

UNIVERSIDAD EAFIT

MAESTRÍA EN CIENCIA DE DATOS Y ANALÍTICA

Análisis post operativo de investigaciones de fraudes en Aguas EPM, utilizando procesamiento de lenguaje natural

Integrantes

Juan Luis Amaya Arbeláez

José Luis Bedoya Martínez

Kevin Daniel Genez Valencia

Álvaro Javier Mutis Guerrero

Profesores

Diego Fernando Fonseca Valero

Edwin Nelson Montoya Munera

Pablo Andrés Saldarriaga Aristizábal

9 de junio de 2025

Tabla de contenido

Tabla de contenido	2
0. Descripción del Problema.....	4
1. Ciclo de vida de los datos y procesamiento analítico:	5
a. Data Source	5
b. Ingesta de Datos:	8
c. Storage:	10
d. Procesamiento (ETL):	10
e. Procesamiento Ciencia de Datos (frameworks para procesamiento):	11
f. Aplicaciones:	11
2. Procesamiento de Lenguaje Natural (Minería de texto):	14
2.1. Preparación de los datos (limpieza, tokenización, remoción de palabras de parada, lemmatización, POS):.....	14
2.2. Modelos de representación de documentos, embedding, reducción de Dimensionalidad	16
2.3. Validación y escalabilidad del modelo:	16
3. Implementación de conceptos de Álgebra Lineal.....	17
3.1 Limpieza y filtrado de datos a través de métricas y distancias adecuadas	17
3.2. Reducción de dimensionalidad:	18
4. Implementación de conceptos de Estadística para Analítica	19
4.1. EDA (Descripción de datos multivariantes) y Limpieza de datos	19
4.2. Análisis de correlaciones	23
4.3. Reducción de variables numéricas por alta correlación:	25
4.4. Reducción de variables categóricas por alta relación:	25
4.5. Clasificación supervisada con entrenamiento y validación	25
5. Despliegue de la aplicación y resultados:	29
Anexo 1. Propuesta de Arquitectura para solución con componentes de MLOps	31
6. Bibliografía:	34

Tabla de Ilustraciones

Ilustración 1. Esquema Data Source	8
Ilustración 2. Arquitectura de la solución	13
Ilustración 3. Observación Original Vs Observación Transformada	16
Ilustración 4. Pipeline Limpieza y transformación de Observación.....	17
Ilustración 5. Aplicación de distancia de Levenshtein	18
Ilustración 6. HeatMap Matriz de Correlación	23
Ilustración 7. Grid de Hiperparametros para calibración	27
Ilustración 8: Mejores Parámetros encontrados por modelo	28
Ilustración 9. App Web Despliegue del Modelo	29
Ilustración 10. Arquitectura Propuesta	33

0. Descripción del Problema

La gestión de las pérdidas comerciales de acueducto requiere investigaciones en campo que logren determinar presencia de fraudes o anomalías en las instalaciones de los clientes del servicio de Agua de EPM en el área metropolitana, el direccionamiento de esta actividad está apoyado en el análisis de patrones de consumos que indican probabilidad de fraude siendo generada una investigación en campo.

Dicha investigación es realizada por un contratista, quien se encarga de determinar la presencia o no de una acción fraudulenta.

El sistema proveedor contiene campos estructurados para indicar si la investigación fue efectiva o no (Campo efectividad) y una clasificación general (Causa Evento). La mayor información de lo encontrado está presente en el campo Observación, que es un campo de texto libre donde el técnico del contratista describe la situación.

Si bien, en principio el seguimiento post operativo se basa en los campos de Efectividad y Causa evento, el problema surge cuando queremos validar si efectivamente esa marcación estuvo bien realizada y si realmente se encontró o no un fraude o anomalía y en los casos de no efectividad, ¿qué se encontró?, esto requiere leer una a una las observaciones realizadas y a partir de dicha lectura y el análisis post operativo determinar si se debe realizar una retroalimentación al contratista o se deben depurar las reglas de direccionamiento.

En tal sentido, se busca implementar una solución que, tomando como base lo contenido en el campo observación y otros predictores se logre clasificar si la investigación realizada corresponde o no a un fraude o anomalía, igualmente si el descargue estuvo bien realizado, para lo cual es necesario implementar técnicas de procesamiento de lenguaje natural que nos permita extraer de manera correcta la información contenida en este campo.

1. Ciclo de vida de los datos y procesamiento analítico:

La solución del problema planteado involucra las siguientes etapas en el ciclo de vida de los datos:

- a. **Data Source:** El desarrollo del proyecto integrador, toma como fuente principal de información un data set en formato csv, cuya construcción requiere de la consulta de un sistema analítico “Hana” y un sistema transaccional “Hidro”. Dado que proviene desde fuentes transaccionales, los datos son estructurados.

Se debe tener en cuenta que, si bien la observación contiene datos no estructurados su almacenamiento se hace en un campo estructurado, a través de una macro en Excel se toma la información relevante y en principio, se realiza un proceso de cálculo con algunas variables como la oportunidad de ejecución y descargue.

A continuación, los campos que conforman el data set y su descripción:

Campo	Descripción
ID_ORDEM_SERVICO	Código único que identifica la orden de trabajo con que es atendida una investigación en terreno
ID_UC	Código único que identifica una instalación en el sistema
Estado Instalación	Describe el estado del contrato de la instalación con la empresa: Activo, Suspendido o Retirado
Uso	Es el uso que la instalación le da al servicio de Agua Potable: Residencial, Comercial, Industrial, entre otros
Actividad Económica	Es la actividad económica que se desarrolla en la instalación
Fecha Generación	Fecha en que se genera la orden de trabajo

Fecha Ejecución	Fecha en que se ejecuta la orden de trabajo
Fecha Descargue	Fecha en que se realiza el descargue de la ejecución de la Orden de trabajo
Fecha Ultima Lectura	Es la última fecha en que se le tomo la lectura a la instalación
Estado OT	Es el estado de la Orden de trabajo: Cerrada, descarga parcial o abierta.
Observación	Es el campo clave del análisis. Contiene la descripción en detalle de lo observado y realizado en el campo por parte del técnico
Efectividad	Indica si la investigación fue efectiva o no
Causa Efectividad	Es un campo estructurado que permite escoger la causa de la efectividad o no efectividad de la investigación
Causa Evento	Es un campo estructurado que indica la anomalía encontrada durante la investigación en los casos de efectividad positiva
Consumo promedio	Es el consumo mensual promedio de la instalación
Lectura en Visita	Valor de lectura durante la investigación
Ultimo Valor Leído	Valor de lectura tomado en la fecha de Ultima lectura
Oportunidad Ejecución	Campo calculado: Diferencia entre la fecha de generación y la fecha de visita
Oportunidad Descargue	Campo calculado: Diferencia entre la fecha de Visita y la fecha de Descargue

Diferencia Lectura	Campo Calculado: Diferencia entre la lectura tomada en la última fecha de lectura y en la fecha de visita
Días entre lecturas	Campo Calculado: Días transcurridos entre la fecha de la última lectura y la fecha de la visita
Promedio Consumo día	Campo Calculado: [Diferencia de Lecturas]/[Días Transcurridos]
Análisis Lectura	Campo Calculado: Campo que clasifica la lectura de acuerdo con el promedio consumo día
Promedio Día Histórico	Campo Calculado: [Consumo Promedio]/30
Menor a Promedio	Campo Calculado: Indica si el promedio consumo día es menor o no al consumo promedio día histórico
Score Lectura	Campo Calculado: Asigna un puntaje de acuerdo con la lectura
Score Promedio	Campo Calculado: Asigna un puntaje de acuerdo con el promedio
Total Puntaje	Campo Calculado: Es la suma entre Score Lectura y Score Promedio
ID_Accion	Código que identifica la acción que generó la investigación
Nombre Acción	Nombre de la acción que generó la investigación

El esquema del data source se presenta a continuación:



Ilustración 1. Esquema Data Source

Adicionalmente, se tienen dos archivos en formato csv de apoyo, para las acciones referentes al Procesamiento de lenguaje natural del campo observación, los cuales se describen a continuación:

Diccionario Inicial.csv: Es un archivo que fue construido durante la etapa de entrenamiento, donde se identificaron un conjunto de palabras que deben ser homologadas para mantener la consistencia y corregir errores de digitación o de ortografía. Contiene dos columnas, la primera con la palabra tal como aparece en el texto y la segunda con su “traducción”, esto es, la palabra corregida. Este diccionario se debe seguir alimentando en el ciclo de vida de la aplicación para ir alimentando las palabras validas usadas por el modelo.

StopWords.csv: Archivo que contiene palabras a eliminar de la observación dado que no tienen un significado semántico o no aportan a la descripción. Este uso de stopWords ayuda a reducir el número de columnas en el set de entrenamiento y a evita dar al modelo palabras sin significancia

b. Ingesta de Datos:

El proyecto integrador, usa como método de ingesta el Batch., el archivo fuente de datos se obtendrá con una periodicidad semanal donde se espera tener en promedio

500 registros a procesar, siendo esta la capacidad operativa del contrato de investigación y con un peso aproximado de 1 GB de datos.

El mecanismo de Ingesta seguirá el siguiente procedimiento:

- **Nuevos data set:** el analista encargado del procesamiento de datos generará el data set a través de la macro Excel de manera semanal.
- **Carga en destino:** los archivos se almacenarán en S3 de AWS, en la fase de entrenamiento se tendrá en la carpeta raw del datalake los archivos base para entrenamiento, el diccionario inicial y el archivo con stopwords y en la fase de aplicación, el analista subirá a la carpeta raw el data set a predecir.

Para la implementación del proyecto integrador, se creó un entorno virtual en el nodo maestro del clúster EMR con el fin de garantizar que todas las dependencias del proyecto permanecieran aisladas y reproducibles, generando el directorio `venv_spacy`, activando dicho entorno y actualizando las utilidades de empaquetado `–pip`, `setuptools` y `wheel` para mantener las versiones al día y evitar incompatibilidades (Raschka & Mirjalili, 2019), a continuación se instaló el paquete `ipykernel` dentro de ese entorno y se registró un kernel llamado “spacy-env” en JupyterHub, de modo que todos los notebooks del proyecto se ejecutaran usando las versiones de Python y librerías definidas en `venv_spacy`, asegurando que, si otros miembros del equipo requirieran replicar el flujo de trabajo en el mismo clúster o en otro nodo EMR, pudieran reproducir el entorno de ejecución sin diferencias inesperadas, lo cual es crucial para mantener la trazabilidad de las versiones y evitar efectos colaterales derivados de actualizaciones globales del sistema (Kuhn & Johnson, 2019).

A partir de la activación de “spacy-env” se procedió a instalar las bibliotecas esenciales para el procesamiento de datos y de lenguaje, comenzando por Pandas y NumPy para la manipulación de DataFrames y arreglos numéricos, SciPy para soporte de pruebas estadísticas y cálculos avanzados, scikit-learn para las etapas de preprocesamiento y modelado supervisado, y python-Levenshtein para medir distancias entre cadenas, recurso conveniente para detectar similitudes o errores

ortográficos en texto libre, seguidamente se incorporó spaCy en su versión 3.8.7 junto con los modelos en español es_core_news_md y es_core_news_sm, la elección de versiones fijas y de ruedas binarias persiguió dos objetivos principales: maximizar la compatibilidad binaria en el entorno EMR y reducir el tiempo de instalación evitando compilaciones locales (Fernández et al., 2016), el uso de spaCy permitirá, en fases posteriores, aplicar tokenización, lematización y etiquetado gramatical en español de forma eficiente, preparándonos para la vectorización y la ingeniería de características basadas en atributos textuales (García & Sánchez, 2019).

Amazon EMR (Elastic MapReduce) es un servicio administrado de AWS que permite crear clústeres de procesamiento distribuido para analizar grandes volúmenes de datos mediante frameworks como Hadoop y Spark, aprovechando la escalabilidad y flexibilidad de la nube (AWS EMR Developer Guide, 2021), los datos se almacenan en S3, lo que facilita su ingestión directa en EMR y garantiza un flujo de trabajo integrado entre almacenamiento y cómputo (Tan et al., 2019).

c. Storage:

El almacenamiento se realiza en Amazon S3, lo cual nos da la posibilidad de poder realizar analítica descentralizada. En esta parte, se almacenan los datos de la ingesta ya descritos anteriormente.

El proyecto requiere poco almacenamiento, pues se espera un nuevo archivo de datos una vez por semana y con un peso aproximado de 1 GB de datos, además, no es una solución que requiera técnicas de big data y la principal ventaja de montarlo en S3 es poder habilitar el acceso a los datos para una analítica descentralizada y una ejecución desde diferentes puntos de conexión.

d. Procesamiento (ETL):

Para conectar con los datos almacenados en AWS, se instalaron los paquetes necesarios para permitir que pandas leyera directamente desde la ruta s3://proyectointegradorfina/raw/, sin necesidad de descargar manualmente los archivos, de este modo se respetó la arquitectura de Data Lake con una zona “raw”

en S3, donde los archivos en bruto quedan disponibles para su procesamiento (Tan et al., 2019). Una vez completada la configuración del entorno, se procedió a cargar la base principal de entrenamiento llamada Base_Entrenamiento.csv, asignando explícitamente a cada columna un tipo de dato determinado. Las columnas de identificadores y categorías se marcaron como cadenas de texto, las variables numéricas quedaron definidas como flotantes y las columnas de fecha se convirtieron a objetos datetime, además, se especificó que cualquier marcador de error típico de hojas de cálculo (#VALUE! o #DIV/0!) se convirtiera en un valor nulo, homogeneizando la forma en que se representan las ausencias de información (Little & Rubin, 2014). Paralelamente se importaron el archivo DiccionarioInicial.csv con términos, sinónimos y patrones lingüísticos clave, y Stopwords.csv, una lista de palabras vacías en español cuyo separador de columnas era punto y coma y cuya codificación se estableció en latin-1 para preservar caracteres especiales, extrayendo únicamente la primera columna que contenía las stopwords (Fernández et al., 2016).

e. Procesamiento Ciencia de Datos (frameworks para procesamiento):

Dado que se requiere una analítica descentralizada, se utiliza Jupyter Hub para realizar todas las actividades necesarias para el procesamiento y entrenamiento del modelo de ML. La explicación del procesamiento como tal se realizará más adelante, pero en términos de tecnologías y arquitecturas (en este punto) se indica que se configuró el entorno para la utilización de las librerías de Pandas, Numpy, matplotlib, Scikit-learn y Spacy para el procesamiento de lenguaje natural. Así mismo, se utiliza joblib para la construcción y exportación del modelo.

f. Aplicaciones:

- **Tipo de Aplicación:** El reto que se plantea, requiere diseñar y entrenar un modelo de clasificación multiclase, para ello, se instaló dentro del entorno virtual “spacy-env” las bibliotecas necesarias para el proceso de modelado, incluyendo nltk para soporte de procesamiento de lenguaje, imbalanced-learn para técnicas de sobre muestreo y xgboost para el clasificador de

boosting, garantizando así que todos los paquetes se encuentren en versiones controladas y aisladas, lo cual es crucial para reproducir los resultados sin dependencias externas impredecibles (Raschka & Mirjalili, 2019)

- **Almacenamiento:** Una vez ajustado, el pipeline resultante se serializó en el archivo `mejor_modelo_entrenado.joblib` y se subió a la carpeta `refined/` del bucket S3 (`proyectointegradorfinal`), asegurando su disponibilidad para entornos de producción y facilitando su integración en servicios de inferencia automatizados (Raschka & Mirjalili, 2019).
- **Despliegue:** Para facilitar la aplicación del pipeline completo sin requerir que cada usuario reescriba las etapas de limpieza y transformación, se desarrolló una aplicación web con Streamlit que automatiza todo el proceso (preprocesamiento e inferencia) sobre un archivo CSV denominado **BasePrediccion2.csv**.

El pipeline serializado (`mejor_modelo_entrenado.joblib`), que incluye imputación de medianas, escalado, codificación one-hot, vectorización TF-IDF y el clasificador XGBoost, se carga y almacena en caché con `@st.cache_resource` para acelerar las solicitudes sucesivas y garantizar que siempre se emplee exactamente la misma versión entrenada (Raschka & Mirjalili, 2019). El usuario, simplemente sube un CSV con las mismas columnas y tipos que el conjunto original, el pipeline reconoce de forma inmediata esa estructura y al invocar `modelo.predict(df)`, encadena las siguientes etapas: imputación y escalado de variables numéricas, imputación y one-hot de variables categóricas, vectorización TF-IDF del texto y clasificación final con `XGBClassifier`. El resultado numérico (0, 1 o 2) se convierte en una etiqueta legible (“Normal”, “Anomalía” o “No Revisado”) mediante una función de mapeo, la interfaz, ofrece un botón para descargar un nuevo CSV que incluye esta columna de predicción junto con los datos originales. De este modo, cualquier miembro del equipo o usuario externo, puede replicar el pipeline sin necesidad de re implementar manualmente las etapas de limpieza y transformación, asegurando que las

predicciones sean idénticas a las obtenidas durante el entrenamiento en EMR (Chawla et al., 2002; Chen & Guestrin, 2016).

A continuación, se presenta la arquitectura de la aplicación:

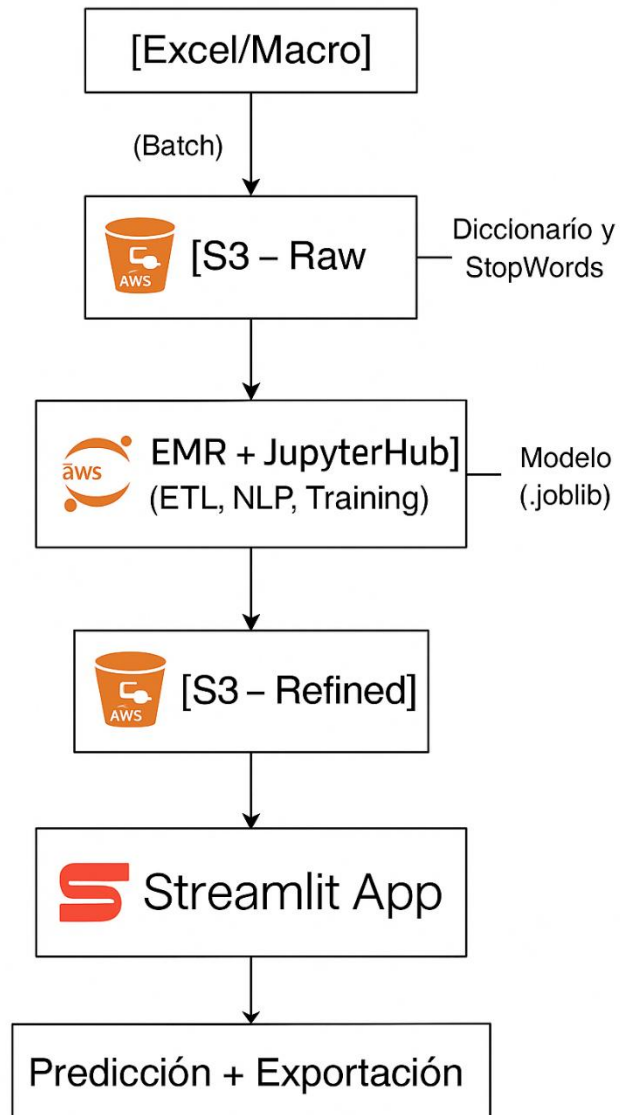


Ilustración 2. Arquitectura de la solución

2. Procesamiento de Lenguaje Natural (Minería de texto):

El problema planteado, nos pone como principal reto el poder aprovechar la información presente en el campo de observación como elemento clave para la clasificación de los resultados obtenidos. Para lograrlo, nos centramos en realizar un trabajo de procesamiento de lenguaje natural y poder llevar a datos estructurados la información que es de naturaleza no estructurada.

Existen varios problemas que resolver, donde la limpieza de datos es fundamental para poder obtener procesos de tokenización y lematización exitosos. Se usan tanto funciones propias como herramientas del paquete spacy para lograrlo.

A continuación se describe en detalle el procedimiento de limpieza y transformación que se realiza:

2.1. Preparación de los datos (limpieza, tokenización, remoción de palabras de parada, lemmatización, POS):

Para llevar a cabo estas tareas se genera un pipeline que incluye:

- ***Función depurarObservacion:*** Si bien el contenido es no estructurado, las observaciones por lineamiento contienen 3 segmentos claramente identificados:
 - Encabezado: Contiene datos informativos sobre el nombre del técnico que realiza la investigación y la fecha de la visita
 - Cuerpo: es la descripción completa de lo encontrado durante la investigación.
 - Otros datos: contiene la razón social de la instalación investigada cuando aplica

La función depurarObservacion se encarga de tomar el cuerpo.

- ***Función depurar_causa_evento:*** Cuando la observación es cargada en HANA en el proceso de ETL, se anexa la información de causa evento encontrado. Si esta etiqueta permanece, podemos estar induciendo un predictor que genere sesgo en el proceso de entrenamiento, lo cual haría que el modelo no cumpliera el objetivo. Esta función se encarga de remover esta etiqueta de la observación.

- ***Función QuitarInformacionDeSello:*** Dentro de la observación, los técnicos incluyen información del sello que tiene el medidor. Esto posee una estructura definida, por lo cual se utiliza expresiones regulares para identificar el patrón y reemplazarlo.
- ***Función QuitarNumericos y Función LimpiarCarateres:*** Se utilizan para quitar números y caracteres diferentes a letras del abecedario y sus respectivas tildes. Para ambas funciones se utilizan expresiones regulares aplicando la librería RE.
- ***Función escogerpalabras:*** Para el análisis, se tomarán tipos de palabras que generen valor. En este punto, se realiza el proceso de tokenizacion, lemmatizacion y categorías gramaticales (POS). La función determina si la palabra pertenece a cualquier de las categorías gramaticales de sustantivo, adjetivos, verbos o adverbios, una vez se selecciona, esta palabra es lematizada y así se conserva en la transformación.
- ***Función diccionario:*** Define el corpus con el cual se entrenará el modelo. Esta función examina una a una las palabras que componen una observación luego de ser lematizado el texto. Si la palabra está en el conjunto de palabras de parada (Stopwords) es ignorada, por el contrario, si la palabra no hace parte del conjunto StopWords se busca en el diccionario, en caso de no ser encontrada se realiza un valor de similitud calculado al aplicar la distancia Levenshtein y se estandariza con la longitud de la palabra, si se encuentra en el diccionario una palabra que sea similar en más de un 70% la palabra es remplazada, pero se almacena para luego ser examinada para el analista y determinar si realmente es un sinónimo o debe ser incluida en el diccionario como una nueva palabra, si aun así, no es encontrada, se agrega a la lista de palabras nuevas, y se remplacea por la palabra "imputada" para tenerla en cuenta en el entrenamiento.

En la siguiente figura vemos la comparación entre las observaciones que ingresan y el resultado luego del proceso de limpieza y transformación:

index	Observacion ▲	Observaciones_Limpias
1	Carlos Jaramillo Medidor Water tech de ½ serie#20113047667 con lectura 03164 en posición correcta sin signos de manipulación externa registra en prueba de llaves predio no residencial actividad económica es un bar razón social bar el Oscar el cual tiene 2 puntos de consumo el cual usuario manifiesta que el consumo varía de acuerdo a la cantidad de personas que transcurren en el lugar Usuario: EPMICARLOS.JARAMILLO Fecha: 2021-08-05 15:13:27	serie lectura posicion correcta sin signo manipulacion externa registra prueba llave predio no residencial actividad economica bar imputada social bar tener punto consumo usuario manifiesta consumo varian imputada cantidad persona transcurso lugar
0	Se encuentra vivienda sin medidor, con servicio directo. Usuario dice que nunca ha tenido medidor de acueducto. Usuario: EPMJACOSAC Fecha: 2021-09-13 10:43:15	encontrar vivienda sin medidor servicio directo decir tener medidor acueducto
2	Yonatan Giraldo predio solo no hay quien suministre información medidor bien instalado con cúpula impactada medidor marca wáter tech de 1/2 serie 2015450174 lectura ilegible se envía medidor para el laboratorio en bolsa de seguridad BM -17 00236 presinto CB 17-00366 se normaliza instalación mediante cambio de medidor provisional se dejan llaves y racores sin fugas fecha laboratorio 01-09-2021 predio residencial Usuario: EPMYONATAN.GIRALDO Fecha: 2021-08-19 13:22:16	predio no suministrar informacion medidor instalado cupula impactada serie lectura ilegible enviar medidor para laboratorio bolsa seguridad precinto normalizar instalacion mediante cambio provisional dejar llave racor sin fuga fecha laboratorio predio residencial

Ilustración 3. Observación Original Vs Observación Transformada

2.2. Modelos de representación de documentos, embedding, reducción de Dimensionalidad

Una vez se obtiene la observación limpia, se busca representar el texto como una matriz dispersa a través de TfidfVectorizer. En nuestro caso, se generan 1.885 columnas, es decir que nuestro data set va a contener para el manejo de la observación un total de 1.885 variables.

Se realiza un proceso de reducción de dimensionalidad utilizando técnicas de álgebra lineal que se explicarán en el apartado correspondiente.

2.3. Validación y escalabilidad del modelo:

Uno de los principales problemas, es cómo manejar las palabras nuevas que aparecen en el lenguaje y como mantener el modelo actualizado con esos cambios que pueden surgir desde el proceso con el ingreso de nuevo personal o nuevos hallazgos interesantes en el terreno. La manera que se implementa para lograrlo, es identificando y almacenando las nuevas palabras no presentes al momento del entrenamiento, utilizando una técnica de remplazo, que consiste en marcar una nueva palabra como “imputada” y se define una métrica que haga seguimiento a la cantidad de veces que se repite esta palabra, de tal modo que, al alcanzar un umbral determinado se realice actualización del diccionario y el listado de stopwords para un reentrenamiento.

A continuación, se presenta el ciclo completo de limpieza y transformación de datos:

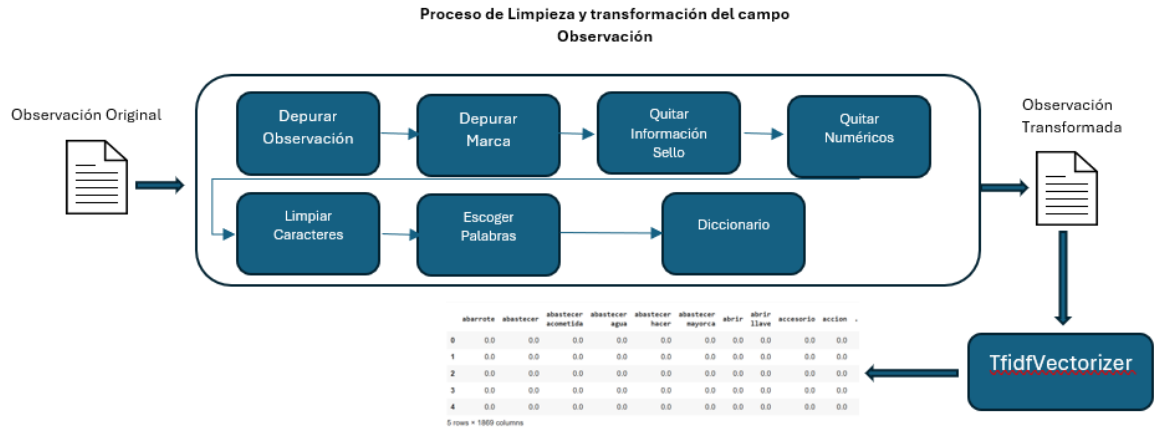


Ilustración 4. Pipeline Limpieza y transformación de Observación

3. Implementación de conceptos de Algebra Lineal

3.1 Limpieza y filtrado de datos a través de métricas y distancias adecuadas

El poder aprovechar adecuadamente las ventajas de contar con información en el campo de observación, supone un reto donde la limpieza de los datos es fundamental para obtener un modelo consistente. En esta primera tarea, resultó fundamental poder aplicar la distancia Levenshtein como herramienta óptima para lograr determinar palabras sinónimas y reducir desde la limpieza el número de variables a ingresar al modelo. Recordemos, que la distancia Levenshtein se define como el número mínimo de operaciones que se requieren para transformar una cadena en otra, cada operación tiene un peso asignado y se definen 3 operaciones: Inserción, eliminación y sustitución.

En nuestro proyecto, usamos esta métrica en la función Diccionario, cuando al encontrar una palabra nueva examinamos si existe alguna que contenga una similaridad tal que permita ser tratada como sinónimo, pero por un problema de digitación u ortografía.

La implementación, no utiliza la distancia en valor absoluto, sino que calcula un porcentaje de similaridad de acuerdo con la longitud de la palabra para lograr ponderar una mejor aproximación en palabras más largas. Por ejemplo, al usar la distancia de Levenshtein en el par de palabras {sn, sin} y {medidor,mdidor} en ambos casos se requiere una sola acción de inserción y la distancia es la misma. Sin embargo, al dividir la distancia levenshtein entre la longitud del texto tenemos que

para el primer caso el porcentaje de coincidencia es de $1/3$, con una similitud de $1 - \text{distancia}$, esto es, un 66% (se toma la longitud mayor); mientras que para el segundo caso es de $1/7$ por lo que la similitud es del 86%. En el segundo caso tengo mayor certeza de poder corregir la palabra real.

```
else:
    mejor = ''
    porcentaje = 0
    for key in dict_sinonimos:
        dist = levenshtein(palabra, key)
        max_len = max(len(palabra), len(key))
        sim = 1 - (dist / max_len)
        if sim > porcentaje:
            porcentaje = sim
            # Store the first synonym from the list
            mejor = dict_sinonimos[key][0]
    if porcentaje >= 0.7:
        remplazos[palabra] = mejor
        f.append(mejor)
    else:
        palabras_nuevas.append(palabra)
        f.append('imputada')
```

Ilustración 5. Aplicación de distancia de Levenshtein

3.2. Reducción de dimensionalidad:

El resultado de la transformación del campo de observación, nos da como resultado una matriz dispersa de tamaño (2.877x1.869), es decir, tenemos un gran set de datos, donde la mayoría de los campos son cero. En este punto, nos apoyamos del álgebra lineal para determinar colinealidad entre variables y varianza utilizando la matriz de covarianza de la matriz de datos. Iniciamos con un análisis exploratorio que nos da información sobre el tipo de matriz con la que estamos trabajando

3.2.1. Determinante de la matriz: La determinante de la matriz, es una buena medida de la presencia de colinealidad o no en una matriz de covarianza dada. En nuestro caso, el valor encontrado del determinante es de 0 lo que implica que existen variables altamente colineales.

3.2.2. **Índice de Condicionalidad:** nos permite establecer que tan cerca está una matriz de ser singular, es decir, no ser invertible. En nuestro caso, tenemos un índice de condicionalidad de 4.21×10^{18} , que es consistente con el valor del determinante, confirmando que existen variables de la matriz con alta colinealidad o colinealidad perfecta.

3.2.3. **Traza de la matriz:** es la suma de las varianzas individuales de cada variable, por sí sola no es un indicador que permita concluir colinealidad. En nuestra matriz el valor de la traza es de 1.858.

Dado que nos encontramos con una matriz que indica tener variables altamente colineales, usamos el producto interno entre las variables luego de ser escaladas y centradas en su media. Un producto interno cercano o igual a cero indica una baja colinealidad, pues las direcciones de ellos estarían formando un ángulo de 90° o cercano. Nuestro método realiza el cálculo de cada par de variables y al encontrar valores absolutos por encima de 0.9 elimina la variable con menor varianza.

Adicionalmente, tomamos como medida los eigenvalores de cada columna de la matriz, donde al encontrarnos un eigenvalor con un valor menor a 0.01 se deduce que no está aportando suficiente varianza al modelo y por tanto se elimina.

Una vez realizados estos procedimientos, pasamos de una matriz de (2.877x1.869) a una matriz de (2877 x 1630), logrando una reducción de 239 variables.

4. Implementación de conceptos de Estadística para Analítica

4.1. EDA (Descripción de datos multivariantes) y Limpieza de datos

Una vez cargados los datos, se examinó la presencia de valores faltantes en cada variable, identificándose que la columna “Causa Efectividad” presentaba aproximadamente un 35 % de

registros vacíos, mientras que las variables numéricas “Promedio Consumo día”, “Score Promedio” y “Total Puntaje” contenían cerca de un 9 % de datos ausentes, la columna “Observación” mostraba alrededor de un 8,7 % de valores nulos, requiriendo decidir cómo tratar el texto ausente: si imputarlo con cadena vacía, clasificarlo como “Sin texto” o mantener dichos registros para un análisis específico; este análisis inicial resultó determinante, porque la ausencia de texto en “Observación” implica que el operario no registró comentarios adicionales, circunstancia que podría correlacionarse con la detección de fraudes o con la eficacia de la lectura en campo, por tanto, despejar esta cuestión desde el inicio resultó fundamental para no introducir sesgos en las fases de preprocesamiento de texto (Rubin, 1976).

Para garantizar que cada registro contuviera la información temporal requerida, se eliminaron las filas que carecían de cualquiera de las fechas clave: Fecha Generación, Fecha Ejecución, Fecha Descargue o Fecha Última Lectura, reduciendo así el conjunto de datos de 3.150 a 2.895 observaciones; esta decisión obedece a que, sin fechas completas, no es posible calcular métricas temporales esenciales, como el intervalo en días entre lecturas o la oportunidad real de ejecución de la orden de servicio, además, la secuenciación de eventos resulta ambigua cuando falta al menos una fecha, lo cual podría invalidar cualquier análisis que dependa de la información cronológica para detectar retrasos o patrones de comportamiento anómalos en los consumos (Little & Rubin, 2014). A continuación, se creó una lista de columnas categóricas y otra de columnas numéricas, descartando aquellas filas en las que todas las variables de tipo texto estuviesen vacías o de manera independiente, aquellas en las que todas las variables numéricas no aportaran datos; sin embargo, en este caso particular no se produjo una reducción adicional, porque después de eliminar por fechas, cada fila permanecía con al menos un valor significativo, no obstante, se mantuvo la práctica de eliminar registros “vacíos” para evitar que datos irrelevantes o ruidosos entorpecieran la construcción de modelos (Patel & Connington, 2017).

Una vez verificado que los registros remanentes poseían como mínimo alguna columna no nula, se volvió a contabilizar la cantidad de valores faltantes en cada campo, el recuento arrojó 777 ausencias en “Causa Efectividad”, junto a 17 faltantes en cada una de las columnas numéricas “Promedio Consumo día”, “Score Promedio” y “Total Puntaje”, asimismo, “Análisis Lectura” y “Score Lectura” presentaban 7 ausencias cada una, mientras que columnas clave como “Fecha

Generación”, “Fecha Ejecución” y “ID_ORDEM_SERVICIO” no registraban valores nulos, corroborándose así, la efectividad de la limpieza inicial de fechas (Tan et al., 2019).

Dado que “Causa Efectividad” es una variable categórica central para entender si la acción en campo fue efectiva o no, se decidió imputar todos los valores faltantes en esa columna con la etiqueta “No Aplica”, recurso que permite que el modelo interprete estos casos simplemente como registros sin causa anotada, evitando que las filas se eliminen masivamente o que se genere un sesgo al sustituir nulos por categorías erróneas; tras esta imputación, el conteo mostró 1 922 casos de “No fraude”, 777 de “No Aplica” y el resto distribuido en categorías como “Agua Propia”, “Requiere investigación por otro equipo de trabajo” y “Otras Fuentes” (Little & Rubin, 2014).

El siguiente paso, consistió en atender las columnas de texto “Análisis Lectura”, “Observación” y “Causa Evento”, donde la ausencia de comentarios implicaba que el operario no proporcionó información cualitativa adicional; se consideró que esta falta de texto no podía interpretarse de forma automática como “No fraude” ni como “No hay problema”, optando por asignar la etiqueta genérica “Desconocido” a cada celda vacía en dichos campos, al reemplazar valores nulos por “Desconocido” se mantuvo la coherencia del conjunto y se posibilitó que los modelos de procesamiento de lenguaje natural capturaran la ausencia de descripción como un rasgo informativo más, en lugar de descartar el registro o interpretarlo erróneamente (Fernández et al., 2016), gracias a este etiquetado, los valores faltantes disminuyeron drásticamente y quedó pendiente únicamente un número reducido de ausencias en variables numéricas –17 en “Total Puntaje”, “Promedio Consumo día” y “Menor a Promedio”, así como 7 en “Score Lectura” y “Análisis Lectura”–, estos últimos casos, podrán resolverse en fases posteriores mediante imputación estadística (por ejemplo, con medianas o regresiones simples), pero en esta etapa preliminar se decidió posponer la estrategia para no introducir artificios prematuros (Little & Rubin, 2014).

Con los campos categóricos y textuales depurados, se procedió a calcular los estadísticos básicos de las variables numéricas –media, desviación estándar, coeficiente de asimetría (skewness) y curtosis– para describir la forma de sus distribuciones, los resultados evidenciaron asimetrías muy elevadas en “Consumo promedio” ($\text{skew} \approx 16$), “Lectura en Visita” ($\text{skew} \approx 37$), “Diferencia Lectura” ($\text{skew} \approx 37$) y “Días entre lecturas” ($\text{skew} \approx 53$), indicando que la mayoría de los

consumos y diferencias se concentra en valores bajos, mientras que existen pocos casos con magnitudes extremadamente altas; del mismo modo, la curtosis de dichas variables resultó mayor a 300, lo que denota colas pesadas y presencia de outliers extremos que distorsionan la distribución, por el contrario, “Score Lectura”, “Score Promedio” y “Total Puntaje” exhibieron asimetrías moderadas –entre 0,7 y 0,9– y curtosis cercana a cero o negativa, sugiriendo que sus datos se aproximan más a una distribución normal (Field, 2018), estos hallazgos son determinantes porque, en etapas posteriores de modelado, permitirá decidir si es necesario aplicar transformaciones logarítmicas o de Box-Cox para reducir la influencia de valores extremos o si conviene usar técnicas de Winsorización o recorte para limitar el impacto de los outliers, asimismo, la identificación de variables con sesgos pronunciados servirá para determinar si se normalizan mediante escalado robusto –por mediana e IQR– antes de alimentar algoritmos sensibles a la magnitud de las características, como SVM o redes neuronales (Friedman et al., 2001).

Posteriormente, se exploraron las frecuencias de las categorías principales de las variables categóricas, en “Estado Instalación” se observó que “ActivoCon suministro” concentraba 2 850 casos –alrededor del 98 % del total– mientras que las etiquetas relacionadas con retiros, suspensiones y casos especiales sumaban solo 45 registros –2 %–, para simplificar la variable y evitar que las categorías minoritarias dispersaran la información en niveles con muy pocas observaciones, se agrupó todo aquello distinto de “ActivoCon suministro” en una nueva etiqueta llamada “Inactivo/Retiros”, criterio que no solo redujo la cardinalidad, sino que facilitó la generación de variables dummy sin columnas escasamente pobladas que añadieran ruido (Kuhn & Johnson, 2019), en “Uso” se constató que “Residencial” y “Comercial” representaban conjuntamente más del 75 % de los casos, mientras que “Industrial” constituía otro bloque relevante, por lo que se mantuvieron estas tres categorías, las restantes se unificaron en “Otros Usos”; dicha consolidación mitigó el riesgo de sobreajuste en aquellos modelos que asignan un parámetro o peso por cada nivel categórico, de manera similar, en “Actividad Económica” los tres estratos residenciales, “Comercial” e “Industrial” se conservaron como categorías independientes, mientras que las actividades de baja frecuencia se agruparon en “Otros Sectores”, en “Estado OT” se constató que “Cerrada” representaba el 84 % de los casos y “DescargadaParcial” el 16 %, proporción suficientemente equilibrada para mantener ambas categorías sin alteraciones,

finalmente, “Menor a Promedio” presentó una distribución 68 % vs. 32 %, lo cual se consideró adecuado para conservar ambas clases sin agregar subniveles (Tan et al., 2019).

Para materializar estos cambios, se reetiquetaron los registros de “Estado Instalación” de modo que, cualquier valor distinto de “ActivoCon suministro” pasara a “Inactivo/Retiros”, en “Uso” se conservaron “Residencial”, “Comercial” e “Industrial” y todo lo demás se clasificó como “Otros Usos”, de forma similar, en “Actividad Económica” se retuvieron los estratos 1, 2 y 3, “Comercial” e “Industrial” y el restante se etiquetó como “Otros Sectores”; al ejecutar este mapeo, “Estado Instalación” quedó con 2 850 casos de “ActivoCon suministro” y 45 de “Inactivo/Retiros”, haciendo que la variable resultara más manejable y verdaderamente representativa de la realidad operativa (Friedman et al., 2001).

4.2. Análisis de correlaciones

A continuación, con las variables numéricas ya consolidadas, se construyó un mapa de calor (heatmap) de las correlaciones entre todas las columnas de tipo float64, para detectar la fuerza y el signo de las relaciones lineales.

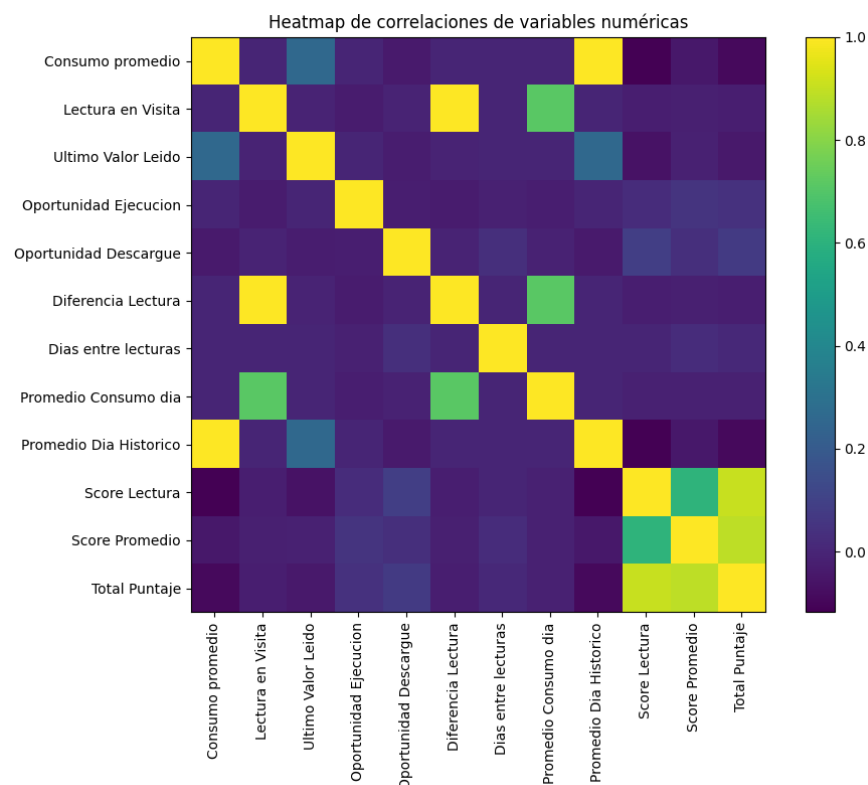


Ilustración 6. HeatMap Matriz de Correlación

En primer lugar, “Consumo promedio” y “Promedio Día Histórico” aparecen fuertemente correlacionados (valor cercano a +1), esto era de esperarse, ya que, el promedio histórico se basa directamente en el consumo registrado con anterioridad. Mantener ambos atributos podría generar multicolinealidad, por lo cual se decidió descartar uno de ellos o combinarlos (por ejemplo, calculando la diferencia entre consumo real e histórico).

Asimismo, “Lectura en Visita” y “Diferencia Lectura” muestran una correlación muy alta (también próxima a +1), dado que “Diferencia Lectura” se obtiene restando el “Último Valor Leído” de la “Lectura en Visita”, resulta evidente que ambas variables aportan información prácticamente idéntica. En la práctica, bastaría con conservar solo “Diferencia Lectura” o bien la “Lectura en Visita”, puesto que, incluir las dos conduciría a redundancia en el modelo.

Por otro lado, “Promedio Consumo día” y “Oportunidad Ejecución” presentan una correlación positiva moderada (aprox. +0,65), esto sugiere que las órdenes con mayor consumo diario tienden a ejecutarse más rápidamente, tal vez, porque el sistema prioriza casos de alto consumo. Esta relación, al ser menos intensa que las anteriores, puede resultar valiosa como predictor para acelerar la detección de posibles fraudes y merece conservarse en el conjunto final de variables.

En la zona inferior derecha del heatmap, las tres métricas de puntaje interno: “Score Lectura”, “Score Promedio” y “Total Puntaje”, aparecen también fuertemente correlacionadas entre sí (con valores por encima de +0,8), este hallazgo indica que “Score Lectura” y “Score Promedio” comparten una base común y que “Total Puntaje” se construye a partir de ambas, en consecuencia, se recomienda elegir solo una de estas variables (por ejemplo, “Total Puntaje” para evitar sobrecargar el modelo con información repetida).

Otras correlaciones moderadas se detectan entre “Oportunidad Descargue” y “Score Lectura” ($\approx +0,3$), apuntando a que un descargue más rápido de la medición suele asociarse con puntajes de lectura ligeramente superiores, de igual modo, “Promedio Consumo día” y “Oportunidad Ejecución” vuelven a reflejar ese vínculo operativo: un mayor consumo diario está vinculado a una ejecución más rápida, probablemente porque se priorizan lecturas de alto consumo.

Por último, la variable “Días entre lecturas” apenas se correlaciona con el resto (salvo una relación débil con “Diferencia Lectura”, esto indica que el intervalo entre lecturas parece tener un impacto limitado sobre los puntajes o los promedios de consumo histórico, por ende, podría considerarse menos relevante para el modelo.

4.3. Reducción de variables numéricas por alta correlación:

Para eliminar variables con alta correlación, se implementó la función `correlacion_numericas`, que examina cada par de variables numéricas presentes en el data set y al encontrar una alta correlación, que en nuestro caso, es un valor absoluto mayor a 0.7, elimina aquella cuya correlación con la variable objetivo sea menor.

El resultado de este proceso es:

Columnas Eliminadas: Promedio Dia Histórico, Lectura en Visita, Promedio Consumo día y Total Puntaje

4.4. Reducción de variables categóricas por alta relación:

Se implementa la función `chi_sq_fisher`, que establece tablas de contingencia para cada par de variables y aplica la prueba chi cuadrado cuando la tabla de contingencia es mayor de 2X2 o la prueba exacta de fisher para los casos de tablas de contingencia de 2X2. Al aplicar la prueba y encontrar que rechazar la hipótesis nula de independencia, se pasa a examinar cuál de las dos tiene mayor aporte a la variable objetivo, luego de realizar la prueba se descarta la variable “Estado de instalación”

4.5. Clasificación supervisada con entrenamiento y validación

Con el conjunto de datos completamente preprocesado y reducido, donde los atributos numéricos “Último Valor Leído”, “Diferencia Lectura”, “Promedio Día Histórico”, “Score Lectura” y “Score Promedio”, entre otros, han sido escalados, las variables categóricas “Uso”, “Actividad Económica”, “Análisis Lectura” convertidas a formato one-hot y el campo de texto “Observaciones_Limpas” representado mediante TF-IDF tras un filtrado por redundancia, se dispone de una matriz de características coherente y consistente para abordar la fase de construcción de clasificadores, ahora resta determinar qué algoritmo será capaz de explotar esta combinación heterogénea de información para predecir el estado de “CausaEvento” (clasificado como Normal, Anomalía o No revisado), para ello se evaluaron diferentes familias de modelos de

clasificación con el objetivo de maximizar la métrica F1 macro y garantizar la capacidad de generalización en datos nuevos (Kuhn & Johnson, 2019; Chen & Guestrin, 2016)

Se dividió el conjunto final de características (X) y la variable objetivo-codificada (CausaClasificada) en dos subconjuntos estratificados: entrenamiento (70 %) y prueba (30 %), de esta manera, se respetó la proporción original de instancias “Normal”, “Anomalía” y “No revisado” en ambas particiones, evitando que el sesgo de clases mayoritarias afectara la evaluación posterior (Chawla et al., 2002)

Sobre el subconjunto de entrenamiento, se definieron dos pipelines de preprocesamiento en un único ColumnTransformer:

- Variables numéricas: se aplicó SimpleImputer(strategy='median') para imputar valores faltantes con la mediana y a continuación, StandardScaler, para centrar y escalar cada atributo, de modo que ninguna variable domine por su magnitud (Raschka & Mirjalili, 2019)
- Variables categóricas: se empleó SimpleImputer(strategy='constant', fill_value='missing') para reemplazar nulos con la etiqueta “missing” y luego OneHotEncoder(handle_unknown='ignore'), para convertir cada categoría en variables binarias, garantizando que en la fase de prueba no surgieran errores al encontrar categorías desconocidas

Este preprocesador único se combinó con los clasificadores dentro de pipelines, asegurando que, en cada iteración, los datos se procesaran de forma automática y coherente (Kuhn & Johnson, 2019)

Dado que la clase “Normal” era claramente mayoritaria frente a “Anomalía” y “No revisado”, se incorporó SMOTE para generar instancias sintéticas en las clases minoritarias durante el entrenamiento, mejorando la capacidad de los modelos para aprender patrones de las clases menos representadas, sin necesidad de eliminar ejemplos de la clase mayoritaria “Normal” (Chawla et al., 2002), sin embargo, se observó que algunos algoritmos basados en árboles, en particular XGBoost, mostraban robustez intrínseca frente al desequilibrio, gracias a sus parámetros de regularización y a la capacidad de manejar pesos de clase de forma nativa.

Se seleccionaron seis familias de modelos, cada una integrada en un pipeline completo (preprocesado + clasificador) y se configuró una grilla de búsqueda de hiperparámetros para cada algoritmo:

- **Regresión Logística Multinomial**, para establecer una línea de base probabilística y ofrecer interpretabilidad en términos de coeficientes (Raschka & Mirjalili, 2019).
- **XGBoost**, algoritmo de boosting basado en árboles que combina múltiples árboles débiles para corregir iterativamente los errores y que permite controlar el sobreajuste mediante parámetros de profundidad (max_depth) y tasa de aprendizaje (learning_rate) (Chen & Guestrin, 2016).
- **Árbol de Decisión (Tree)**, modelo interpretable que divide el espacio de características en nodos basados en condiciones, regulable a través de la profundidad máxima para evitar sobreajuste (Friedman, Hastie & Tibshirani, 2001).
- **Support Vector Machine (SVC)**, que busca maximizar el margen de separación en espacios de alta dimensión y con kernels, pudiendo capturar relaciones no lineales (Cortes & Vapnik, 1995).
- **Random Forest (RF)**, ensayo de árboles de decisión, que reduce la varianza a través de bagging y aleatorización de atributos, ofreciendo estabilidad frente a ruido (Breiman, 2001).
- **LDA / QDA**, métodos discriminantes basados en supuestos de normalidad condicional de las variables, con LDA proyectando a un espacio lineal de baja dimensión y QDA permitiendo covarianzas específicas por clase (Hastie, Tibshirani & Friedman, 2009).

Para cada modelo se definió el siguiente cuadro de búsqueda de hiperparámetros:

Modelo	Hiperparámetros
Regresión Logística	C: [0.01, 0.1, 1]; solver: ['lbfgs', 'liblinear']; penalty: ['l2']
XGBoost	n_estimators: [100, 200, 300]; max_depth: [3, 5, 7, 10]; learning_rate: [0.01, 0.1, 0.2]
Árbol de Decisión	criterion: ['gini', 'entropy']; max_depth: [5, 10, 20]
SVC	C: [1, 10]; gamma: ['scale', 'auto']
Random Forest	n_estimators: [10, 50]; criterion: ['gini', 'entropy']; max_depth: [5, 10]
LDA	solver: ['svd', 'lsqr']
QDA	reg_param: [0.1, 0.5, 1.0]

Ilustración 7. Grid de Hiperparametros para calibración

Cada pipeline (preprocesado + clasificador) se ajustó mediante GridSearchCV con validación cruzada estratificada en cinco particiones (StratifiedKFold), optimizando la métrica F1 macro para penalizar por igual errores en todas las clases, de esta forma se aseguró que los hiperparámetros se seleccionaran de acuerdo con su capacidad real de generalizar, más allá del conjunto de entrenamiento y no simplemente por sobre ajustar a un subconjunto particular (Kuhn & Johnson, 2019).

Al finalizar la búsqueda de hiperparámetros, se obtuvieron los siguientes resultados en el conjunto de prueba (F1_Test), ordenados de mayor a menor:

Modelo	F1-score	Hiperparámetros
XGBoost	0.549	learning_rate=0.2, max_depth=5, n_estimators=200
Árbol de Decisión	0.5154	criterion='entropy', max_depth=10
Random Forest	0.5069	criterion='entropy', max_depth=10, n_estimators=50
QDA	0.4517	reg_param=1.0
Regresión Logística	0.4506	C=1, penalty='l2', solver='lbfgs'
LDA	0.4391	solver='lsqr'
SVC	0.4306	C=10, gamma='auto'

Ilustración 8: Mejores Parámetros encontrados por modelo

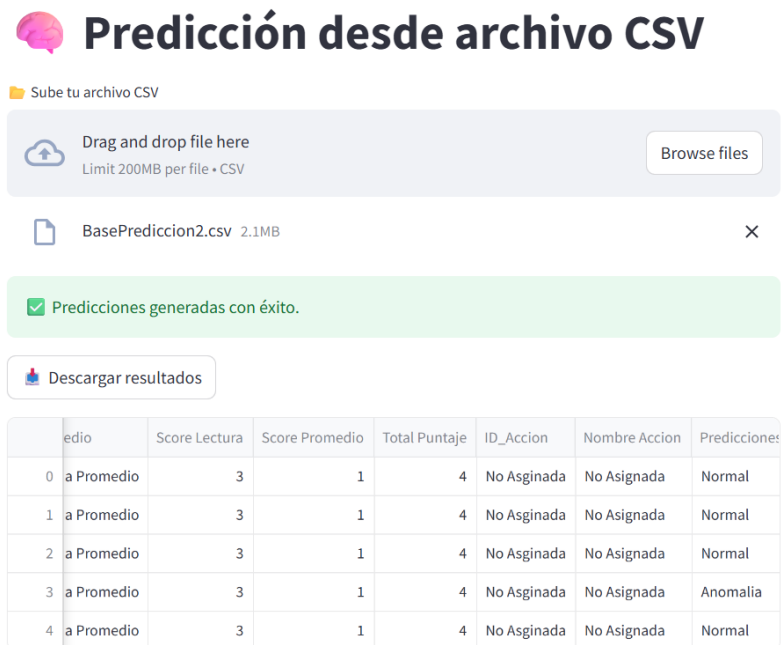
El desempeño superior de XGBoost, se atribuye a su capacidad para modelar interacciones entre características numéricas, categóricas y textuales, gracias a la construcción secuencial de árboles que corrigen errores previos y a la regularización que previene el sobreajuste (Chen & Guestrin, 2016; Friedman, Hastie & Tibshirani, 2001).

Con XGBoost como modelo definitivo, se construyó el pipeline completo (preprocesado + XGBClassifier con los hiperparámetros óptimos) y se entrenó sobre todo el conjunto de entrenamiento (X_train, y_train) para aprovechar la totalidad de la información disponible.

Una vez ajustado, el pipeline resultante se serializó en el archivo mejor_modelo_entrenado.joblib y se subió a la carpeta refined/ del bucket S3 (proyectointegradorfinal), asegurando su disponibilidad para entornos de producción y facilitando su integración en servicios de inferencia automatizados (Raschka & Mirjalili, 2019).

5. Despliegue de la aplicación y resultados:

Para facilitar la interacción del usuario con el modelo entrenado, se utiliza la librería Streamlit, que construye una aplicación web que corre localmente y logra dar una interfaz sencilla y simple pero efectiva para el uso de la herramienta. En esta, basta con subir el conjunto de datos a entrenar y en pocos segundos recibe un archivo con las predicciones y que puede descargar en formato .csv para su uso en el proceso que lo requiera.



	edio	Score Lectura	Score Promedio	Total Puntaje	ID_Accion	Nombre Accion	Predicciones
0	a Promedio	3	1	4	No Asignada	No Asignada	Normal
1	a Promedio	3	1	4	No Asignada	No Asignada	Normal
2	a Promedio	3	1	4	No Asignada	No Asignada	Normal
3	a Promedio	3	1	4	No Asignada	No Asignada	Anomalia
4	a Promedio	3	1	4	No Asignada	No Asignada	Normal

Ilustración 9. App Web Despliegue del Modelo

A partir de las predicciones generadas por la aplicación Streamlit, el desglose de las etiquetas en el conjunto procesado es el siguiente:

- Anomalía: 1 515 registros (56,85 %)

- Normal: 1 063 registros (39,89 %)
- No Revisado: 87 registros (3,26 %)

Más de la mitad de los casos (1515), fueron catalogados como “Anomalía” (56,85 %). Esto sugiere que, en este conjunto de datos, el modelo detectó patrones (ya sea en consumo, texto de observaciones u otras características) que coinciden con comportamientos atípicos o potenciales irregularidades, el hecho de que la clase Anomalía sea mayoritaria, podría reflejar un sesgo real en los datos (por ejemplo, un porcentaje alto de lecturas problemáticas) o indicar que el modelo con los umbrales establecidos, tiende a clasificar más registros como anomalías.

Por otro lado, el 40% de las observaciones quedaron clasificadas como “Normal” (1068), lo cual indica que el modelo también reconoce un volumen importante de lecturas cuyo comportamiento concuerda con el perfil esperado (consumo dentro de rangos usuales y sin señales textuales de irregularidades), ese porcentaje sugiere que, a pesar de la prevalencia de anomalías en la predicción, todavía existe un grupo significativo de casos con patrones “limpios”.

Poca proporción de “No Revisado” (3,26 %): Solo 87 filas recibieron la etiqueta “No Revisado”, dicha categoría agrupa aquellas “CausaEvento” que originalmente se consideraron: ni claramente normales, ni claramente anómalas (por ejemplo, casos en que no se autorizó trabajo o faltaba información). La proporción reducida indica que, dentro del pipeline de preprocesamiento, la mayor parte de registros fue imputada en alguna de las otras dos clases definitivas (“Normal” o “Anomalía”), quedando muy pocos que el modelo trate como “No Revisado”.

Anexo 1. Propuesta de Arquitectura para solución con componentes de MLOps

A continuación, se propone una arquitectura, que permita robustecer la solución adicionando elementos y componentes de MLOps en la plataforma de AWS:

Entrada / Ingesta de datos

- AWS Glue / AWS DataBrew: Automatiza la ingesta de datos desde Excel o archivos planos hacia S3.
- Amazon EventBridge: Para ejecutar procesos por eventos (por ejemplo, nuevos archivos cargados).

Almacenamiento

- Amazon S3 (Raw, Processed, Refined): Usar distintos buckets/prefijos con políticas de versionado y control de acceso.

Procesamiento y Feature Engineering

- AWS Glue / EMR: Para el preprocesamiento batch.
- Amazon SageMaker Processing: Alternativa más robusta para procesamiento escalable con logs integrados.

Entrenamiento de Modelos

- Amazon SageMaker Training: Entrena modelos con trazabilidad de métricas, hyperparameter tuning y jobs escalables.
- MLflow / SageMaker Experiments: Para trackeo de experimentos.

Registro y Versionado de Modelos

- SageMaker Model Registry: Control de versiones, aprobación y despliegue de modelos.
- MLflow Model Registry (opcional si usas MLflow).

Despliegue

- Streamlit App + API Gateway + Lambda: La app puede enviar requests a un endpoint o trigger Lambda para predicción.

Monitoreo y Mantenimiento

- Amazon CloudWatch: Para logs, métricas y alarmas.
- SageMaker Model Monitor: Detecta data drift, concept drift y problemas de performance.
- Amazon Athena / QuickSight: Para consultas y dashboards analíticos sobre logs/resultados.

CI/CD para ML (Automatización)

- AWS CodePipeline + CodeBuild + CodeCommit / GitHub Actions:

Entrenamiento automático cuando se actualizan datos o código.

- Validación de modelos y promoción a producción.
- Pruebas automáticas (unitarias y de validación de desempeño).

Seguridad y Gobernanza

- IAM Roles + KMS + S3 Encryption + Audit Logs
- Amazon SageMaker Clarify: Para auditoría de sesgos y explicabilidad de modelos.

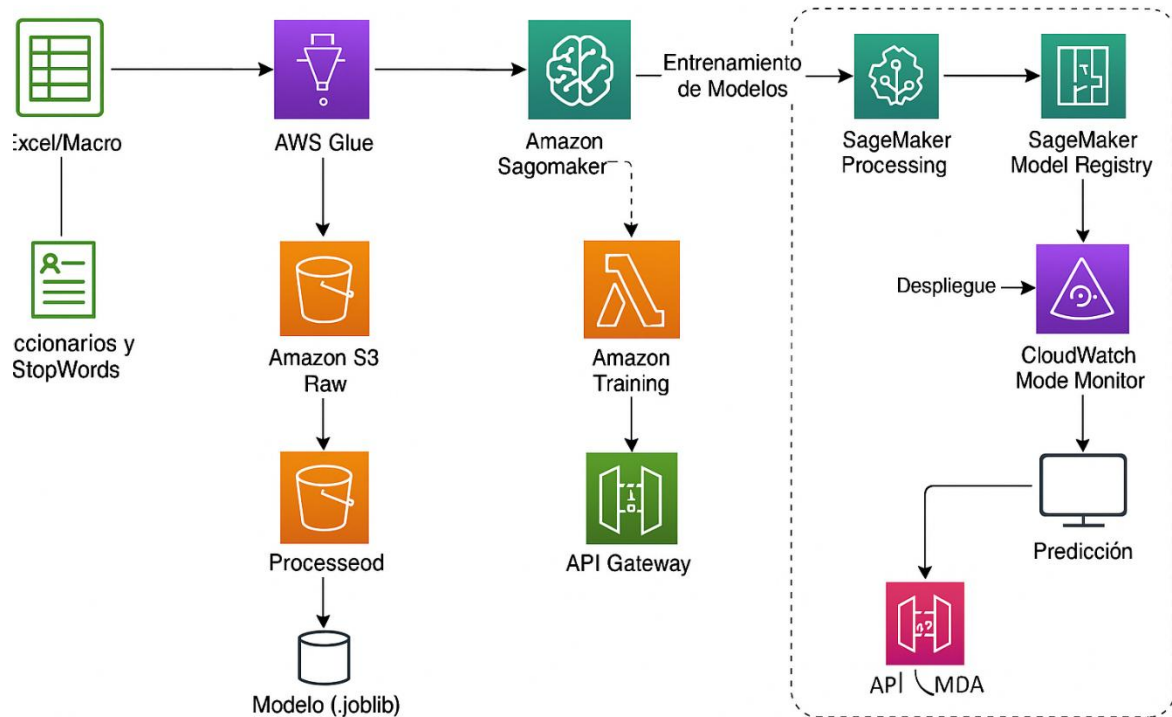


Ilustración 10. Arquitectura Propuesta

6. Bibliografía:

AWS EMR Developer Guide. (2021). *AWS Elastic MapReduce Developer Guide*. Amazon Web Services.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. En *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Fernández, A., del Río, S., & Herrera, F. (2016). *Preprocesamiento de datos para minería de datos y aprendizaje automático*. Springer.

Field, A. (2018). *Discovering statistics using IBM SPSS Statistics* (5ª ed.). Sage Publications.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

García, S., & Sánchez, J. (2019). *Ingeniería de características para aprendizaje supervisado*. Editorial Académica.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2ª ed.). Springer.

Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.

Little, R. J. A., & Rubin, D. B. (2014). *Statistical analysis with missing data* (2ª ed.). John Wiley & Sons.

Raschka, S., & Mirjalili, V. (2019). *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow* (3ª ed.). Packt Publishing.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3), 581–592.

Tan, P.-N., Steinbach, M., & Kumar, V. (2019). *Introduction to data mining* (2ª ed.). Pearson.