización del conocimiento**; cada reclamo analizado queda registrado.

Además, el sistema ha sido validado a través de múltiples pruebas funcionales, demostrando una **robustez operativa alta**, una **correcta gestión de errores** y un **rendimiento en tiempo real**, que lo hacen apto para ser utilizado en entornos productivos sin modificaciones sustanciales. También es importante destacar la **versatilidad del enfoque técnico** adoptado. La separación entre la lógica de backend, la capa de IA explicativa y la interfaz de usuario permite que esta solución pueda **escalar fácilmente**: integrarse con sistemas internos de aseguradoras, ampliarse a otros tipos de seguros, ...

Por último, cabe señalar el **diseño centrado en el usuario**. La organización del formulario, los mensajes de error claros, la navegación simple entre reclamos y la presencia de un asistente virtual como interfaz de bienvenida refuerzan la sensación de estar utilizando una herramienta moderna, amigable y eficaz. En conjunto, este proyecto es una **propuesta real y práctica**, lista para ser escalada y generar un impacto positivo en las aseguradoras.

6.2. LIMITACIONES DEL MODELO Y POSIBLES MEJORAS

Aunque el sistema desarrollado presenta un alto grado de funcionalidad y utilidad práctica, existen algunas **limitaciones inherentes al estado actual del proyecto** que abren la puerta a importantes **mejoras evolutivas**, especialmente si se contempla su integración en un entorno productivo real:

1. **Entrenamiento con datos reales y enriquecidos**: la primera y más importante mejora consiste en permitir que la herramienta se alimente de bases de datos internas de una aseguradora real. Esto mejoraría la precisión del modelo, al reflejar los patrones específicos del entorno en el que se va a aplicar. También se podrían incorporar variables adicionales (datos de geolocalización, tipología del cliente, etc.) para enriquecer la predicción.
2. **Aprendizaje activo (*active learning*)**: se podría implementar un sistema en el que cada vez que un empleado marque un reclamo como realmente fraudulento o legítimo, esa información se guarde y se utilice para realimentar el modelo. Este mecanismo permitiría que la herramienta aprenda continuamente y se adapte a nuevos patrones de fraude emergentes.

3. **Control de versiones y trazabilidad avanzada (*audit trail* ampliado):** una mejora fundamental sería registrar no solo el resultado del análisis, sino también todo el proceso seguido: quién está llevando el caso, qué pasos se han tomado, cuál es el estado actual del reclamo, etc. Esto sería especialmente valioso en contextos legales/auditorías internas.
4. **Sistema de usuarios con inicio de sesión:** se podría desarrollar un sistema de autenticación y gestión de usuarios, donde cada empleado acceda con sus credenciales y tenga un historial personalizado de reclamos.
5. **Interfaz administrativa y sistema de revisión colaborativa:** ampliando la idea anterior, podría añadirse una interfaz para administradores, que permita revisar reclamos analizados por otros, añadir observaciones o aprobar decisiones antes de cerrar el caso.
6. **Módulo de métricas y rendimiento:** otra mejora posible sería incorporar un *dashboard* con estadísticas globales, mostrando el número de reclamos procesados, tasas de fraude estimadas, evolución histórica de los riesgos detectados, ... Esto convertiría a la herramienta en un sistema de apoyo también a la gestión estratégica.
7. **Ampliación del dominio de aplicación:** finalmente, el enfoque y arquitectura del sistema podrían extenderse a otros tipos de seguros (hogar, salud, viajes), adaptando el formulario, el modelo predictivo y las reglas de negocio a cada caso particular.

REFERENCIAS BIBLIOGRÁFICAS Y FUENTES

- (2019). IT y Machine Learning en Seguros: Aplicación práctica en Fraudes. *Fundación MAPFRE*.
https://documentacion.fundacionmapfre.org/documentacion/i18n%20media/group/1109460.d_o
- (2022). Datos sobre el fraude en el seguro español: Las reclamaciones fraudulentas al seguro son más frecuentes en automóviles y responsabilidad civil. *UNESPA*.
<https://www.unespa.es/main-files/uploads/2022/06/NdP-El-fraude-en-el-seguro-espanol-Datos-2021-FINAL.pdf>
- (2024). La inteligencia artificial en el sector asegurador. *Revista Consorsegueros*, 20, 5–10.
<https://www.conorseguerosdigital.com/almacen/pdf/numero-20-es.pdf>
- Badal Valero, E., Sanjuán Díaz, A., & Segura Gisbert, J. (2020). Algoritmos de machine learning para la detección del fraude en el seguro de automóviles. *Anales Del Instituto De Actuarios Españoles*, (26), 23–46. https://doi.org/10.26360/2020_2
- Domínguez Rodríguez, D. (2025). El aumento de precios y el creciente fraude en los seguros de coche en España: un panorama preocupante para los conductores. *El Diario de Madrid*.
<https://www.eldiariodemadrid.es/articulo/sociedad/aumento-precios-creciente-fraude-segueros-coche-espana-panorama-preocupante-conductores/20250104120713087140.html>
- Fuentes, J. (2008). El fraude en el seguro del automóvil. Trabajo de Fin de Máster, *Universidad de Barcelona*. <https://hdl.handle.net/2445/134984>
- Guan, L., Xu, J., Wang, S., Xing, X., Lin, L., Huang, H., Liu, P., & Lee, W. (2016). From Physical to Cyber: Escalating Protection for Personalized Auto Insurance. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 1-12.
<https://arxiv.org/abs/1609.02234>
- Schrijver, G., Kapoor Sarmah, D., & El-Hajj, M. (2024). Automobile Insurance Fraud Detection Using Data Mining: A Systematic Literature Review. *Journal of Risk and Financial Management*, 17(2), 1–21. <https://doi.org/10.3390/jrfm17020021>
- Bacorelle, J. (2024). Simular un golpe para obtener dinero es el fraude más común al seguro del coche. ABC.
<https://www.abc.es/motor/reportajes/simular-golpe-obtener-dinero-fraude-comun-seguro-20240206123553-nt.html>

ANEXOS

Anexo 1. Enlace a la fuente del dataset “carclaims.csv”

→ <https://www.kaggle.com/datasets/khusheekapoor/vehicle-insurance-fraud-detection>

Anexo 2. Descripción y tipo de las variables del conjunto de datos

Variable	Tipo	Descripción
Month	object	Mes en el que ocurrió el accidente.
WeekOfMonth	int64	Semana del mes en el que ocurrió el accidente.
DayOfWeek	object	Día de la semana en el que ocurrió el accidente.
Make	object	Fabricante del vehículo involucrado en el siniestro.
AccidentArea	object	Área donde ocurrió el accidente (urbana o rural).
DayOfWeekClaimed	object	Día de la semana en el que se procesó la reclamación.
MonthClaimed	object	Mes en el que se procesó la reclamación.
WeekOfMonthClaimed	int64	Semana del mes en la que se procesó la reclamación.
Sex	object	Género del asegurado.
MaritalStatus	object	Estado civil del asegurado.
Age	int64	Edad del asegurado.
Fault	object	Indica si el asegurado tuvo la culpa del accidente.
PolicyType	object	Tipo de póliza de seguro.
VehicleCategory	object	Categoría del vehículo.
VehiclePrice	object	Precio del vehículo.
FraudFound_P	int64	Variable objetivo: indica si se detectó fraude en la reclamación.
PolicyNumber	int64	Identificador único de la póliza de seguro.
RepNumber	int64	Identificador único del representante que gestionó la reclamación.
Deductible	int64	Deducible que debe pagar el asegurado antes de que la aseguradora cubra los costos restantes.
DriverRating	int64	Puntuación del conductor.
Days_Policy_Accident	object	Días desde la emisión de la póliza hasta el accidente.

Days_Policy_Claim	object	Días desde la emisión de la póliza hasta la reclamación.
PastNumberOfClaims	object	Número de reclamaciones anteriores del asegurado.
AgeOfVehicle	object	Antigüedad del vehículo.
AgeOfPolicyHolder	object	Edad del asegurado.
PoliceReportFiled	object	Indica si se presentó un informe policial.
WitnessPresent	object	Indica si hubo testigos presentes en el accidente.
AgentType	object	Tipo de agente que gestionó la póliza (interno o externo).
NumberOfSuppliments	object	Número de documentos o reclamaciones suplementarias relacionadas con el caso.
AddressChange_Claim	object	Indica si el asegurado cambió de domicilio en el momento de la reclamación.
NumberOfCars	object	Número de vehículos involucrados en el accidente.
Year	int64	Año en el que se realizó o procesó la reclamación.
BasePolicy	object	Tipo de póliza base.

Anexo 3. Valores únicos presentes en cada variable

Variable	Valores únicos	Valores
Month	12	{'Apr': 1280, 'Aug': 1127, 'Dec': 1285, 'Feb': 1273, 'Jan': 1285, 'Jul': 1284, 'Jun': 1286, 'Mar': 1279, 'May': 1283, 'Nov': 1283, 'Oct': 1282, 'Sep': 1283}
WeekOfMonth	5	{1: 3187, 2: 3558, 3: 3640, 4: 3398, 5: 1637}
DayOfWeek	7	{'Friday': 2445, 'Monday': 2616, 'Saturday': 1462, 'Sunday': 2293, 'Thursday': 2279, 'Tuesday': 2636, 'Wednesday': 2425}
Make	19	{'Accura': 472, 'BMW': 15, 'Chevrolet': 1681, 'Chrysler': 1565, 'Dodge': 3483, 'Ford': 1048, 'Honda': 3172, 'Hyundai': 1398, 'Kia': 1260, 'Lexus': 1501, 'Mazda': 2420, 'Mercedes': 2326, 'Nissan': 2145, 'Pontiac': 3104, 'Toyota': 2489, 'Volkswagen': 1392, 'Volvo': 445}
AccidentArea	2	{'Rural': 1598, 'Urban': 13822}

DayOfWeekClaimed	7	{'Friday': 2445, 'Monday': 2616, 'Saturday': 1462, 'Sunday': 2293, 'Thursday': 2279, 'Tuesday': 2636, 'Wednesday': 2425}
MonthClaimed	12	{'Apr': 1280, 'Aug': 1127, 'Dec': 1285, 'Feb': 1273, 'Jan': 1285, 'Jul': 1284, 'Jun': 1286, 'Mar': 1279, 'May': 1283, 'Nov': 1283, 'Oct': 1282, 'Sep': 1283}
WeekOfMonthClaimed	5	{1: 3187, 2: 3558, 3: 3640, 4: 3398, 5: 1637}
Sex	2	{'Female': 7790, 'Male': 7630}
MaritalStatus	2	{'Divorced': 5749, 'Married': 5502, 'Single': 4169}
Age	45	[int values]
Fault	2	{'No': 13510, 'Yes': 1910}
PolicyType	2	{'Corporate': 4959, 'Personal': 10461}
VehicleCategory	2	{'Commercial': 3559, 'Private': 11861}
VehiclePrice	2	{'High': 6164, 'Low': 9256}
FraudFound_P	2	{0: 14473, 1: 947}
PolicyNumber	15420	[unique values]
RepNumber	15420	[unique values]
Deductible	6	{100: 1282, 200: 1257, 300: 2545, 400: 4322, 500: 2761}
DriverRating	5	{1: 2320, 2: 2889, 3: 3410, 4: 2381, 5: 1282}
Days_Policy_Accident	30	[numeric range]
Days_Policy_Claim	30	[numeric range]
PastNumberOfClaims	6	{0: 7343, 1: 3686, 2: 396, 3: 62, 4: 3}
AgeOfVehicle	32	[numeric range]
AgeOfPolicyHolder	60	[numeric range]
PoliceReportFiled	2	{'No': 11002, 'Yes': 4418}
WitnessPresent	2	{'No': 8523, 'Yes': 6897}
AgentType	2	{'Internal': 9022, 'External': 6398}
NumberOfSuppliments	7	{0: 9614, 1: 3499, 2: 1298, 3: 458, 4: 45, 5: 4}

AddressChange_Claim	2	{'No': 11998, 'Yes': 3422}
NumberOfCars	6	{1: 14321, 2: 1034, 3: 60, 4: 4, 5: 1}
Year	7	{2015: 1284, 2016: 1282, 2017: 1279, 2018: 1286, 2019: 1283, 2020: 1285, 2021: 1282}
BasePolicy	2	{'Corporate': 4959, 'Personal': 10461}

Anexo 4. Análisis de la distribución estadística de las variables numéricas

- Estadísticas clave:

	WeekOfMonth	WeekOfMonthClaimed	Age	FraudFound_P	PolicyNumber	RepNumber	Deductible	DriverRating	Year
count	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000
mean	2.788586	2.693969	39.855707	0.059857	7710.500000	8.483268	407.704280	2.487808	1994.866472
std	1.287585	1.259115	13.492377	0.237230	4451.514911	4.599948	43.950998	1.119453	0.803313
min	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	300.000000	1.000000	1994.000000
25%	2.000000	2.000000	31.000000	0.000000	3855.750000	5.000000	400.000000	1.000000	1994.000000
50%	3.000000	3.000000	38.000000	0.000000	7710.500000	8.000000	400.000000	2.000000	1995.000000
75%	4.000000	4.000000	48.000000	0.000000	11565.250000	12.000000	400.000000	3.000000	1996.000000
max	5.000000	5.000000	80.000000	1.000000	15420.000000	16.000000	700.000000	4.000000	1996.000000

- Coefficiente de asimetría (*skewness*):

WeekOfMonth	0.115426
WeekOfMonthClaimed	0.158233
Age	0.152314
FraudFound_P	3.711164
PolicyNumber	0.000000
RepNumber	0.006628
Deductible	6.078803
DriverRating	0.009283
Year	0.245689

Anexo 5. Análisis de la distribución estadística de las variables categóricas

Variable	unique	top	freq
Month	12	Jan	1411
DayOfWeek	7	Monday	2616
Make	19	Pontiac	3837
AccidentArea	2	Urban	13822
DayOfWeekClaimed	8	Monday	3757
MonthClaimed	13	Jan	1446
Sex	2	Male	13000
MaritalStatus	4	Married	10625

Fault	2	Policy Holder	11230
PolicyType	9	Sedan - Collision	5584
VehicleCategory	5	Commercial	7047
VehiclePrice	3	Low	14324
Days_Policy_Accident	8	7 years	5807
Days_Policy_Claim	9	31 to 35	5593
PastNumberOfClaims	4	2 to 4	5485
AgeOfVehicle	5	7 years	15333
AgeOfPolicyHolder	5	31 to 35	15179
PoliceReportFiled	2	No	14992
WitnessPresent	2	No	15333
AgentType	2	External	15179
NumberOfSuppliments	4	none	7047
AddressChange_Claim	5	no change	14324
NumberOfCars	5	1 vehicle	14316
BasePolicy	3	Collision	5962

Anexo 6. Análisis de colinealidad y correlación de las variables numéricas

- Mapa de calor de correlaciones (heatmap) con degradado de color:

	WeekOfMonth	WeekOfMonthClaimed	Age	FraudFound_P	PolicyNumber	RepNumber	Deductible	DriverRating	Year
WeekOfMonth	1.000	0.275	-0.005	-0.012	-0.008	0.005	-0.004	-0.017	-0.004
WeekOfMonthClaimed	0.275	1.000	0.002	-0.006	0.012	0.009	0.005	-0.000	0.012
Age	-0.005	0.002	1.000	-0.030	0.026	-0.007	0.069	0.002	0.025
FraudFound_P	-0.012	-0.006	-0.030	1.000	-0.020	-0.008	0.017	0.007	-0.025
PolicyNumber	-0.008	0.012	0.026	-0.020	1.000	0.009	0.002	-0.012	0.937
RepNumber	0.005	0.009	-0.007	-0.008	0.009	1.000	0.001	0.011	0.009
Deductible	-0.004	0.005	0.069	0.017	0.002	0.001	1.000	0.004	-0.001
DriverRating	-0.017	-0.000	0.002	0.007	-0.012	0.011	0.004	1.000	-0.014
Year	-0.004	0.012	0.025	-0.025	0.937	0.009	-0.001	-0.014	1.000

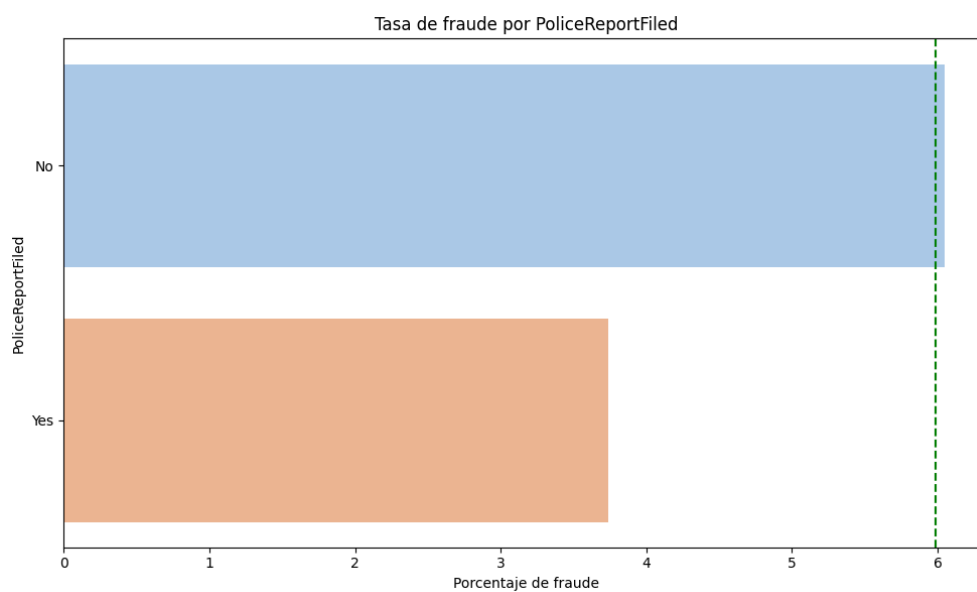
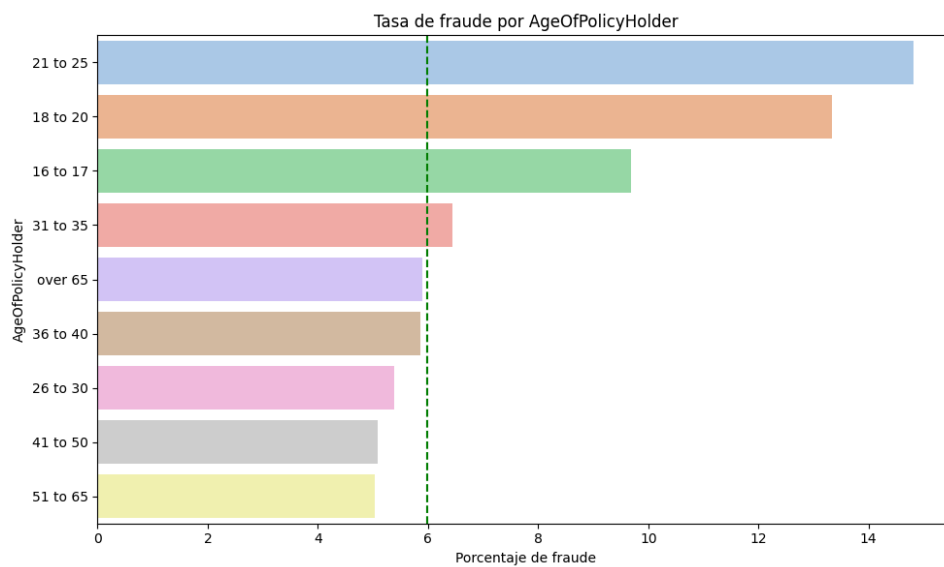
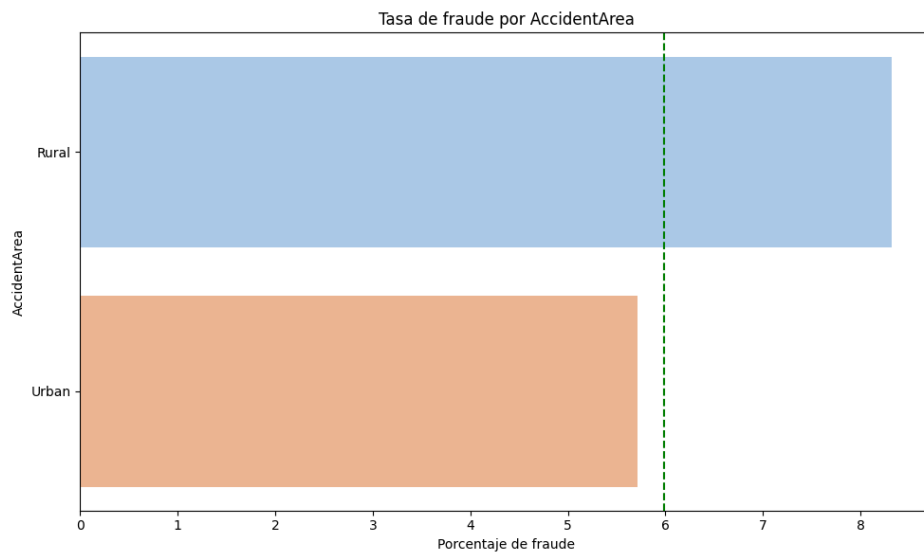
- Clasificación de correlaciones absolutas respecto a la variable objetivo:

FraudFound_P	
FraudFound_P	1.000000
Age	0.029741
Year	0.024760
PolicyNumber	0.020345
Deductible	0.017348
WeekOfMonth	0.011861
RepNumber	0.007551
DriverRating	0.007266
WeekOfMonthClaimed	0.005761

Anexo 7. Análisis de colinealidad y correlación de las variables categóricas

variable	cardinalidad	v_cramer
PolicyType	9	0.166880
BasePolicy	3	0.161237
VehicleCategory	3	0.136892
Fault	2	0.130839
AddressChange_Claim	5	0.080827
Deductible	4	0.067096
VehiclePrice	6	0.063803
PastNumberOfClaims	4	0.057230
Make	19	0.052072
MonthClaimed	13	0.044305
AgeOfPolicyHolder	9	0.040269
Month	12	0.034914
AccidentArea	2	0.032056
NumberOfSuppliments	4	0.031336
AgeOfVehicle	8	0.031116
Sex	2	0.028461
Days_Policy_Accident	5	0.022159
AgentType	2	0.020341
DayOfWeek	7	0.016406
PoliceReportFiled	2	0.012863
Days_Policy_Claim	4	0.011045
WeekOfMonth	5	0.000000
DayOfWeekClaimed	8	0.000000
MaritalStatus	4	0.000000
WitnessPresent	2	0.000000
NumberOfCars	5	0.000000
WeekOfMonthClaimed	5	0.000000

Anexo 8. Descomposición de la variable objetivo por subgrupos predictivos



* Las 32 gráficas están disponibles en el notebook del análisis EDA:

https://github.com/emmaarenas/TFM_FraudeSeguros/blob/master/Notebooks/EDA.ipynb

Anexo 9. Diccionario de encoders para la conversión a variables binarias

```
AccidentArea mapping: {'Rural': 0, 'Urban': 1}
Sex mapping: {'Female': 0, 'Male': 1}
Fault mapping: {'Policy Holder': 0, 'Third Party': 1}
PoliceReportFiled mapping: {'No': 0, 'Yes': 1}
WitnessPresent mapping: {'No': 0, 'Yes': 1}
AgentType mapping: {'External': 0, 'Internal': 1}
```

Anexo 10. Mapa de conversión de variables categóricas que presentan rangos

```
{ 'PastNumberOfClaims': {
  'none': 0,
  '1': 1,
  '2 to 4': 3,
  'more than 4': 5
}},
{ 'AgeOfPolicyHolder': {
  '16 to 17': 16.5,
  '18 to 20': 19,
  '21 to 25': 23,
  '26 to 30': 28,
  '31 to 35': 33,
  '36 to 40': 38,
  '41 to 50': 45.5,
  '51 to 65': 58,
  'over 65': 66
}},
{ 'NumberOfSupplements': {
  'none': 0,
  '1 to 2': 1.5,
  '3 to 5': 4,
  'more than 5': 6
}},
{ 'VehiclePrice': {
  '20000 to 29000': 24500,
  '30000 to 39000': 34500,
  '40000 to 59000': 49500,
  '60000 to 69000': 64500,
  'less than 20000': 15000,
  'more than 69000': 70000
}},
{ 'Days_Policy_Accident': {
  'none': 0,
  '1 to 7': 4,
  '8 to 15': 11.5,
  '15 to 30': 22.5,
  'more than 30': 35
}},
{ 'Days_Policy_Claim': {
  'none': 0,
  '8 to 15': 11.5,
  '15 to 30': 22.5,
  'more than 30': 35
}},
{ 'AgeOfVehicle': {
  '2 years': 2,
  '3 years': 3,
  '4 years': 4,
  '5 years': 5,
  '6 years': 6,
  '7 years': 7,
  'more than 7': 8,
  'new': 0.5
}},
{ 'NumberOfCars': {
  '1 vehicle': 1,
  '2 vehicles': 2,
  '3 to 4': 3.5,
  '5 to 8': 6.5,
  'more than 8': 9
}}
```

Anexo 11. Diccionario de orden secuencial para la codificación de las variables categóricas

```
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
dayofweek_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
make_order =
['Accura', 'BMW', 'Chevrolet', 'Dodge', 'Ferrari', 'Ford', 'Honda', 'Jaguar', 'Lexus', 'Mazda', 'Mecedes', 'Mercury', 'Nissan', 'Pontiac', 'Porche', 'Saab', 'Saturn', 'Toyota', 'VW']
marital_order = ['Divorced', 'Married', 'Single', 'Widow']
vehiclecat_order = ['Sedan', 'Sport', 'Utility']
basepolicy_order = ['All Perils', 'Collision', 'Liability']
addresschange_order = ['under 6 months', 'no change', '1 year', '2 to 3 years', '4 to 8 years']
```

Anexo 12. Resultados de la comparación inicial entre modelos

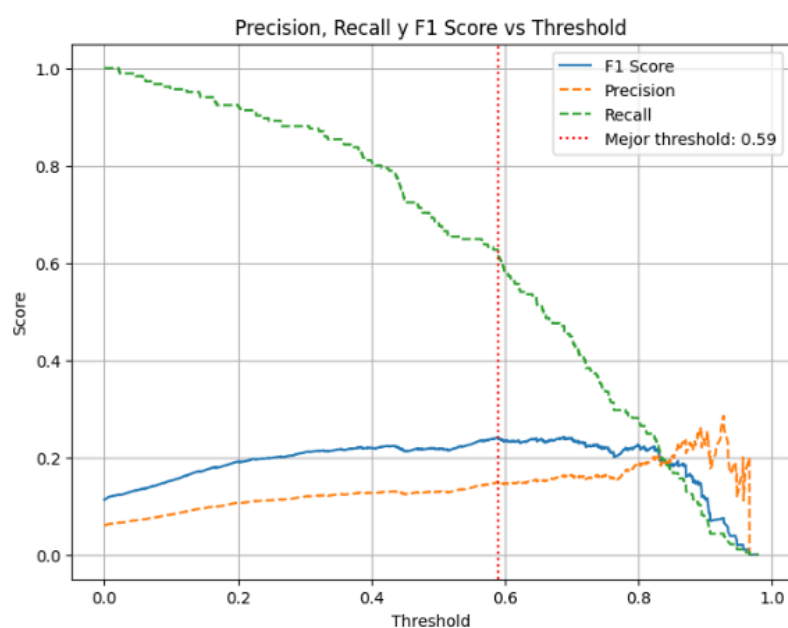
	model	run_time_min	test_accuracy	test_recall	test_precision	test_f1	test_mcc	test_roc_auc
	DecisionTree	0.08	0.887160	0.205405	0.158996	0.179245	0.120847	0.568036
	RandomForest	1.97	0.939040	0.048649	0.428571	0.087379	0.128522	0.522255
	BalancedRandomForest	3.14	0.938392	0.043243	0.380952	0.077670	0.111918	0.519379
	LogisticRegression	0.14	0.622568	0.551351	0.086221	0.149123	0.087152	0.589232
	XGBoost	0.05	0.939689	0.081081	0.483871	0.138889	0.179874	0.537781
	LightGBM	0.06	0.941310	0.043243	0.666667	0.081218	0.159678	0.520932

Anexo 13. Mejores parámetros obtenidos para Logistic Regression

- **C** = 100
- **penalty** = 'l1'
- **solver** = 'liblinear'
- **class_weight** = balanced

Anexo 14. Mejor threshold según F1 Score para Logistic Regression

Mejor threshold según F1 Score: 0.59



Anexo 15. Resultados del modelo Logistic Regression con el mejor umbral

[[2242 657]					
[69 116]]					
	precision	recall	f1-score	support	
0	0.9701	0.7734	0.8607	2899	
1	0.1501	0.6270	0.2422	185	
accuracy			0.7646	3084	
macro avg	0.5601	0.7002	0.5514	3084	
weighted avg	0.9209	0.7646	0.8236	3084	

Anexo 16. Resultados del modelo Logistic Regression con las mejores variables

[[2079 820]					
[59 126]]					
	precision	recall	f1-score	support	
0	0.9724	0.7171	0.8255	2899	
1	0.1332	0.6811	0.2228	185	
accuracy			0.7150	3084	
macro avg	0.5528	0.6991	0.5242	3084	
weighted avg	0.9221	0.7150	0.7893	3084	

Anexo 17. Resultados del modelo ensemble

[[2141 758]					
[51 134]]					
	precision	recall	f1-score	support	
0	0.9767	0.7385	0.8411	2899	
1	0.1502	0.7243	0.2488	185	
accuracy			0.7377	3084	
macro avg	0.5635	0.7314	0.5450	3084	
weighted avg	0.9272	0.7377	0.8056	3084	

Anexo 18. Función del pipeline de ingeniería de características

```
def preprocess_data(data, schema, feature_cols):
    col_map = {
        'PastNumberOfClaims': {'none': 0, '1': 1, '2 to 4': 3, 'more than 4': 5},
        'AgeOfPolicyHolder': {'16 to 17': 16.5, '18 to 20': 19, '21 to 25': 23, '26 to 30': 28, '31 to 35': 33, '36 to 40': 38, '41 to 50': 45.5, '51 to 65': 58, 'over 65': 66},
        'NumberOfSupplements': {'none': 0, '1 to 2': 1.5, '3 to 5': 4, 'more than 5': 6},
        'VehiclePrice': {'20000 to 29000': 24500, '30000 to 39000': 34500, '40000 to 59000': 49500, '60000 to 69000': 64500, 'less than 20000': 15000, 'more than 69000': 70000},
        'Days_Policy_Accident': {'none': 0, '1 to 7': 4, '8 to 15': 11.5, '15 to 30': 22.5, 'more than 30': 35},
        'Days_Policy_Claim': {'none': 0, '8 to 15': 11.5, '15 to 30': 22.5, 'more than 30': 35},
        'NumberOfCars': {'1 vehicle': 1, '2 vehicles': 2, '3 to 4': 3.5, '5 to 8': 6.5, 'more than 8': 9} ]
    # Renombrado de columnas según esquema
    rename_map = {}
    for feature, info in schema.items():
        if 'original' in info:
            rename_map[feature] = info['original']
    data.rename(columns=rename_map, inplace=True)
    renamed_schema = {}
    for feature, info in schema.items():
        if 'original' in info:
            renamed_schema[rename_map.get(feature, feature)] = info
    # Iteración sobre cada columna del esquema para aplicar transformaciones
    for feature, info in renamed_schema.items():
        print(f"Procesando la columna: {feature}, tipo en esquema: {info['type']}")
        # FECHAS
        if info['type'] == 'date':
            data[feature] = pd.to_datetime(data[feature], format='%d/%m/%Y')
        # CATEGÓRICAS
        elif info['type'] == 'categorical':
            data[feature] = data[feature].map(info['options'])
        # NUMÉRICAS
        elif info['type'] == 'numeric':
            data[feature] = pd.to_numeric(data[feature], errors='coerce')
            # Aproximación a la media si la variable estaba en rangos
            for col_dict in col_map:
                if feature in col_dict:
                    mapping_values = list(col_dict[feature].values())
                    # Conversión EUR → USD para el precio del vehículo
                    eur_to_usd = 1 / 0.86
                    if feature == 'VehiclePrice':
                        data[feature] = data[feature] * eur_to_usd
                    # Redondeo al más próximo
                    def round_to_closest(value):
                        arr = np.array(mapping_values)
                        idx = (np.abs(arr - value)).argmin()
                        return arr[idx]
                    data[feature] = data[feature].apply(round_to_closest)
        # DERIVADAS
        elif info.get('derived', False) or str(info['type']).lower() in ['derived', 'derived_numeric']:
            sources = info['from']
            if len(sources) == 1:
                col_source = rename_map.get(sources[0], sources[0])
                if col_source not in data.columns:
                    raise KeyError(f"Columna fuente '{col_source}' no encontrada en el DataFrame")
                data[col_source] = pd.to_datetime(data[col_source], errors='coerce')
                if 'options' in info:
                    if 'month' in feature.lower():
                        data[feature] = data[col_source].dt.strftime('%b').map(info['options'])
                    elif 'dayofweek' in feature.lower():
                        data[feature] = data[col_source].dt.day_name().map(info['options'])
                    else:
                        # WeekOfMonth
                        data[feature] = ((data[col_source].dt.day - 1) // 7 + 1).astype(int)
            elif len(sources) == 2:
                col1, col2 = sources
                if col1 not in data.columns or col2 not in data.columns:
                    print("Columnas fuente que faltan:", col1, col2)
                    continue
                data[col1] = pd.to_datetime(data[col1], format='%d/%m/%Y', errors="coerce", dayfirst=True)
                data[col2] = pd.to_datetime(data[col2], format='%d/%m/%Y', errors="coerce", dayfirst=True)
                data[feature] = (data[col2] - data[col1]).dt.days
    print(f"Columnas en los datos: {data.columns}")
    missing_cols = [col for col in feature_cols if col not in data.columns]
    if missing_cols:
        raise ValueError(f"Faltan columnas requeridas: {missing_cols}")
    else:
        data = data[feature_cols]
    return data
```

Anexo 19. Función del clasificador del nivel de riesgo de fraude

```
def clasificar_riesgo(score, threshold):
    if score < threshold:
        return 'Bajo riesgo'
    elif score < threshold + 0.1:
        return 'Riesgo medio'
    else:
        return 'Alto riesgo'
```

Anexo 20. Función del generador de recomendaciones

```
def generar_recomendaciones(data):
    recomendaciones = []
    for idx, row in data.iterrows():
        fila = []
        if row.get('NumberOfSuppliments', 0) > 0:
            fila.append("Consultar los documentos suplementarios adjuntos al reclamo.")
        if row.get('WitnessPresent', 0) == 1:
            fila.append("Solicitar testimonio o contacto del testigo.")
        if row.get('PoliceReportFiled', 0) == 1:
            fila.append("Revisar el informe policial relacionado con el accidente.")
        fila.append("Confirmar la responsabilidad declarada por el asegurado.")
        if row.get('PastNumberOfClaims', 0) > 6:
            fila.append("Historial de reclamos elevado: revisar patrones o recurrencia.")
        if row.get('Days_Policy_Accident', 999) < 30:
            fila.append("El accidente ocurrió poco después de contratar la póliza: revisar con atención.")
        if row.get('AddressChange_Claim', 1) in [0, 2, 3]:
            fila.append("El asegurado cambió de domicilio recientemente: validar veracidad del cambio.")
        if row.get('VehiclePrice', 0) > 65500:
            fila.append("Vehículo de alto valor: considerar inspección más exhaustiva.")
        if row.get('AgentType', 1) == 0:
            fila.append("Agente externo involucrado: revisar consistencia de la documentación.")
        if row.get('AgeOfPolicyHolder', 100) < 21:
            fila.append("Corroborar historial del asegurado por edad especialmente joven.")
        if row.get('VehicleCategory', -1) == 1:
            fila.append("Evaluar el contexto del accidente por tratarse de un vehículo deportivo.")
        if row.get('BasePolicy', -1) == 0:
            fila.append("Evaluar nivel de cobertura total de la póliza por posible incentivo a fraude.")
        if (
            row.get('WitnessPresent', 0) == 0 and
            row.get('PoliceReportFiled', 0) == 0 and
            row.get('NumberOfSuppliments', 0) == 0
        ):
            fila.append("Falta total de respaldo documental: enviar perito o iniciar investigación formal.")
        if not fila:
            fila.append("No se identificaron recomendaciones automáticas. Evaluar manualmente.")
        recomendaciones.append(fila)
    return recomendaciones
```

Anexo 21. Función del generador de explicación IA contextual

```
def generar_explicacion_llm(resultado, entrada, api_key):
    import json
    client = OpenAI(api_key=api_key)
    # Serializar entrada como texto legible
    entrada_legible = json.dumps({k: str(v) for k, v in entrada.items()}, ensure_ascii=False)
    messages = list(ChatCompletionMessageParam = [
        {
            "role": "system",
            "content": (
                "Eres un asistente experto en detección de fraudes en seguros de vehículos. "
                "Tu tarea es explicar de manera clara y sencilla, pero profesional, por qué un reclamo fue clasificado con un nivel de riesgo determinado, "
                "basándote únicamente en la información proporcionada del caso y en las recomendaciones automáticas generadas. "
                "El texto debe estar dirigido a un empleado del área de siniestros, sin conocimientos técnicos. "
                "No menciones puntuaciones, modelos, scores ni variables técnicas. "
                "En su lugar, elabora una explicación comprensible que justifique el riesgo percibido y las acciones sugeridas. "
                "No debes saludar al empleado ni referirte a él directamente. "
                "Además, nunca dejes la explicación incompleta. Debes garantizar que la explicación esté totalmente terminada, "
                "sin dejar dudas o puntos sin resolver."
                "Para la explicación, puedes basarte en patrones comunes de riesgo. Algunos ejemplos incluyen:\n"
                "- Es sospechoso si no hay testigos del accidente.\n"
                "- Es sospechoso si no se presentó informe policial.\n"
                "- Es sospechoso si no hay documentos adjuntos.\n"
                "- Es sospechoso si hay muchos vehículos involucrados.\n"
                "- Es sospechoso si el número de coches involucrados es 0 o no se especifica.\n"
                "- También pueden influir reclamos pasados, valor del vehículo, edad del asegurado, etc.\n"
            )
        },
        {
            "role": "user",
            "content": f"""
Nivel de riesgo del reclamo: {resultado['riesgo']}
Recomendaciones automáticas: {', '.join(resultado['recomendaciones'])}
Información del reclamo: {entrada_legible}
Redacta una explicación sencilla, clara y profesional que justifique el nivel de riesgo, tomando en cuenta las recomendaciones. La respuesta debe estar completa y no quedarte NUNCA a medias.
"""
        }
    ])
    try:
        response = client.chat.completions.create(
            model="gpt-4o-mini",
            messages=messages,
            max_tokens=400
        )
        # Validar si hay contenido real
        if response.choices and response.choices[0].message and response.choices[0].message.content.strip():
            return response.choices[0].message.content.strip()
        else:
            return "⚠ El modelo no devolvió contenido."
    except Exception as e:
        return f"❌ Error al generar explicación IA: {e}"
```

Anexo 22. Función principal del servicio

```
def model_service(data):
    # Llamada a las cargaas
    models = load_models()
    config = load_ensemble_config()
    schema = load_io_schema()
    feature_cols = load_feature_cols()
    # Preprocesado
    data= preprocess_data(data, schema, feature_cols)
    print("\nDatos preprocesados:")
    for col, val in data.iloc[0].items():
        print(f"{col}: {val}")
    # Predicciones
    preds = {}
    for model_name, model in models.items():
        preds[model_name] = model.predict_proba(data)[:, 1]
        print(f"Predicciones de {model_name}: {preds[model_name]}")
    scores = sum(preds[model_name] * config['weights'][model_name] for model_name in models)
    print("Predicciones ponderadas (scores):", scores)
    threshold = config['threshold']
    riesgos = [clasificar_riesgo(score, threshold) for score in scores]
    recomendaciones = generar_recomendaciones(data)
    for i in range(len(scores)):
        print(f"Riesgo: {riesgos[i]}")
        print("Recomendaciones:")
        for rec in recomendaciones[i]:
            print(f" - {rec}")
    # Explainer
    explainer = load_shap_explainer()
    # Explicar las predicciones usando el explainer cargado
    if 'lgbm' in models:
        shap_values = explainer.shap_values(data)
        shap.summary_plot(shap_values, data)
    else:
        print("Modelo no encontrado en los modelos cargados.")
    return {
        "score": round(float(scores[0]), 2),
        "riesgo": riesgos[0],
        "recomendaciones": recomendaciones[0]
    }
```

Anexo 23. Formulario de entrada de datos

PRIMERO, INGRESA EL NOMBRE DEL RECLAMADOR

FECHA DEL ACCIDENTE

12/09/2025

MARCA DEL VEHÍCULO

Selecciona una opción

ANTIGÜEDAD DEL VEHÍCULO (en años)

Selecciona una opción

FRANQUICIA DE LA PÓLIZA

Selecciona una opción

GÉNERO DEL ASEGURADO

☒ MUJER ☐ HOMBRE

ESTADO CIVIL DEL ASEGURADO

Selecciona una opción

¿CUÁNTO TIEMPO DESPUÉS SE MUDÓ EL ASEGURADOR TRAS EL ACCIDENTE? (en años)

Selecciona una opción

ZONA DONDE OCURRIÓ EL ACCIDENTE

☒ ZONA URBANA ☐ ZONA RURAL

NÚMERO DE COCHES INVOLUCRADOS EN EL ACCIDENTE

0

TESTIGOS DEL ACCIDENTE

☒ EXISTEN ☐ NO EXISTEN

Analizar reclamo

FECHA EN LA QUE SE EMITIÓ LA PÓLIZA

12/09/2025

TIPO DE VEHÍCULO

Selecciona una opción

PRECIO DEL VEHÍCULO

500

TIPO DE PÓLIZA

Selecciona una opción

EDAD DEL ASEGURADO

16

NÚMERO DE RECLAMACIONES PASADAS

0

TIPO DE AGENTE QUE GESTIONÓ LA PÓLIZA

☒ EXTERNO ☐ INTERNO

CULPABLE DEL ACCIDENTE

☒ EL PROPIO ASEGURADO ☐ OTRA PERSONA

INFORME POLICIAL DEL ACCIDENTE

☒ EXISTE ☐ NO EXISTE

NÚMERO DE DOCUMENTOS RELACIONADOS CON EL ACCIDENTE

0

Anexo 24. Modo análisis de la aplicación

- Resumen:

Resumen de la reclamación

FECHA DEL ACCIDENTE: 10/09/2025	FECHA EN LA QUE SE EMITIÓ LA PÓLIZA: 19/06/2025
MARCA DEL VEHÍCULO: HONDA	TIPO DE VEHÍCULO: UTILITARIO
ANTIGÜEDAD DEL VEHÍCULO (en años): MENOS DE 2	PRECIO DEL VEHÍCULO: 33000
FRANQUICIA DE LA PÓLIZA: 300 EUROS	TIPO DE PÓLIZA: COLISION
GÉNERO DEL ASEGURADO: HOMBRE	EDAD DEL ASEGURADO: 20
ESTADO CIVIL DEL ASEGURADO: SOLTERO	NÚMERO DE RECLAMACIONES PASADAS: 3
¿CUÁNTO TIEMPO DESPUÉS SE MUDÓ EL ASEGURADOR TRAS EL ACCIDENTE? (en años): MENOS DE 6 MESES	TIPO DE AGENTE QUE GESTIONÓ LA PÓLIZA: EXTERNO
ZONA DONDE OCURRIÓ EL ACCIDENTE: ZONA URBANA	CULPABLE DEL ACCIDENTE: EL PROPIO ASEGURADO
NÚMERO DE COCHES INVOLUCRADOS EN EL ACCIDENTE: 1	INFORME POLICIAL DEL ACCIDENTE: NO EXISTE
TESTIGOS DEL ACCIDENTE: EXISTEN	NÚMERO DE DOCUMENTOS RELACIONADOS CON EL ACCIDENTE: 1

- Resultados:

Resultado del análisis

Score de fraude

0.29

Nivel de riesgo

ALTO RIESGO

- Recomendaciones:

Recomendaciones

- Consultar los documentos suplementarios adjuntos al reclamo.
- Solicitar testimonio o contacto del testigo.
- Confirmar la responsabilidad declarada por el asegurado.
- El asegurado cambió de domicilio recientemente: validar veracidad del cambio.
- Agente externo involucrado: revisar consistencia de la documentación.
- Corroborar historial del asegurado por edad especialmente joven.

- Explicación del Modelo (IA):

Explicación del Modelo (IA)

El reclamo ha sido clasificado con un alto nivel de riesgo debido a varios factores que sugieren la necesidad de un examen más exhaustivo de la situación.

En primer lugar, la ausencia de un informe policial sobre el accidente es un punto crítico. Este documento proporciona datos objetivos y un registro oficial del incidente, lo cual es esencial para corroborar la versión de los hechos. Sin este informe, resulta difícil validar la situación tal como la describe el asegurado.


Aunque existen testigos del accidente, la recomendación de obtener los testimonios o datos de contacto es importante para confirmar la responsabilidad y la narrativa del accidente. Dado que el asegurado se ha declarado culpable del accidente, es crucial verificar que los testimonios respalden su versión para evitar posibles fraudes.


Otro aspecto a considerar es que el asegurado ha cambiado de domicilio recientemente. Es importante validar la veracidad de este cambio ya que, a veces, este tipo de movimientos puede estar ligado a intentos de eludir responsabilidades. Además, el historial del asegurado es relevante; en este caso, su joven edad y el hecho de haber tenido tres reclamaciones pasadas conllevan una predisposición a riesgos asociados, como la falta de experiencia al volante y un comportamiento más propenso a accidentes.

La implicación de un agente externo en la gestión de la póliza también aumenta la complejidad del caso. Es recomendable revisar la consistencia de los documentos proporcionados, ya que cualquier irregularidad podría reforzar la sospecha de fraude.

Finalmente, aunque no ha habido más vehículos implicados en el accidente, esto no minimiza la necesidad de una investigación detallada. La antigüedad reciente del vehículo, el valor elevado y la alta franquicia de la póliza totalizan una situación donde los incentivos para realizar un reclamo fraudulento podrían ser atractivos.

Anexo 25. Configuración de la API key

 Configuración

 Ingresa la clave API

API key guardada

Anexo 26. Historial de reclamaciones

 Historial de reclamaciones

 Buscar por rango de fechas

2025/09/10 – 2025/09/11

 Buscar por nombre

Miriam Valero_20250911140240

Fernando Martín_20250911125024

Patricia López_20250911124738

Pedro Pereira_20250911124514

Carmen Sánchez_20250910161741

Carlos García_20250910161419

 Borrar historial

Anexo 27. Presentación de la herramienta

CarClaim AI

Bienvenido/a a CarClaim AI, una herramienta diseñada para ayudarte a evaluar el **riesgo de fraude en reclamos de seguros de vehículos** de forma rápida, objetiva y precisa.

¿Cómo funciona esta herramienta?

1. Introduce los datos del reclamo en el formulario.
2. El Agente IA generará automáticamente un **score de fraude**, un **nivel de riesgo** y una serie de **recomendaciones prácticas**.
3. Si introduces una clave de API válida, obtendrás una **explicación detallada generada por el asistente IA**.
4. Utiliza esta información para **aprobar, investigar o escalar** el reclamo de forma informada.

¡Hola! Soy tu Agente IA. Estoy aquí para acompañarte en el análisis de reclamos de vehículos, identificando riesgos de fraude y ofreciéndote explicaciones útiles y recomendaciones prácticas.

