Detección de instalaciones fraudulentas utilizando datos históricos de consumo energéticos

Susana Maria Alvarez Castillo
Luis Miguel Román Pereira

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesores
Walter Mauricio Villa Acevedo, Dr-Ing.
Álvaro Jaramillo Duque, PhD

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2024

Cita	(Alvarez Castillo & Román Pereira, 2024)
Referencia **Estilo APA 7 (2020)**	Álvarez Castillo, S. M., & Román Pereira, L. M. (2024). Detección de instalaciones fraudulentas utilizando datos históricos de consumo energéticos, Trabajo de monografía en la especialización de análitica y ciencia de datos]. Universidad de Antioquia, Medellín, Colombia.

Especialización en Analítica y Ciencia de Datos, Cohorte VI.

Centro de Investigación Ambientales y de Ingeniería (CIA).

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co
Rector: John Jairo Arboleda Céspedes.
Decano: Julio Cesar Saldarriaga Molina
Jefe departamento: Diego José Luis Botia Valderrama

Tabla de contenido

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

CREG	Comisión de Regulación de Energía y Gas
SQL	Lenguaje de consulta estructurada
kVh	Kilovoltio por hora
NaN	Not a Number
SMOTE	Synthetic Minority Over-sampling Technique
KNN	k-vecinos más cercanos
CRISP-DM	Cross-Industry Standard Process for Data Mining
CRISP-DM	Cross-Industry Standard Process for Data Mining
SVM	Support Vector Machine
LR	Logistic Regression
RF	Random Forest
DTC	Decision Tree Classifier
MLPC	Multi-layer Perceptron classifier
NN	Neural Network
PCA	Análisis de Componentes Principales
ReLU	Función de activación
tp	True positives
tn	True negatives
fn	false negatives
fp	false positives
ROC	Receiver Operating Characteristic
AUC	Área bajo la curva
FPR	False Positive Rate
TPR	True Positive Rate
CV	Cross-validation
VIF	Factor de Inflación de la Varianza
GPU	Unidad de Procesamiento Gráfico
TPU	Unidad de Procesamiento Tensorial
GANs	Redes Generativas Adversativas

Resumen

El fraude de electricidad provoca importantes pérdidas económicas para las compañías encargadas de suministrar energía eléctrica, disminuyendo sus ingresos por consumos no facturados. Estos aumentos en los gastos derivados de prácticas fraudulentas suelen repercutir en última instancia en los consumidores, lo que se traduce en tarifas eléctricas más elevadas.

Para abordar el desafío de reducir las pérdidas no técnicas, se generó un modelo de clasificación que permite identificar instalaciones con alta probabilidad de presentar fraudes. Este modelo se entrenó utilizando datos históricos de consumos, inspecciones y otras variables relevantes, aplicando técnicas avanzadas de aprendizaje automático para reconocer patrones asociados con comportamientos fraudulentos.

Abstract

The electricity fraud causes significant economic losses for the companies responsible for supplying electricity, reducing their revenue due to unreported consumption. These increases in expenses resulting from fraudulent practices often ultimately impact consumers, translating into higher electricity tariffs. Through this research, we aim to tackle the challenge of reducing non-technical losses by identifying patterns that could indicate the likelihood of irregularities or fraud in the supply of electrical energy at a given installation.

To address the challenge of reducing non-technical losses, a classification model was developed to identify installations with a high probability of fraud. This model was trained using historical data on consumption, inspections, and other relevant variables, applying advanced machine learning techniques to recognize patterns associated with fraudulent behavior.

1. Descripción del problema

1.1. Problema de negocio

En la mayoría de países, las regulaciones y sanciones relacionadas con el fraude y el robo de energía eléctrica y servicios públicos pueden variar, pero generalmente se considera un delito y puede dar lugar a multas sustanciales y, en casos graves, penas de prisión.

Por ejemplo, en Colombia, estas acciones son consideradas delito y pueden resultar en la imposición de cuantiosas multas y la condena a varios años de prisión:

> ARTÍCULO 256 Ley 599 de 2000 Defraudación de fluidos. El que mediante cualquier mecanismo clandestino o alterando los sistemas de control o aparatos contadores, se apropie de energía eléctrica, agua, gas natural, o señal de telecomunicaciones, en perjuicio ajeno, incurrirá en prisión de dieciséis (16) a setenta y dos (72) meses y en multa de uno punto treinta y tres (1.33) a ciento cincuenta (150) salarios mínimos legales mensuales vigentes. (*Ley 599 De 2000 Congreso De La República De Colombia*, 2000)

Las pérdidas de energía a lo largo de la cadena de producción, transporte y consumo eléctrico son conocidas. Estas pérdidas impactan en la cantidad de energía comprada y en los ingresos de distribución. Así, la reducción de pérdidas beneficia a los consumidores, con efectos positivos en la accesibilidad tarifaria, en Colombia, la tarifa de la electricidad incluye un componente relacionado con las pérdidas eléctricas. Esto significa que el costo del kilovatio-hora (kWh) que los consumidores pagan refleja no solo el precio de la energía generada, sino también las pérdidas que ocurren durante su transporte y distribución

Existen dos tipos de pérdidas: (CREG - Comisión de Regulación de Energía y Gas, n.d.)

1. **Pérdidas Técnicas:** Representan la energía disipada como calor en los elementos de conexión entre la fuente en la red de suministro eléctrico y los puntos de entrega o usuarios. Estas pérdidas se deben a las leyes físicas, efecto Joule, eficiencia de los componentes del sistema, y a la falta de calibración de los equipos de medición.
2. **Pérdidas No Técnicas:** Son la diferencia entre las pérdidas totales y las técnicas. Estás pérdidas se dan debido a robos o conexiones ilegales a la red eléctrica, modificación ilegal de los equipos de medida, errores de medición y de facturación, entre otros.

Adicionalmente, en Colombia, la normativa que regula el umbral de pérdidas no técnicas se establece principalmente a través de la Comisión de Regulación de Energía y Gas (CREG) y se aplica a las empresas de distribución y comercialización de energía eléctrica en el país. Estas regulaciones establecen los límites permitidos de pérdidas no técnicas y definen las medidas que las empresas deben tomar para reducir y controlar estas pérdidas. Si las empresas no

cumplen con los umbrales establecidos, están sujetas a la imposición de multas, sanciones y medidas correctivas, de acuerdo con la normativa vigente en Colombia (Comisión de Regulación de Energía y Gas [CREG], 2018). Dado lo anterior, es muy importante para las empresas distribuidoras identificar dónde ocurren situaciones de fraudes o robos de energía eléctrica en sus redes; hoy en día se propone el uso de herramientas de analitica de datos e inteligencia artificial en la información de facturación y consumos de energía de los usuarios para determinar si existen fraudes en su sistemas.

1.2. Aproximación desde la analítica de datos

A través de una aplicación para la reducción de pérdidas, perteneciente a una empresa de energía eléctrica en Chile, se tiene acceso a la base de datos de clientes consumidores y sus respectivas características incluyendo la variable objetivo “was_fraud”, dicha variable nos indica si un cliente cometió o no fraude. En este contexto se decide optar por desarrollar un modelo de aprendizaje supervisado, específicamente un modelo de clasificación. El modelo resultante, permitiría anticipar y evaluar el riesgo de fraude de un cliente en particular, resultando beneficioso en los siguientes aspectos:

1. **Identificación temprana de potenciales casos de fraude:** Los modelos pueden ayudar a las empresas a detectar clientes que podrían estar más inclinados a cometer fraude, permitiendo tomar medidas preventivas o de seguimiento.
2. **Uso efectivo de recursos:** Al identificar con mayor precisión los casos con mayor probabilidad de fraude, se pueden asignar recursos y esfuerzos de manera más efectiva para la verificación y seguimiento de estos casos específicos.
3. **Mejora en la toma de decisiones:** Los modelos predictivos brindan información valiosa para la toma de decisiones más informadas y estratégicas en cuanto a políticas de prevención de fraude y estrategias de mitigación de riesgos.
4. **Mejora en la eficiencia operativa:** Al reducir el número de casos de fraude no detectados, se mejora la eficiencia operativa y se reducen las pérdidas económicas asociadas a estas actividades fraudulentas, por ejemplo:
    - Recuperación de energía: Corresponderá a la energía cobrada por un periodo de consumo irregular del cliente.
    - Afloramiento: Corresponderá al aumento de facturación que se tiene cuando se normaliza la instalación de un cliente.
    - La reducción de energía comprada que se tiene cuando las instalaciones no están robando.

1.3. Origen de los datos

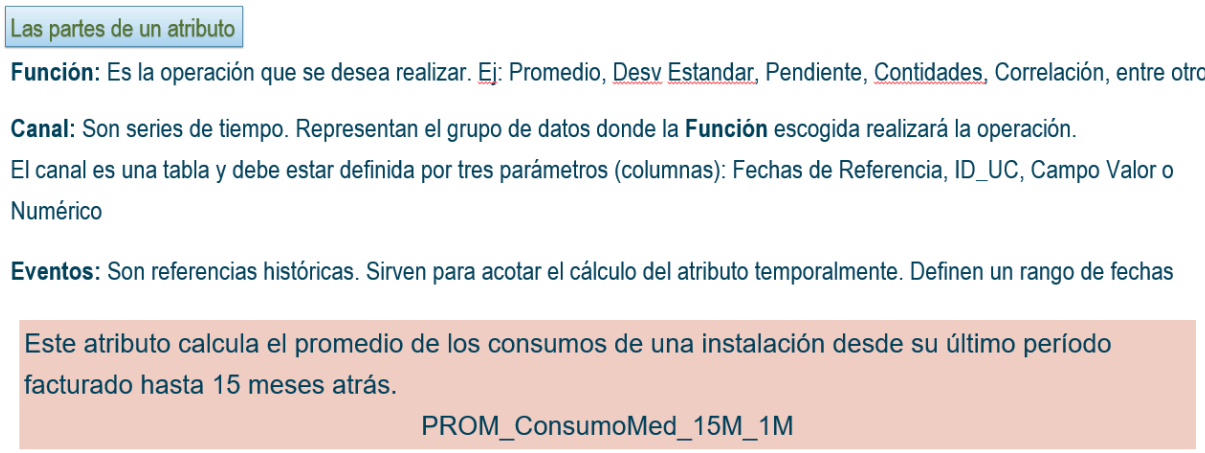
Desde el aplicativo de una empresa comercializadora de energía en Chile, se accedió a una base de datos relacional MySQL, en la que se almacena información sobre las pérdidas no técnicas, los resultados de inspecciones y los atributos que caracterizan a las instalaciones consumidoras de energía. Utilizando sentencias SQL, se extrajeron dichos datos, filtrándose según las principales características y comportamientos presentados durante un periodo de tiempo determinado. El resultado de esta extracción se almacenó en un archivo plano para su posterior análisis.

En el contexto de este trabajo, se llevó a cabo la extracción de datos correspondientes a los 2 últimos años de historia de inspecciones de fraude y el último año de inspecciones normales, esta estrategia se emplea para abordar el desbalance entre las clases de fraude y no fraude.

Cada atributo a utilizar contiene un cálculo basado en una ventana de tiempo, la cual está definida por el rango especificado en cada variable. Esta ventana de tiempo se re-calcula diariamente, según el criterio establecido por la aplicación. Sin embargo, en casos donde el fin del rango de tiempo no esté explícitamente definido en el atributo, se tomará como fecha final el 5 de mayo de 2024, momento en el cual se extrajeron los datos.

Por ejemplo, consideremos el atributo "Pendiente del consumo facturado de los últimos 24 meses". Aquí, la ventana de tiempo abarca los últimos 24 meses, pero no se especifica una fecha de finalización. En este caso, la fecha final se establecerá como el 5 de mayo de 2024, independientemente de la fecha actual. En la siguiente imagen se muestra un ejemplo de las partes del atributo y su interpretación:

Figura 1: Partes de un atributo



1.4. Métricas de desempeño

Métricas de machine learning: En la tarea de clasificación binaria se emplearán métricas como la exactitud, precisión, sensibilidad, y F1 score, para validar los resultados obtenidos:

- **Exactitud (Accuracy):** Proporción de todas las predicciones correctas, incluyendo tanto las instalaciones eléctricas identificadas correctamente como fraudulentas como aquellas identificadas correctamente como no fraudulentas.
- **Precisión (Precisión):** Porcentaje de las instalaciones eléctricas identificadas como fraudulentas que realmente son fraudes. Es decir, de todas las instalaciones eléctricas clasificadas como fraudulentas, ¿cuántas son realmente fraudes?
- **Recuperación (Recall o Sensibilidad):** Porcentaje de los casos reales de fraude que el modelo logra identificar correctamente. De todas las instalaciones eléctricas que realmente son fraudes, ¿cuántas fueron detectadas por el modelo?, esta métrica tiene más importancia, respecto a las otras debido a que es importante capturar todos los casos fraudulentos.
- **Puntuación F1 (F1-Score):** Es una medida equilibrada que combina precisión y recall. Representa la precisión y completitud del modelo para detectar fraudes eléctricos. Ayuda a encontrar un equilibrio entre identificar correctamente fraudes y asegurarse de que las identificaciones positivas son realmente fraudes.

Métricas de negocio:

- Efectividad: El cálculo de efectividad contempla la relación de casos efectivos (positivos) en detección de fraudes e irregularidades, junto con el total de direccionamiento ejecutado. Se tiene como objetivo superar el 12% de efectividad actual de la empresa mediante la implementación del modelo propuesto en este proyecto.

  A continuación la fórmula de cálculo:

$$Efectividad_t = \frac{Fraude + anomalías}{Fraudes + anomalías + No\ fraudes}$$

- Índice de productividad: A través de los indicadores descritos a continuación, se espera superar los valores que actualmente presenta la compañía distribuidora de energía.

$$Energía\ recuperada\ por\ fraude_t = \frac{Energía\ recuperada}{Fraudes}$$

$$Energía\ recuperada\ por\ revisión\ efectiva_t = \frac{Energía\ recuperada}{Fraudes + anomalías}$$

$$Energía\ recuperada\ percápita_t = \frac{Energía\ recuperada}{Fraudes + anomalías + No\ fraude}$$

- Reducción de Pérdidas Económicas: Mide el impacto del modelo en la reducción de pérdidas económicas debidas al fraude. Compara las pérdidas antes y después de la implementación del modelo en términos económicos.

2. Objetivos

2.1. Objetivo general

Identificar casos de fraude en un sistema eléctrico utilizando técnicas de aprendizaje automático, con el fin de reducir las pérdidas de energía no técnicas. Para ello, se analizarán datos recopilados de inspecciones realizadas durante un período de dos años.

2.2. Objetivos específicos

- Identificar si existe relación significativa entre los patrones de consumo de energía o alguna correlación entre las diferentes variables recopiladas por el operador del sistema.
- Realizar un análisis descriptivo de los datos históricos de facturación y consumo de energía para alimentar el modelo y así indicar posibles casos de fraude.
- Implementar modelos de clasificación usando Machine Learning para predecir instalaciones fraudulentas,
- Validar los modelos propuestos para la predicción de instalaciones fraudulentas a través de las métricas de desempeño.

3. Datos

3.1. Datos originales

La fuente principal de datos proviene de un software de reducción de pérdidas no técnicas, donde se almacenan las estructuras principales que describen las instalaciones consumidoras, su historial de consumo, revisión u otras órdenes de servicio, de una compañía de distribución de energía eléctrica en Chile. Esta información se guarda en un motor relacional de base de datos MySQL.

En el contexto de este trabajo, se llevará a cabo la extracción de datos, a través de código SQL, correspondientes a los 2 últimos años de historia de inspecciones de fraude y el último año de inspecciones normales, esta estrategia se usa para abordar el desbalance de clases desde la fuente, a través del submuestreo de los casos de 'no fraude'. Como resultado de esta estrategia quedan disponibles 9.363 registros, donde el 63,5% son datos etiquetados como no fraude y 36,4% como no fraude.

Las variables numéricas a utilizar corresponden a valores extraídos de las series temporales (periodos de tiempo donde existe una variación) asociadas a cada instalación, mediante la utilización de una función y la especificación de un Período de tiempo (Rango de eventos) por ejemplo: PROM_CONSUMOFACT_6M_1M, indicará el promedio del consumo facturado en los últimos 6 meses.

Variables de Entrada:

Variables de entrada numéricas: Son atributos que identifican el comportamiento de la instalación, en el archivo existen 377 indicadores, ejemplos:

Tabla 1: Variables numéricas de ejemplo

Indicador	Descripción
N232	Identifica el número de inspecciones en los 2 meses antes del último consumo.
N255	Valor del consumo medido en la última facturación
N256	Promedio del consumo medido en los últimos 3 meses
N266	Cantidad de consumos medidos en los últimos 24 meses.
N285	Cantidad de meses transcurridos entre la última revisión y la fecha del último consumo
N665	Mínimo de consumo entre el primer consumo y cinco meses

Variable de salida (Target):

was_fraud: Indica si la instalación es fraudulenta o no. Variable binaria 1 indica fraude y 0 para no fraude.

3.2. Datsets

Para la construcción del dataset, se procedió a una limpieza de datos. Se redujo el conjunto de datos original que constaba de 377 variables numéricas y 9.363 registros a 50 variables y 9.352 registros. Las estrategias empleadas se centraron en: Tratamiento de valores nulos. Eliminación de duplicados y variables numéricas con valores en cero (0), junto con la remoción de aquellas variables que mostraban una alta correlación.

Para los valores faltantes, se optó por imputar los valores numéricos utilizando la mediana, Además, se realizó un análisis de “outliers” con el fin de identificar posibles errores en la recopilación de datos, permitiendo validar la calidad de los datos y corregir posibles errores.

Para abordar el desbalance en la variable objetivo "was_fraud" se utilizó el algoritmo SMOTE (Synthetic Minority Over-sampling Technique) de la biblioteca imbalanced-learn (Nogueira & K, 2017). Este método genera nuevas instancias sintéticas de la clase minoritaria mediante la

interpolación entre las muestras existentes y sus vecinos más cercanos, utilizando la técnica de los k-vecinos más cercanos (k-NN) , logrando así un equilibrio deseado entre las clases. En este caso, se utilizó un sampling_strategy de 0.7, lo que significa que la clase minoritaria se aumentó hasta representar el 70% del tamaño de la clase mayoritaria.

3.3. Analítica descriptiva

Variables numéricas:

Son atributos que identifican el comportamiento de la instalación en un periodo dado, en el archivo existen 377 indicadores, ejemplos:

Tabla 2: Descripción de indicadores.

Indicador	Descripción
N232	Identifica el número de inspecciones en los 2M antes del último consumo.
N255	Valor del consumo medido en la última facturación
N256	PROMEDIO del consumo medido en los últimos 3 meses
N266	Cantidad de consumos medidos en los últimos 24 meses.
N285	Cantidad de meses transcurridos entre la última revisión y la fecha del último consumo
N665	Mínimo de consumo entre el primer consumo y cinco meses

La siguiente tabla muestra alguno de los valores presentes en las variables numéricas:

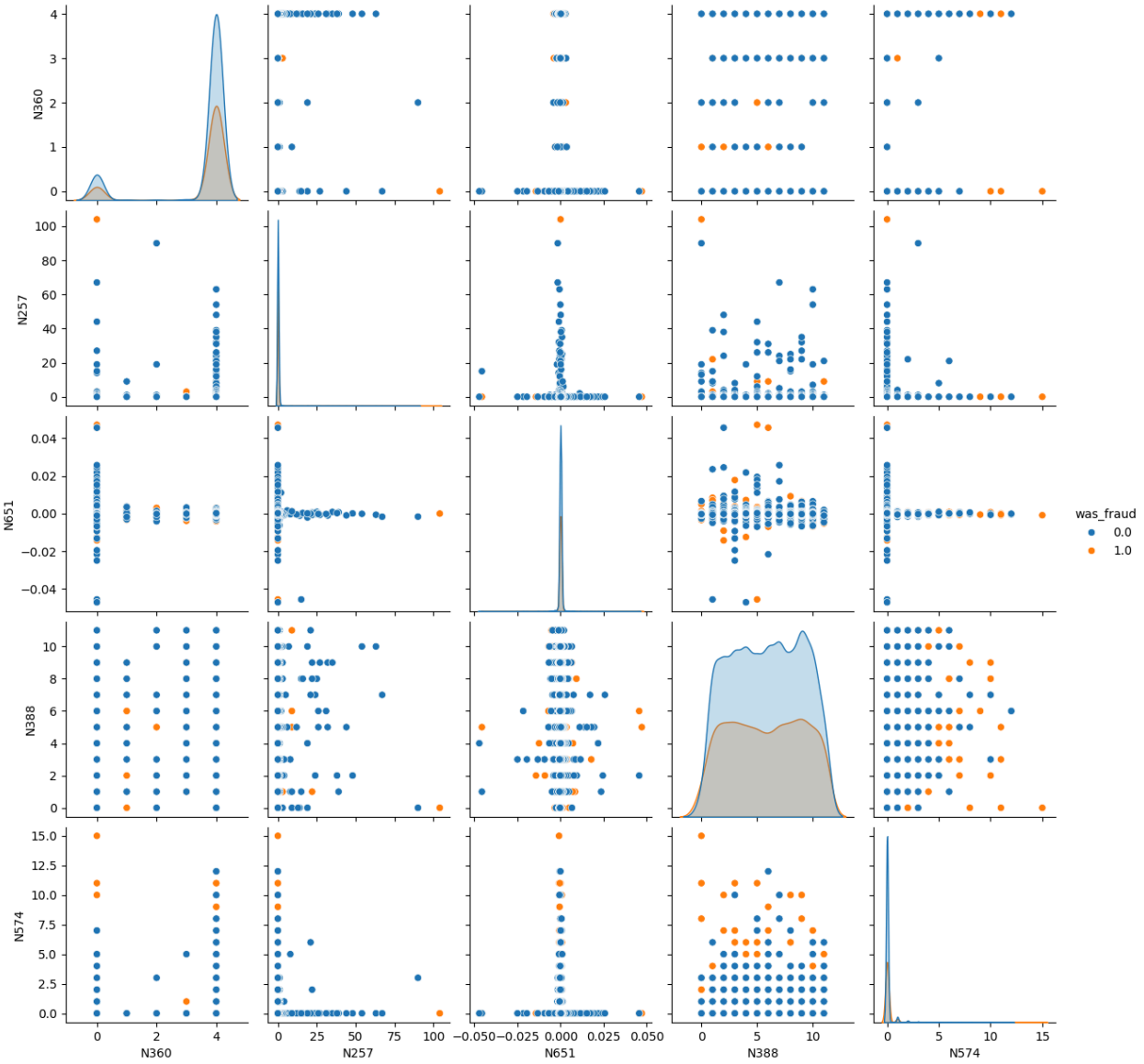
Tabla 3: Ejemplo variables numéricas

N155	N158	N159	N161	N166	N167	N188	N203	N225	...	N623	N624	N625	N645	N647	N650	N651	N654	N661	N662
4.0	-0.003100	0.004290	379.000000	0.212561	0.281981	221.42	20.0	786.0	...	1.0	1.0	1.0	0.001938	0.000306	0.000216	0.000283	475.0	0.0	0.0
0.0	-0.008446	-0.005802	258.916667	0.212561	0.281981	177.87	0.0	312.0	...	1.0	1.0	1.0	0.002214	0.000885	0.001020	0.000879	243.0	0.0	0.0
0.0	-0.001334	-0.004174	257.750000	0.212561	0.281981	220.91	0.0	524.0	...	1.0	1.0	1.0	-0.000243	0.000108	0.000028	0.000101	267.0	0.0	0.0
0.0	0.011478	-0.005792	193.583333	0.212561	0.281981	257.01	0.0	1491.0	...	1.0	1.0	1.0	-0.002185	-0.000572	-0.000408	-0.000633	342.0	0.0	16.0
0.0	0.010146	0.006548	257.833333	0.212561	0.281981	209.60	0.0	680.0	...	1.0	1.0	1.0	-0.001774	0.000070	0.000238	-0.000074	323.0	0.0	0.0

A continuación, se presenta la distribución de algunas de las variables numéricas seleccionadas luego de aplicar técnicas como variables altamente correlacionadas y análisis de componentes principales:

N360: Cantidad de consumos medidos entre: Tres meses antes de la fecha del inicio de la caída con ventana de tres meses, desde el mes 36 hasta el mes 24
N257: Cantidad de meses entre la fecha actual y la última facturación.
N651: Pendiente del consumo facturado desde el primer consumo más 5 meses hasta 1 mes
N388: Cantidad de consumos medidos entre: Tres meses antes de la fecha del inicio de la caída con ventana de tres meses, desde el mes 36 hasta el mes 24; hasta la fecha del inicio de la caída con ventana de tres meses, desde el mes 36 hasta el mes 24
N574: Cantidad Observaciones de posible fraude desde el primer consumo hasta 1 mes

Figura 2: Distribución variables numéricas relevantes

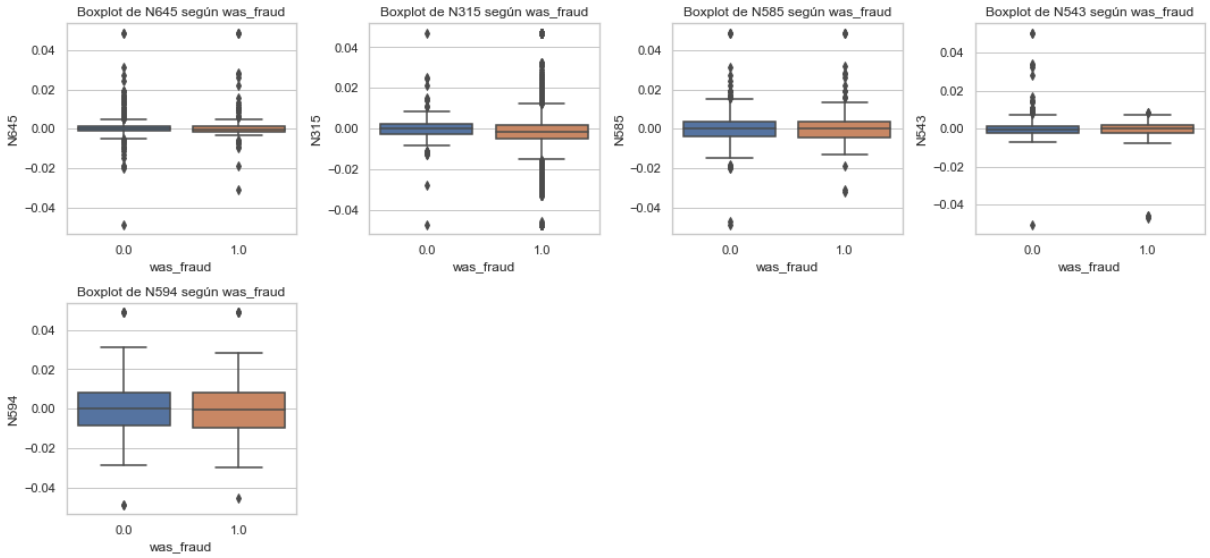


- Podemos observar que la cantidad de consumos medidos (N388) tiene una frecuencia bimodal, indicando el cambio de comportamiento del consumidor antes y después de la caída. Los puntos de fraude (naranja) parecen estar distribuidos de manera diferente en comparación con los puntos de no fraude (azul).
- Se observa que hay una dispersión considerable entre las variables de cantidad de observaciones de posible fraude y cantidad de consumos medidos (N574 y N388) y parece haber una ligera agrupación de puntos de fraude. Adicionalmente para la variable N574, hay algunos puntos de fraude con valores significativamente más altos en comparación con los puntos de no fraude

- La pendiente del consumo facturado (N651) tiene una distribución alrededor de cero, indicando que los cambios en el consumo facturado son mínimos para la mayoría de los casos.

Figura 3: Box plot variables pendientes de consumo

Variables
N645: Pendiente del consumo facturado entre Marzo 2020 y 35 meses atrás
N315: Pendiente consumo de los vecinos lógicos de la instalación (últimos 12 meses)
N585: Pendiente del consumo facturado entre Marzo 2020 y 11 meses atrás
N543: Pendiente del consumo facturado de los últimos 24 meses
N594: Pendiente del consumo facturado entre Marzo 2020 y 5 meses atrás

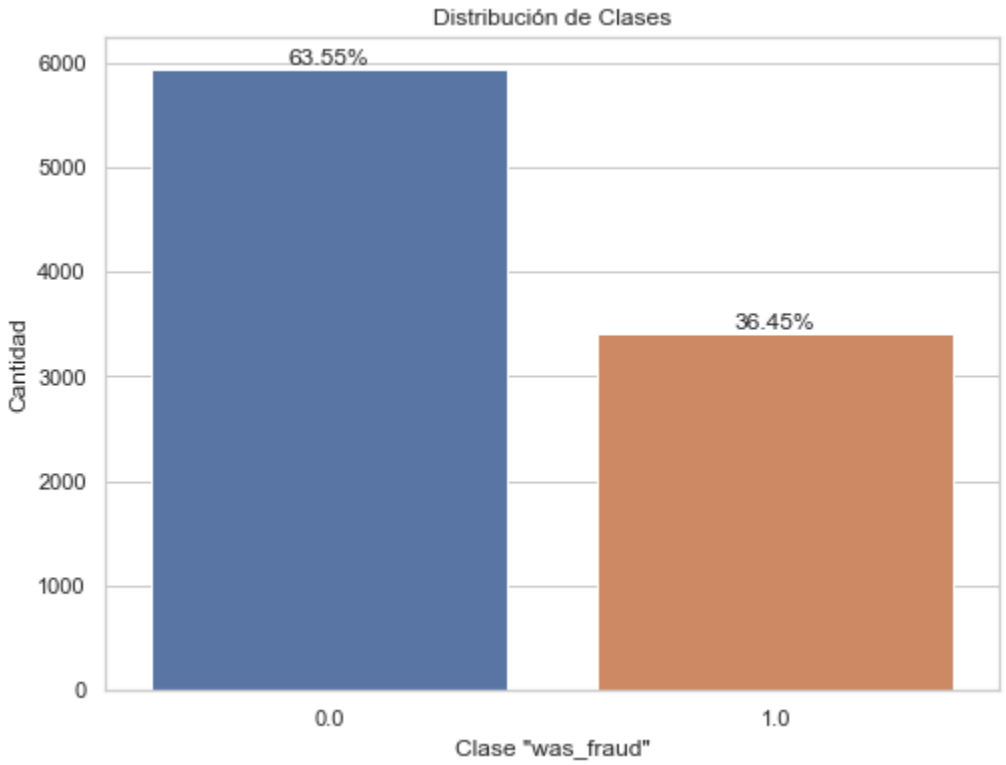


La cuarentena por COVID-19 parece haber aumentado la variabilidad en el consumo de los casos de fraude, reflejado en las variables N645, N585 y N543 y podrían estar influyendo en la capacidad de los modelos para discriminar entre fraude y no fraude. La variabilidad y presencia de outliers en los casos de fraude pueden ser más pronunciadas debido a la inestabilidad económica y las restricciones, llevando a comportamientos más extremos o anómalos.

Al tener una gran dimensionalidad, el reto durante la etapa de procesamiento se centrará en disminuir variables a través de tratamiento de valores nulos, eliminación de duplicados y variables numéricas con valores en 0 y remoción de aquellas variables que muestran una alta correlación.

Variable de salida:
was_fraud: Indica si la instalación es fraudulenta o no. Variable binaria 1 indica fraude y 0 para no fraude, se encuentran 5.943 (63,5%) casos de no fraude y 3.409 con fraude (36,4%):

Figura 4: Distribución de clases variable “was fraud”

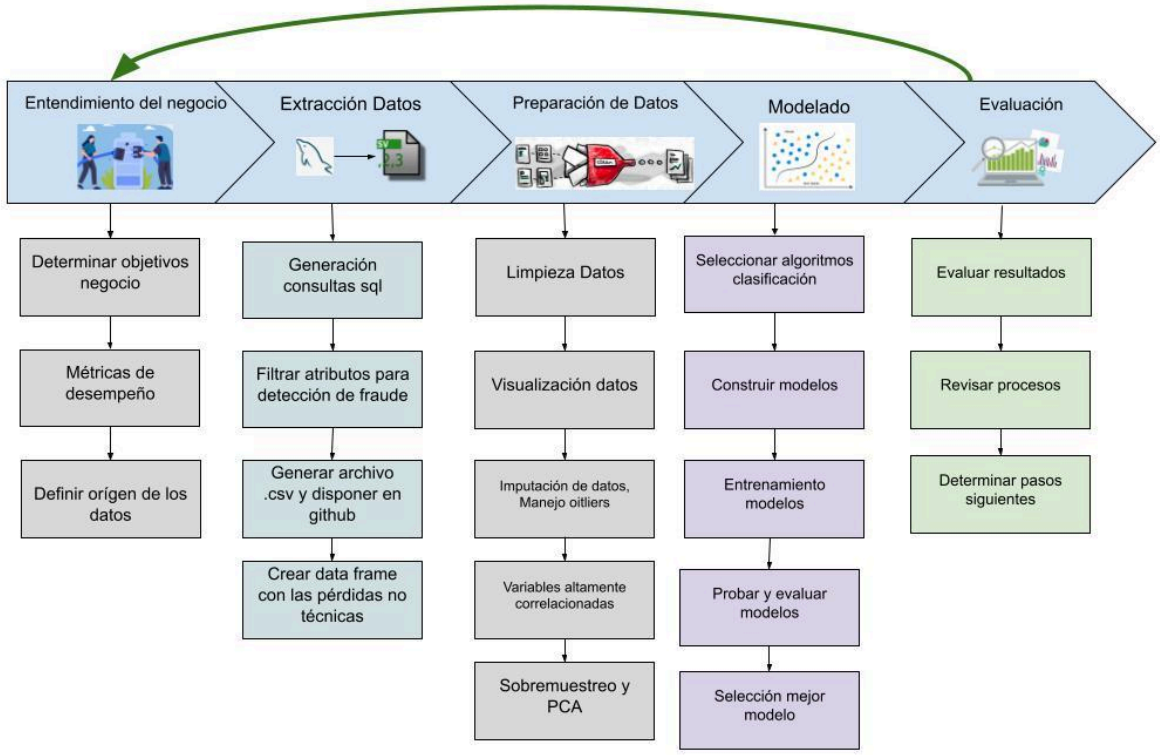


4. Proceso de analitica

4.1. Pipeline principal

El proceso de análisis de datos de este trabajo está basado en la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) (Chapman, n.d.). El pipeline está dividido en cinco fases principales: Entendimiento del Negocio, Extracción de Datos, Preparación de Datos, Modelado y Evaluación. A continuación se describe cada fase y sus actividades correspondientes:

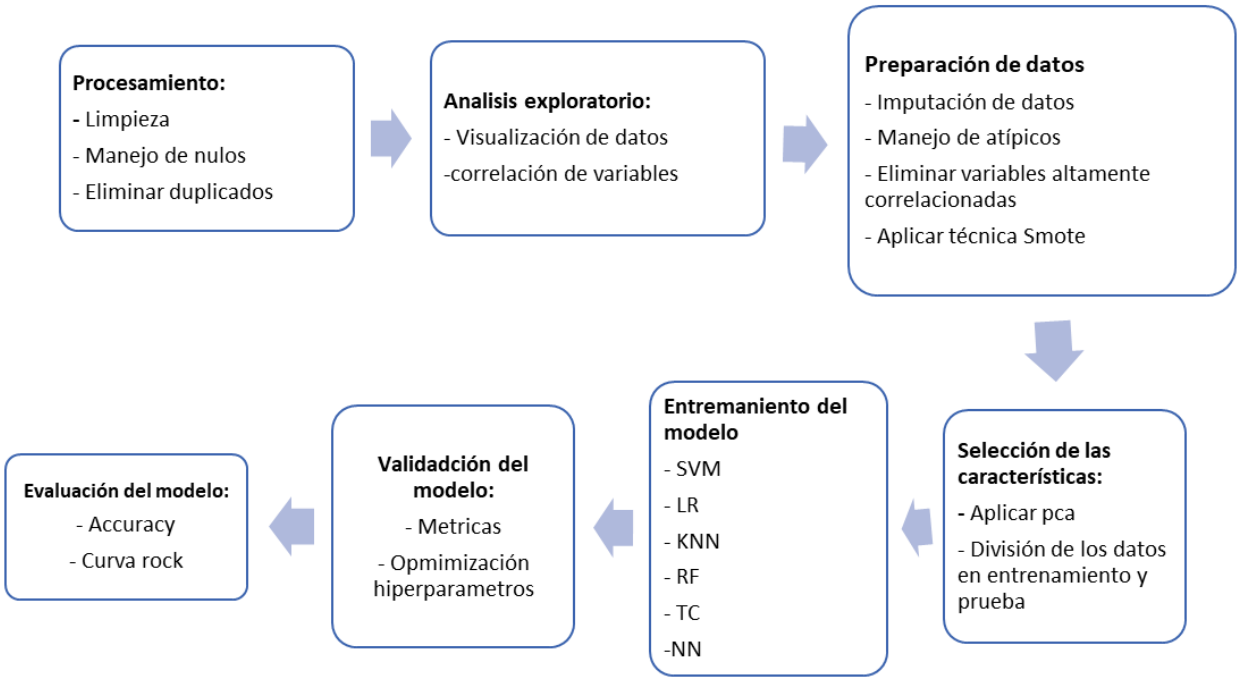
Figura 5: Pipeline principal



En la Figura 5 se observa que en la última fase se evalúa el modelo en relación a los objetivos del negocio, si los resultados son satisfactorios se considera realizar el paso a producción, en caso contrario se recomienda hacer otra iteración desde la fase de entendimiento del negocio y calibrar el modelo. En el alcance de este proyecto, nos enfocaremos hasta la fase de modelado. Para evaluar los resultados, será necesario realizar inspecciones en campo y comparar las diferentes métricas entre el negocio y las predicciones del modelo.

A continuación se presenta un desglose del preprocesamiento y el modelado de datos para más detalle:

Figura 6: Fases de procesamiento y modelado



4.2. Preprocesamiento

Se consideraron varias alternativas de preprocesamiento para abordar problemas como valores nulos, duplicados, datos atípicos y desequilibrio de clases. A continuación, se describen las principales estrategias y técnicas empleadas:

1. **Tratamiento de Valores Nulos**: Se opta por la imputación con la siguiente estrategia: Se revisaron las operaciones de cada atributo del suministro eléctrico, distinguiendo los siguientes:

   Cantidad: 135 columnas.

   Regresión Lineal: 17 columnas.

   Consumo: 175 columnas.

Correlación: 2 columnas.

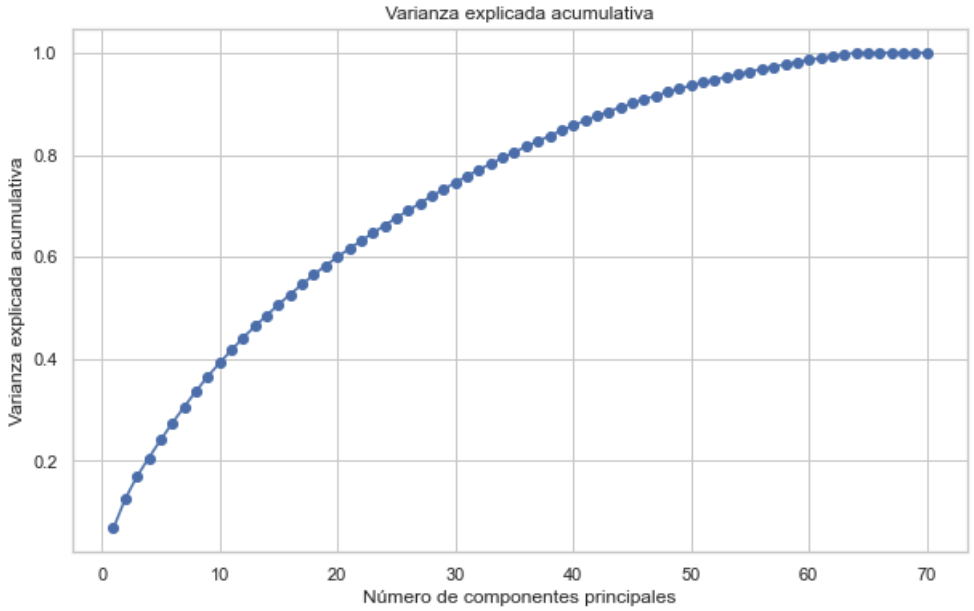
Percentiles: 2 columnas.

Desviación Estándar: 3 columnas.

En este caso, se optó por imputar con la mediana los atributos categorizados como Consumo y Cantidad debido a su naturaleza continua y para evitar la influencia de valores atípicos. Para el resto de las categorías, se optó por dejarles el valor 0, considerando que estas categorías representan valores calculados que, en ausencia de datos, pueden considerarse como no disponibles o insignificantes.

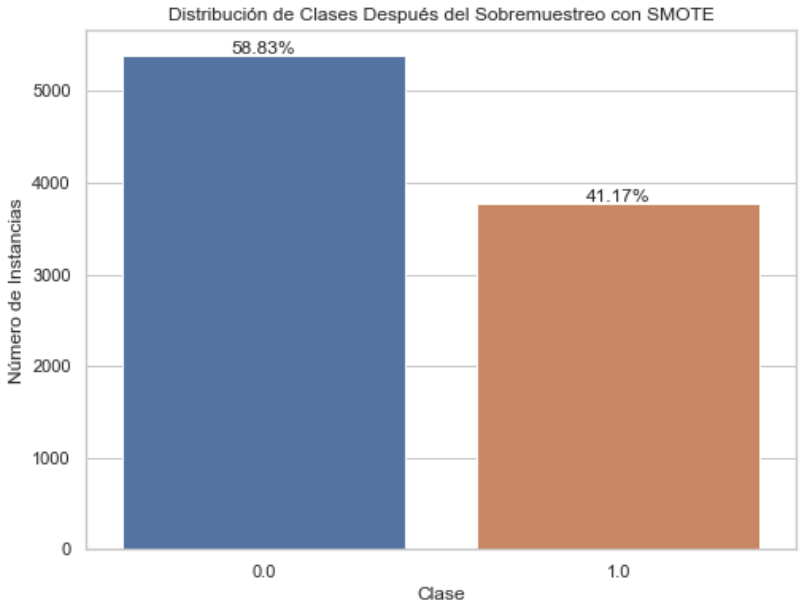
2. **Eliminación de Duplicados**: Se eliminan registros duplicados para asegurar que cada instancia en el conjunto de datos sea única y relevante.
3. **Eliminación de Variables con Valores en 0 o nulos**: Las variables numéricas que contenían más del 30% de valores faltantes se eliminaron. Además, se eliminaron las variables con todos los registros en ceros, ya que no aportan información relevante y podrían distorsionar los resultados.
4. **Tratamiento de Datos Atípicos**: Se utilizó el modelo LocalOutlierFactor (Scikit-learn -LocalOutlierFactor, n.d.) para identificar y eliminar valores atípicos en las variables de entrada. Después de una búsqueda de hiperparámetros para optimizar el valor de n_neighbors, se determinó que 10 vecinos más cercanos eran óptimos para la detección de valores atípicos.
5. **Remoción de Variables Altamente Correlacionadas**: Se identificaron las variables que tenían una correlación mayor al umbral predefinido (0,7) con cualquier otra variable. Posteriormente, estas variables fueron eliminadas para evitar problemas de multicolinealidad y reducir la dimensionalidad del conjunto de datos.
6. **Reducción del Conjunto de Datos**: Para reducir la dimensionalidad del conjunto de datos y capturar la mayor parte de la varianza en los datos originales, se aplicó el Análisis de Componentes Principales (PCA) (Jolliffe, 2002, 63). Este método permite transformar las variables originales en un conjunto de variables no correlacionadas llamadas componentes principales. Se seleccionó el número de componentes que explicaban al menos el 70-95% de la varianza total (60 componentes principales), asegurando así que se mantenía la mayor parte de la información relevante del conjunto de datos original, en el siguiente gráfico se observa como la varianza explicada acumulativa aumenta con el número de componentes principales y ayuda a determinar un número óptimo de componentes que retienen la mayor parte de la información relevante del conjunto de datos.

Figura 7: Gráfico de Varianza Explicada Acumulativa vs. Número de Componentes Principales



7. **Sobremuestreo**: Para abordar el desbalance en la variable objetivo was_fraud, se utilizó la técnica de sobremuestreo SMOTE (Synthetic Minority Over-sampling Technique) (Chawla & Bowyer, 2002). Este proceso consistió en aumentar la proporción de la clase minoritaria al 70% de la clase mayoritaria, generando un conjunto de datos medianamente balanceado.

Figura 8: Distribución clases con SMOTE



4.3. Modelos

Los modelos seleccionados permiten realizar un análisis supervisado, ideal para la detección de fraudes en el consumo de energía eléctrica mediante clasificación binaria.

4.3.1 Support Vector Machine (SVM):

Las Máquinas de Vectores de Soporte (SVM) son un conjunto de métodos de aprendizaje supervisado utilizados para clasificación, regresión y detección de valores atípicos. Entre sus ventajas se encuentran su efectividad en espacios de alta dimensionalidad, su capacidad para manejar casos donde el número de dimensiones es mayor que el número de muestras, su eficiencia en el uso de un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), y su versatilidad al permitir especificar diferentes funciones de kernel para la función de decisión. Sin embargo, tienen desventajas como la necesidad de evitar el sobreajuste al elegir funciones de kernel y términos de regularización, y la falta de estimaciones de probabilidad directas, que deben calcularse mediante una costosa validación cruzada de pliegues. (Géron, 2019, 153-174)

Para encontrar el hiperplano óptimo, se realizó validación cruzada con kernel polinomial, radial, lineal, sigmoide y el parámetro de regularización c con valores 0.1, 1 y 10. En este caso el mejor modelo fue el polinomial en donde la regularización moderada y la no linealidad en la función de decisión fueron importantes para el rendimiento del modelo:

- Mejores parámetros del modelo: *C=10, kernel='poly', probability=True*

4.3.2 Logistic Regression (LR):

La regresión logística es un modelo lineal utilizado para clasificación en lugar de regresión, según la nomenclatura de scikit-learn/ML. También conocida como regresión logit, clasificación de máxima entropía (MaxEnt) o clasificador log-lineal, esta técnica modela las probabilidades de los posibles resultados de un único evento mediante una función logística. Esta implementación puede ajustar regresiones logísticas binarias, Uno-contra-Resto o multinomiales con regularización opcional L1, L2 o Elastic-Net. (Scikit-learn LogisticRegression, n.d.)

El algoritmo utilizado para la validación cruzada fue *LogisticRegression* de la librería sklearn (*Scikit-learn LogisticRegression, n.d.*). Los hiperparámetros utilizados fueron los siguientes:

- **C**: Controla la fuerza de la regularización. Valores más pequeños indican una regularización más fuerte. Se probaron tres valores: 0.1, 1 y 10.
- **penalty**: Especifica el tipo de regularización a aplicar. Se probaron dos opciones: 'l1' y 'l2'. 'l1' corresponde a la regularización L1 (Lasso), que penaliza la suma de los valores absolutos de los coeficientes, mientras que 'l2' corresponde a la regularización L2 (Ridge), que penaliza la suma de los cuadrados de los coeficientes.
- **solver**: Este parámetro indica el algoritmo a usar en el problema de optimización. Se especificó 'liblinear', que es adecuado para problemas de clasificación binaria como la detección de fraudes.
- **class_weight**: Especifica cómo se manejarán los desequilibrios de clases. Se probaron dos opciones: None y 'balanced'. 'None' indica que no se aplicará ningún peso a las clases, mientras que 'balanced' da más peso a la clase minoritaria, lo que puede ser útil para problemas con desequilibrios en la distribución de las clases.

El mejor modelo fue aquel con regularización L1, un parámetro C moderado y utilizando el ajuste de pesos de clase 'balanced'. Esta configuración puede ayudar a prevenir el sobreajuste y a mejorar el rendimiento del modelo, especialmente en presencia de desequilibrios en los datos:

- Mejores parámetros del modelo: *C=1, class weight='balanced', penalty='l1', random_state=0, solver='liblinear'*

4.3.3 Modelo K-Nearest Neighbors (KNN):

Es un algoritmo de clasificación que predice la clase de una observación basándose en la mayoría de los “*k*” puntos más cercanos en el espacio de características. No requiere suposiciones sobre la distribución de los datos y es fácil de entender, aunque puede ser computacionalmente costoso con grandes conjuntos de datos. (*KNeighborsClassifier — Scikit-Learn 1.5.0 Documentation*, n.d.)

Para determinar el mejor rendimiento del modelo K-Nearest Neighbors (KNN), se utilizó el algoritmo *neighbors.KNeighborsClassifier* de *sklearn* , en donde exploraron las siguientes combinaciones de hiperparámetros:

- *'n_neighbors': [5, 10, 15, 20]* Número de vecinos utilizados en la clasificación
- *'weights': ['uniform', 'distance']:* Determina cómo se ponderan los vecinos cercanos durante la clasificación. Uniform: Todos los vecinos tienen el mismo peso en la votación. Distance: Los vecinos más cercanos tienen un peso mayor en la votación que los vecinos más lejanos, según la inversa de su distancia
- *'metric': ['euclidean', 'manhattan', 'chebyshev', 'minkowski']:* Métrica de distancia que se utilizará para calcular la distancia entre los puntos de datos
- *'leaf_size': [30, 40, 50]* Tamaño de las hojas en el árbol de búsqueda

    - Mejores parámetros del modelo: *metric='manhattan', n_neighbors=20, leaf_size=50*

4.3.4 Modelo Decision Tree Classifier (DTC)

Un árbol de decisión es un modelo de aprendizaje supervisado utilizado para clasificación y regresión. Este algoritmo divide el espacio de características en regiones simples basadas en umbrales de características. Cada división se realiza seleccionando la característica que mejor separa las clases en los datos; con el objetivo de maximizar la pureza de las clases en los nodos resultantes. Este modelo puede ser propenso al sobreajuste, especialmente en conjuntos de datos ruidosos o con muchos atributos. En tales casos, se utilizan técnicas como la poda de datos. (Géron, 2019, 175-186)

Para determinar el mejor rendimiento del modelo Decision Tree Classifier, se exploraron las siguientes combinaciones de hiperparámetros utilizando *tree.DecisionTreeClassifier* de *scikit-learn*:

- *criterion*: *['gini', 'entropy']* Criterio utilizado para medir la calidad de la división.
- *max_depth*: *[4, 6, 8]* Profundidad máxima del árbol. Se exploraron valores de 4, 6 y 8 para evitar un sobreajuste del mode*lo.*

Después de evaluar estas combinaciones de hiperparámetros, se determinó que el mejor rendimiento se obtuvo con los siguientes parámetros:

- Mejores parámetros del modelo: *criterion='entropy', max_depth=4*

4.3.5 Modelo Random Forest (RF)

Random Forest es un conjunto de árboles de decisión, donde cada árbol se entrena en una submuestra aleatoria del conjunto de datos y con un subconjunto aleatorio de características. Combina las predicciones de múltiples árboles de decisión para obtener una predicción final más robusta y generalizada. Es eficaz para evitar el sobreajuste y tiene una buena capacidad de generalización. (Géron, 2019, 197-211)

Para encontrar el mejor rendimiento del modelo Random Forest, se exploraron las siguientes combinaciones de hiperparámetros utilizando *RandomForestClassifier* de *scikit-learn*:

- *'n_estimators': [110, 117, 200]*. Número de árboles en el bosque
- *'max_depth': [5, 6, 8]*. Profundidad máxima de cada árbol
- *'min_samples_split': [2, 5, 10]*. Número mínimo de muestras necesarias para dividir un nodo
- *'min_samples_leaf': [1, 2, 4]*. Número mínimo de muestras requeridas para ser un nodo hoja
- *max_features': ['sqrt', 'log2'].* Cantidad de características a considerar en cada división
- *'class_weight': [None, 'balanced']*. Peso de clase para manejar desequilibrios

El mejor rendimiento se obtuvo con los siguientes parámetros:

- Mejores parámetros del modelo: *max_depth=8, max_features='log2', min_samples_split=5, n_estimators=200*

4.3.6 Modelo Multi-layer Perceptron classifier (MLPC)

Es un tipo de red neuronal artificial que consiste en múltiples capas de nodos, incluyendo una capa de entrada, una o más capas ocultas, y una capa de salida. Cada nodo (o neurona) en una capa está conectado a todos los nodos de la siguiente capa, y cada conexión tiene un peso ajustable. El MLP utiliza una función de activación no lineal, como la sigmoide o la ReLU, para modelar relaciones complejas en los datos. Durante el entrenamiento, del MLP se ajustan los pesos de las conexiones utilizando algoritmos como el *backpropagation* para minimizar el error entre las predicciones de la red y los valores reales, permitiendo así la clasificación de datos en diferentes categorías. (*MLPClassifier — Scikit-Learn 1.5.0 Documentation*, n.d.)

Para encontrar el mejor rendimiento del modelo MLPClassifier, se exploraron las siguientes combinaciones de hiperparámetros utilizando *MLPClassifier* de *scikit-learn*:

- *'hidden_layer_sizes': [(50,), (25,), (10,)]* Número de neuronas en cada capa oculta.
- *'alpha': [1e-5, 1e-4, 1e-3]* Tasa de regularización L2 (también conocida como parámetro de penalización). Se evaluaron valores de regularización de 0.00001, 0.0001 y 0.001 para mitigar el sobreajuste.

Los mejores hiperparámetros después de la evaluación de combinaciones son:

- Mejores parámetros del modelo: *alpha=0.00001, hidden_layer_sizes=(50,)*

4.4. Métricas

Utilizando el método hold-out (InteractiveChaos, n.d.), el dataset se dividió en 80% de datos para entrenamiento y 20% para prueba. Después de entrenar los modelos, se realizaron predicciones sobre el conjunto de prueba y se calcularon las siguientes métricas de desempeño utilizando la función de *matriz de confusión* de *scikit-learn* (*tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()*) (Sklearn confusion_matrix, n.d.):

- **Accuracy:** Mide la proporción de predicciones correctas (*true positives, true negatives, false positives, false negatives*).

$$Accuracy = (tp + tn)/(tp + tn + fp + fn)$$

- **Precision:** Indica la proporción de casos predichos como positivos que son verdaderamente positivos (fraude).

$$Precision = tp/(tp + fp)$$

- **Recall (Sensibilidad):** Muestra la proporción de casos positivos (fraude) que fueron correctamente identificados por el modelo.

$$Recall = tp/(tp + fn)$$

- **F1 Score:** Es una medida de precisión balanceada entre Precision y Recall.

$$F1\ Score = 2 * (Precision * Recall)/(Precision + Recall)$$

Una vez seleccionado el mejor modelo, se generó una matriz de confusión y un informe de clasificación (`classification_report`) (sklearn.metrics.classification_report, n.d.) específicamente para el modelo de Random Forest.

Además, se graficaron las curvas ROC para todos los modelos, calculando y mostrando el AUC-ROC (Area Under the Receiver Operating Characteristic Curve), junto con las tasas de FPR (False Positive Rate) y TPR (True Positive Rate). Un valor alto de AUC-ROC cercano a 1 indica una alta capacidad de discriminación, donde el modelo puede separar efectivamente las clases fraude y no fraude.

4.5. Sintonización de hiperparametros de los modelos

Inicialmente, se empleó *GridSearchCV* de la biblioteca *sklearn* para la selección de los mejores parámetros considerando en la validación cruzada. Sin embargo, posteriormente se optó por *RandomizedSearchCV* debido a su eficiencia computacional al explorar de manera aleatoria el espacio de hiperparámetros, lo que resulta en un tiempo de entrenamiento más rápido y escalable. Esta estrategia asegura la búsqueda de combinaciones óptimas de hiperparámetros para cada modelo, maximizando así su capacidad de detección de fraudes con la mayor precisión posible. A continuación se muestran las métricas obtenidas por cada uno de los enfoques, observando que no hay diferencia significativa:

Figura 9: Estrategia *GridSearchCV*

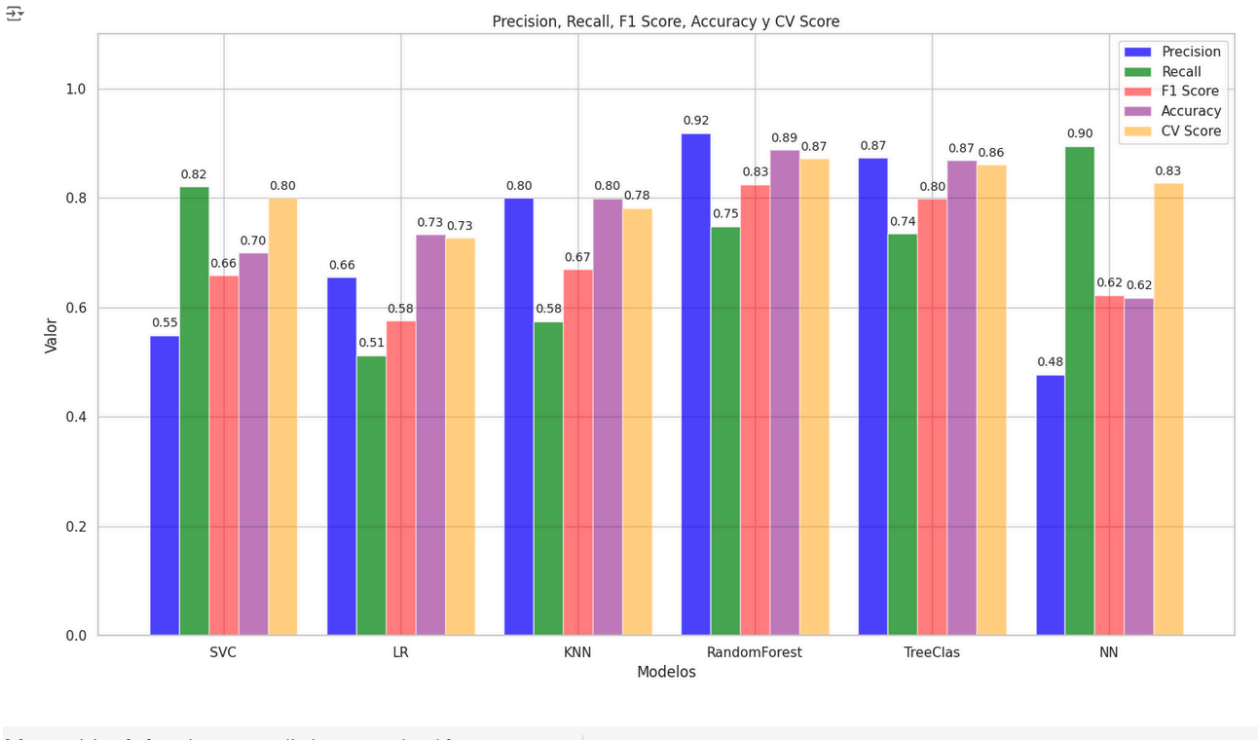
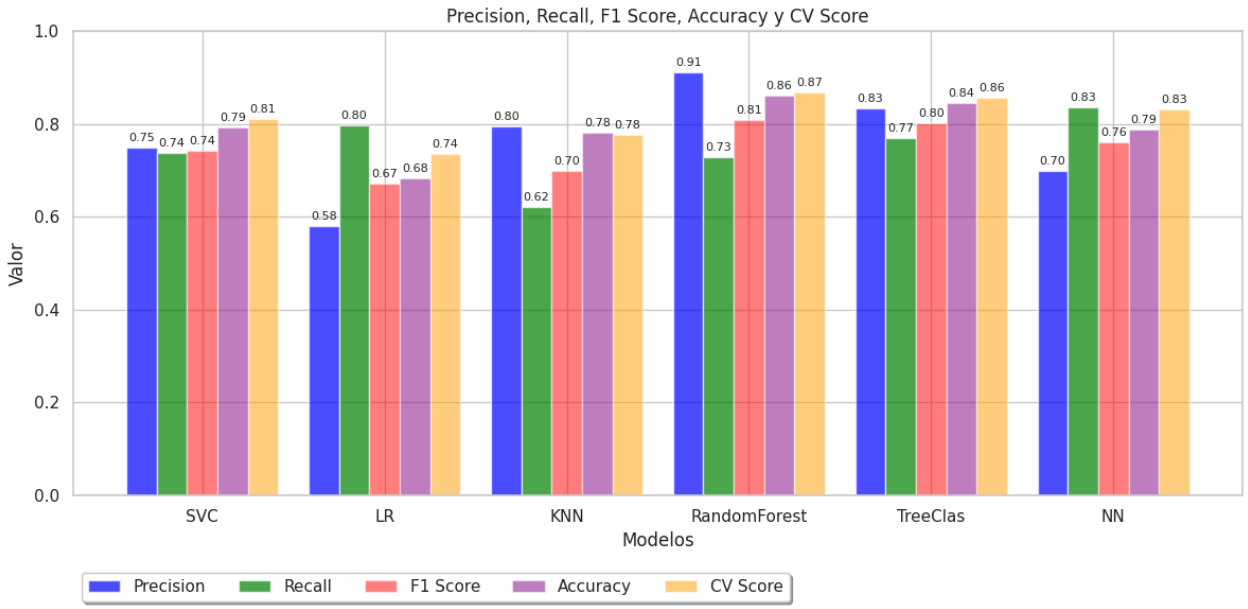


Figura 10: Estrategia *RandomizedSearchCV*



En las gráficas observamos las métricas de rendimiento de precision, recall, accuracy, f1_score y adicionalmente el cv_score, cuando se utiliza RandomizedSearchCV, se calcula el puntaje de precisión para cada combinación de hiperparámetros probados utilizando validación cruzada. El valor que se almacena en cv_score es el mejor puntaje de accuracy encontrado entre todas las combinaciones probadas.

En la Tabla 4 se presentan los componentes usados en cada experimento, la varianza explicada luego de aplicar pca al conjunto de datos y el porcentaje de muestras sintéticas usado en la clase minoritaria para cada iteración realizada. Esto es útil para comprender cuánta información se ha conservado después de la reducción de dimensionalidad. Luego de comparar las métricas en los resultados de cada modelo se observó que con 60 componentes se obtienen resultados satisfactorios y ahorro en tiempo de cómputo.

Tabla 4: Varianza explicada al aplicar PCA

Experimento	Varianza explicada (%)	componentes principales	Smote (%)	Distribución clase [0] (%)	Distribución clase [1] (%)
Iteración 1	99	62	0,70	58,83	41,17
Iteración 2	93	50	0,70	58,83	41,17
Iteración 3	**93**	**60**	**0,70**	**58,83**	**41,17**
Iteración 4	93	50	0,90	52,63	47,37
Iteración 5	93	50	0,70	58,83	41,17

En la Tabla 5 se presentan los resultados para cada modelo y las métricas usadas en su evaluación usando los componentes mencionados en la Tabla 4. En el experimento 3 se obtienen los mejores resultados en el modelo de Random Forest. Se dará mayor importancia al Recall o Sensibilidad para evitar los falsos negativos. Es decir las instalaciones fraudulentas que el modelo clasifica como legítimas.

Tabla 5: Resultados de los modelos

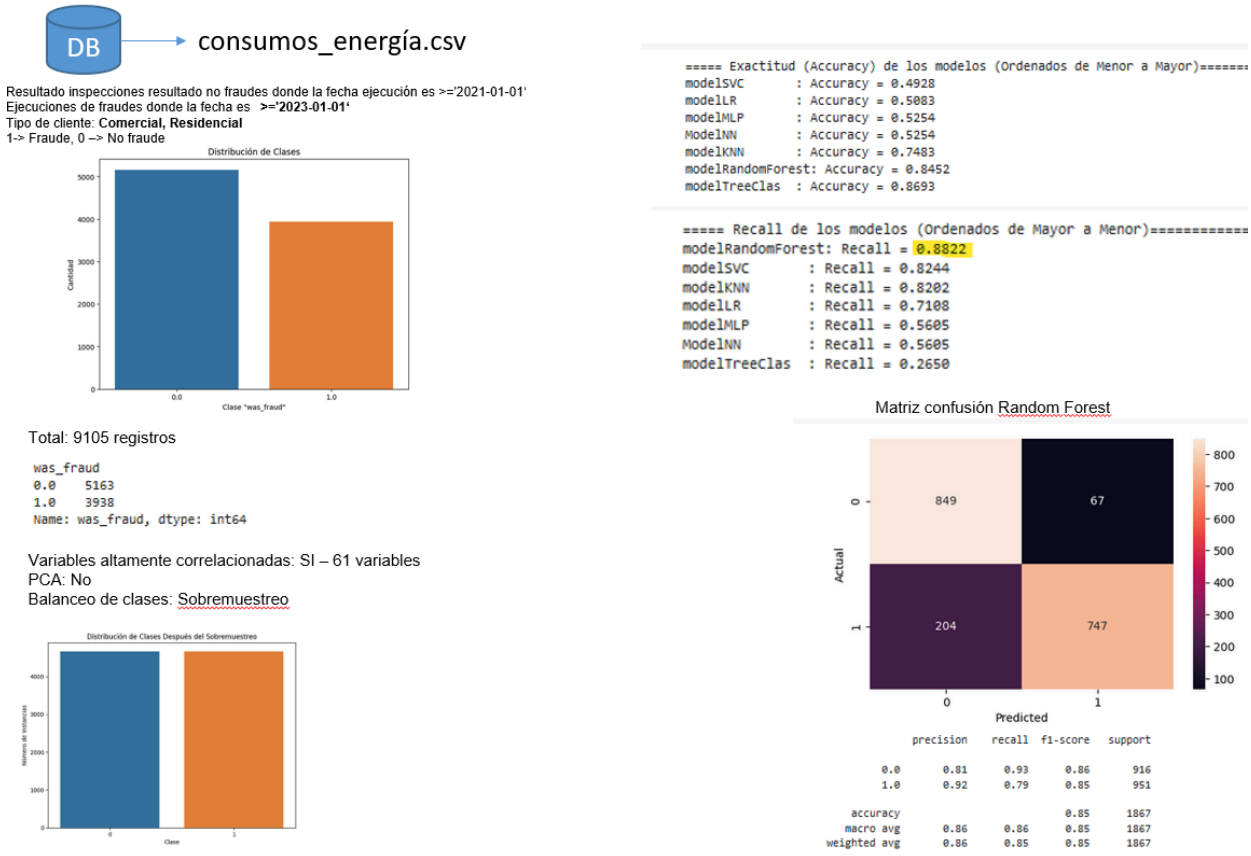
métrica/modelo	*SVC*	*LogisticRegression*	*KNeighbors*	*RandomForest*	*Decision Tree*	*MLPClassifier*
$Accuracy_{(1)}$	0,7340	0,6217	0,7577	0,8783	0,8694	0,7767
$Precision_{(1)}$	0,5887	0,4777	0,7330	0,9409	0,8737	0,6445
$Recall_{(1)}$	0,8111	0,7943	0,4907	0,6981	0,7352	0,8162
$F1\ score_{(1)}$	0,6823	0,5966	0,5879	0,8015	0,7985	0,7202
$Accuracy_{(2)}$	0,6085	0,6025	0,6897	0,6745	0,6298	0,6069
$Precision_{(2)}$	0,9643	0,8947	0,6336	0,7363	0,5791	0,9600
$Recall_{(2)}$	0,0363	0,0228	0,5578	0,3078	0,3199	0,0323
$F1\ score_{(2)}$	0,0699	0,0446	0,5933	0,4341	0,4121	0,0624
$Accuracy_{(3)}$	0,7001	0,7292	0,7559	**0,8884**	0,8694	0,6182
$Precision_{(3)}$	0,5497	0,6171	0,7420	**0,9193**	0,8737	0,4775
$Recall_{(3)}$	0,8212	0,6088	0,4705	**0,7487**	0,7352	0,8954
$F1\ score_{(3)}$	0,6586	0,6129	0,5759	**0,8253**	0,7985	0,6229
$Accuracy_{(4)}$	0,7892	0,7423	0,7951	0,8800	0,8622	0,7761
$Precision_{(4)}$	0,6776	0,6681	0,8054	0,8982	0,8737	0,6525
$Recall_{(4)}$	0,7656	0,5329	0,5514	0,7437	0,7116	0,7791
$F1\ score_{(4)}$	0,7189	0,5929	0,6547	0,8137	0,7844	0,7102
$Accuracy_{(5)}$	0,7595	0,7363	0,7910	0,8646	0,8640	0,7732
$Precision_{(5)}$	0,6247	0,6770	0,7946	0,9083	0,8625	0,6476
$Recall_{(5)}$	0,7943	0,4806	0,5481	0,6847	0,7302	0,7808
$F1\_score_{(5)}$	0,6993	0,5621	0,6487	0,7808	0,7909	0,7080

5. Metodología

5.1. Baseline

A continuación se describe la línea base para abordar la detección de fraudes en energía eléctrica, se encontró en la primera iteración del modelo que al tener un desbalance de clases se disparaban los falsos negativos, es decir los casos fraudulentos que el modelo clasificó como no fraude.

Figura 11: Primer iteración modelo



La primer iteración del proyecto de detección de fraude, se recolectaron datos de consumo de energía tanto de clientes residenciales como comerciales, el dataset inicial contenía un total de 9105 registros, con una distribución de clases desbalanceada: 5163 registros no fraudulentos y 3938 registros fraudulentos, como se presentan en la Figura 11.

Se eliminaron variables altamente correlacionadas para evitar la multicolinealidad. Además, se realizó un sobremuestreo de clases utilizando el método *RandomOverSampler* para balancear el conjunto de datos.

Se generaron diferentes modelos de clasificación, en donde Random Forest fue el que obtuvo mejores resultados:

- Exactitud en el entrenamiento: 0.84
- Exactitud en la prueba: 0.86

Los siguientes fueron los inconvenientes técnicos a resolver:

- Buscando mejorar la proporción de datos y tratar de balancear los datos desde el origen, se optó por manejar únicamente datos de clientes residenciales. Esto se debe a que el fraude en el consumo de energía puede manifestarse de manera diferente en contextos residenciales y comerciales. Al enfocarse en clientes residenciales, se puede desarrollar y ajustar el modelo específicamente para detectar fraudes típicos en este segmento. Posteriormente, se podría extender el análisis y la modelización a clientes comerciales con un enfoque adaptado a sus características particulares.
- El método *RandomOverSampler* se utilizó para equilibrar las clases en el conjunto de datos mediante la replicación de muestras de la clase minoritaria. Sin embargo, este enfoque generó muestras sintéticas que no reflejaban de manera realista la distribución de la clase minoritaria, lo que resultó en sobreajuste del modelo y en la pérdida de información relevante.
- Aunque se eliminaron variables altamente correlacionadas para reducir dimensionalidad, se observó redundancia en las características restantes.
- La ejecución de los modelos tardaba un tiempo considerable en finalizar. más de 50 minutos.

5.2. Validación

Se dividió el conjunto de datos en dos partes, 80% para entrenamiento y 20% para prueba. Al conjunto de entrenamiento se aplicó las funciones *GridSearchCV* y *RamdomizedSearchCV* de la librería *scikit-learn* ambos implementan la validación cruzada *k-fold* con 5 pliegues para evaluar el desempeño del modelo y encontrar la mejor combinación de hiperparámetros. También se definió que cada pliegue tenga una distribución similar de clases al conjunto de datos original mediante la clase *stratifiedkFold* es decir una validación cruzada estratificada. Para garantizar la reproducción de los datos se usó la misma semilla en cada experimento.

5.3. Iteraciones y evolución

Se realizaron varias iteraciones, comenzando con el refinamiento del preprocesamiento el desafío en esta etapa era reducir el número de variables en el conjunto de datos sin perder una cantidad significativa de la información original, primero se probó eliminando variables con alta multicolinealidad utilizando Factor de Inflación de la Varianza (VIF), esto redujo de manera

significativa el set datos a 71 variables. En una iteración posterior, se intentó además aplicar Análisis de Componentes Principales (PCA), lo cual resultó en un conjunto de 60 variables.

Inicialmente, se identificó un desbalance significativo en los datos, con un 82% de muestras pertenecientes a la clase no fraude y solo un 18% a la clase fraude. Se implementó un filtrado directo de los datos desde la fuente, limitando los casos a un año para la clase no fraude y dos años para la clase fraude, pero aún persistía un desbalance del 63% y 37%, respectivamente.

Se intentó abordar el desbalance mediante sobremuestreo, utilizando la técnica RandomOverSampler para aumentar el número de muestras de la clase minoritaria (fraude) para igualar el número de muestras con la clase mayoritaria (no fraude). Sin embargo, esta estrategia no produjo mejoras significativas en el rendimiento del modelo.

Finalmente, se optó por aplicar el algoritmo SMOTE (Synthetic Minority Over-sampling Technique). En contraste con la generación de datos dummies, SMOTE genera nuevas muestras para la clase minoritaria (fraude) tomando en cuenta los vecinos más cercanos en el espacio de características. Esta técnica fue seleccionada por su capacidad para equilibrar el conjunto de datos al proporcionar ejemplos sintéticos que reflejan mejor la distribución subyacente de la clase minoritaria. Este enfoque ayudó a mejorar la representación de la clase fraude en el conjunto de datos, lo cual es fundamental para reducir el sesgo del modelo hacia la clase mayoritaria y mejorar la capacidad predictiva en la detección de fraudes.

La transición de Grid Search a Random Search permitió una búsqueda más eficiente y escalable de hiperparámetros. Esto llevó a encontrar configuraciones que optimizaron el rendimiento del modelo sin comprometer significativamente la precisión.

5.4 Herramientas

MySQL: Base de datos relacional utilizada como fuente principal de datos para el proyecto. Desde MySQL se extrajeron los datos necesarios para su posterior análisis y modelización.

Google Colaboratory: Plataforma basada en la nube que proporciona acceso gratuito a recursos de cómputo, incluyendo GPU y TPU. Google Colaboratory fue utilizado para desarrollar y ejecutar código en cuadernos de Jupyter, facilitando la realización de experimentos y análisis de datos de manera colaborativa y escalable.

GitHub: Plataforma de desarrollo colaborativo basada en Git, utilizada para el control de versiones del código y la gestión de los datos del proyecto. GitHub permitió almacenar de manera segura y gestionar los cambios en el código y conjuntos de datos a lo largo del desarrollo del proyecto.

Librerías Utilizadas para el Tratamiento, Visualización y Modelización de Datos:

- **pandas:** Para la manipulación y análisis de datos tabulares.

- **numpy:** Para operaciones numéricas eficientes y manipulación de arrays.
- **matplotlib y seaborn:** Para la visualización de datos, incluyendo la creación de gráficos estadísticos y visuales que facilitan la comprensión de los patrones en los datos.

Librerías Utilizadas para la Modelización y Evaluación de Datos:

- **scikit-learn (sklearn):** Proporcionó un conjunto amplio de herramientas para el modelado de datos, así como funciones para la evaluación de modelos y la optimización de hiperparámetros.

6. Resultados y discusión

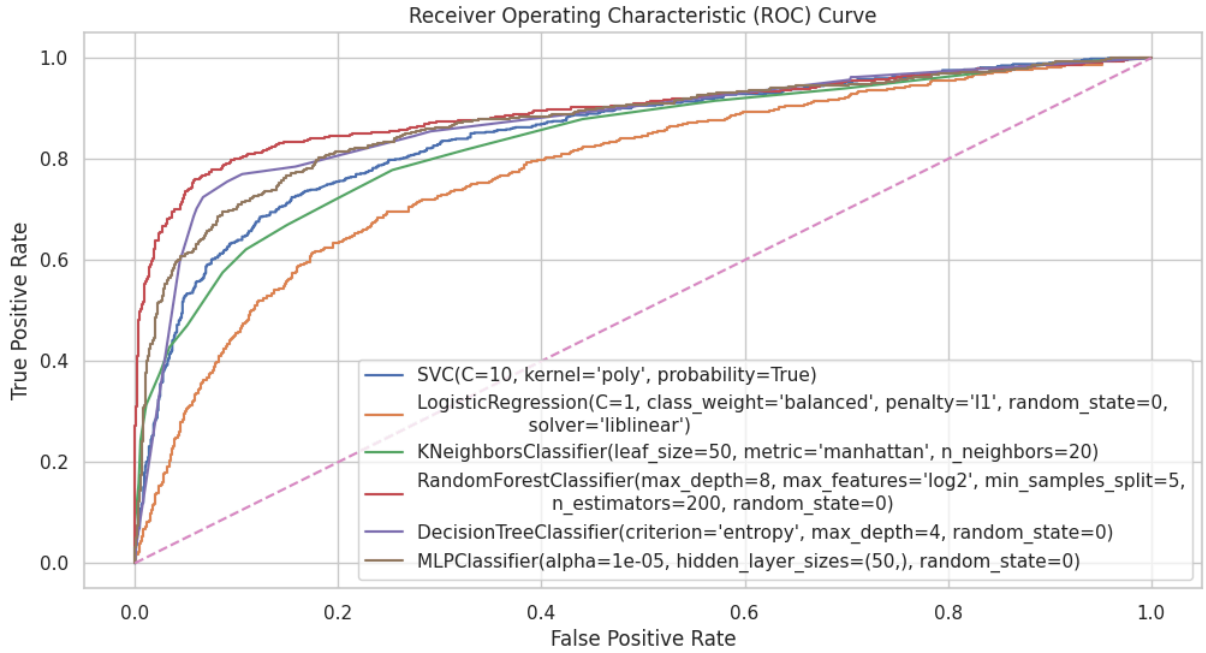
6.1. Métricas

- **Gráfico de Curva ROC**

El gráfico de la curva ROC proporcionado visualiza las tasas de verdaderos positivos (TPR) frente a las tasas de falsos positivos (FPR) para cada modelo evaluado, permitiendo una comparación visual del rendimiento de los modelos.

Figura 12: Curva ROC modelos clasificación

```
0.85: SVC(C=10, kernel='poly', probability=True)
0.78: LogisticRegression(C=1, class_weight='balanced', penalty='l1', random_state=0,
                  solver='liblinear')
0.84: KNeighborsClassifier(leaf_size=50, metric='manhattan', n_neighbors=20)
0.90: RandomForestClassifier(max_depth=8, max_features='log2', min_samples_split=5,
                      n_estimators=200, random_state=0)
0.87: DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=0)
0.87: MLPClassifier(alpha=1e-05, hidden_layer_sizes=(50,), random_state=0)
```

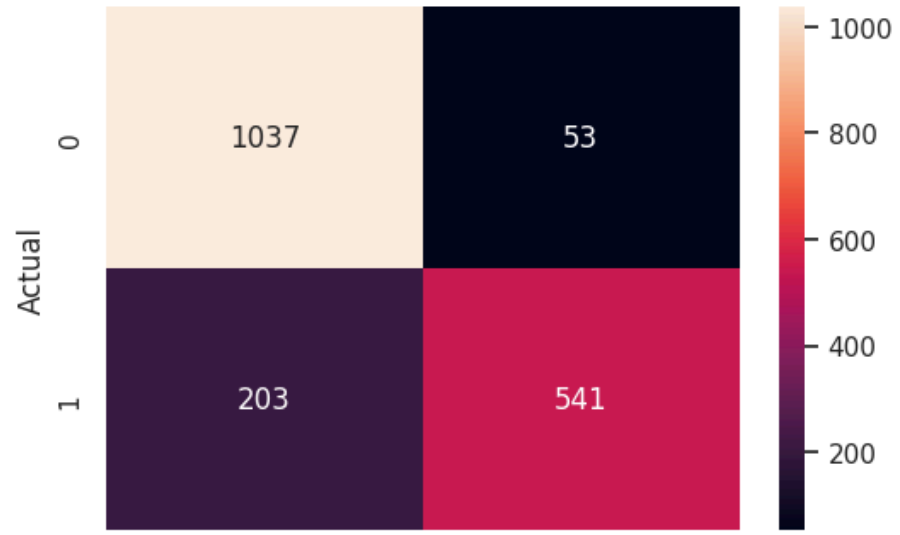


El modelo RandomForestClassifier tiene el mejor desempeño con un AUC-ROC de 0,90, seguido por el DecisionTreeClassifier con un AUC-ROC de 0,87 y el MLPC con un AUC-ROC de 0,87. Estos resultados sugieren que los modelos basados en árboles (Random Forest y Decision Tree) son los más efectivos a la hora de detectar fraudes en energía eléctrica.

- **Matriz de confusión**

El modelo *RandomForestClassifier* clasificó correctamente 1037 casos de no fraude y 541 casos de fraude. Clasificó erróneamente 53 casos como fraudulentas cuando en realidad eran no fraudulentas es decir los falsos positivos y 203 casos donde el modelo no logró identificar instalaciones fraudulentas, es decir los falsos negativos. Esto se evidencia en la tasa de recuperación *Recall* con un 73% para los casos donde hay fraude y el modelo no identifica todos los casos en su totalidad.

Figura 13: Matriz de confusión modelo Random Forest



	precision	recall	f1-score	support
0.0	0.84	0.95	0.89	1090
1.0	0.91	0.73	0.81	744
accuracy			0.86	1834
macro avg	0.87	0.84	0.85	1834
weighted avg	0.87	0.86	0.86	1834

6.2. Evaluación cualitativa

Según los resultados de la métricas de exactitud y F1-Score presentados en la Figura 14 y Figura 15, se observa que los modelos Support Vector Classifier (SVC), K vecinos más cercanos (KNN), y red neuronal MLP (NN), muestran signos de posible sobreajuste debido a diferencias notables entre las métricas de entrenamiento y prueba. Por otro lado, Random Forest y Decision Tree Classifier (TeClas) muestran un mejor equilibrio entre el rendimiento en entrenamiento y prueba, indicando una mejor capacidad de generalización, este rendimiento se logra ajustando los hiperparametros (profundidad máxima, número mínimo de muestras por nodo) y utilizando la validación cruzada con 5 pliegues para encontrar los mejores hiperparametros.

Figura 14: Exactitud de los modelos para el conjunto de entrenamiento y prueba

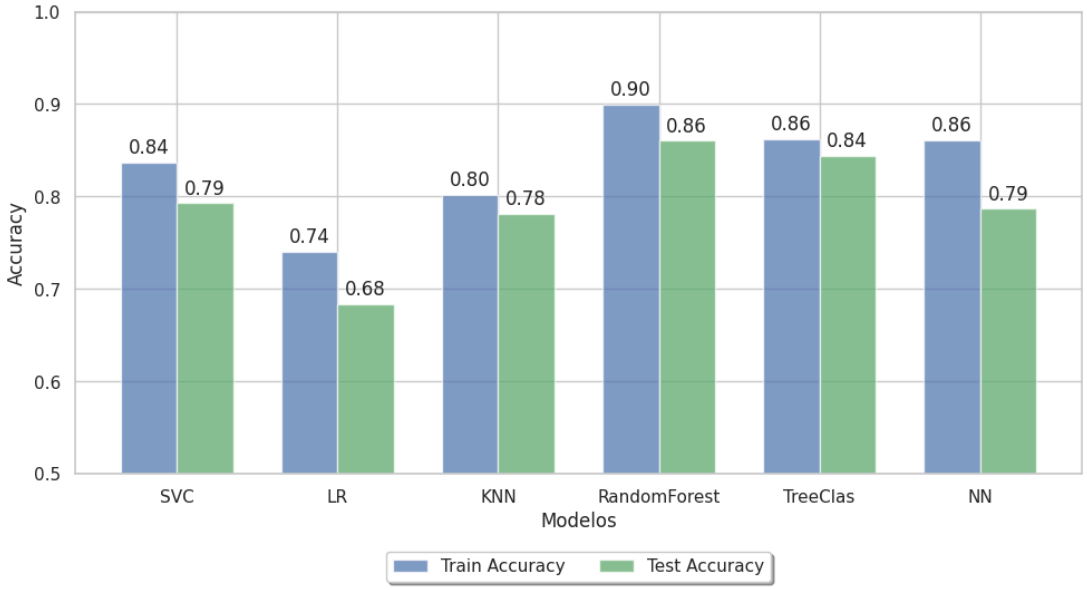
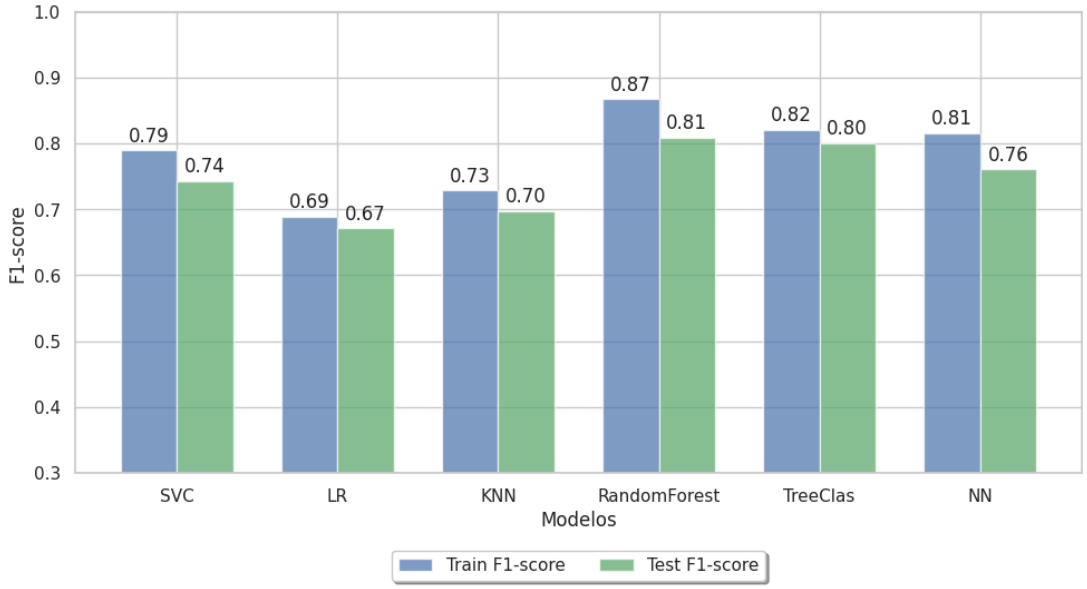


Figura 15: F1-Score de los modelos para el conjunto de entrenamiento y prueba



6.3. Condiciones de producción

- Los modelos seleccionados tienen el potencial de ser afinados aún más al manejar diferentes umbrales de clasificación, especialmente enfocados en maximizar el recall ya que es primordial para el negocio maximizar los positivos reales. Además, sería beneficioso desarrollar un modelo adicional de detección de fraudes específicamente para el sector de clientes comerciales.
- La integración de este nuevo modelo con la herramienta actual de detección de pérdidas sería clave para proporcionar una herramienta más robusta y efectiva a los expertos de negocio. Esto les permitiría tomar decisiones más informadas y estratégicas en la gestión de riesgos y la prevención de fraudes.
- Es fundamental realizar un monitoreo constante de las métricas de negocio en relación con los resultados del clasificador. Esto proporcionará una evaluación continua del rendimiento y la efectividad del modelo, permitiendo ajustes y mejoras continuas para optimizar su desempeño en tiempo real.

7. Conclusiones

- Para la detección de fraudes en el sistema eléctrico, se han identificado los modelos *Random Forest* y *Decision Tree Classifier* como los más efectivos, destacándose por su capacidad para equilibrar la precisión en la identificación de fraudes con la minimización de falsos positivos. Este equilibrio es fundamental en la gestión de fraudes, ya que el objetivo no solo es detectar transacciones fraudulentas, sino también reducir las alarmas innecesarias, lo que mejora significativamente la eficiencia operativa al momento de enviar cuadrillas para la inspección de instalaciones.

- A través del análisis exploratorio y la aplicación de técnicas para detectar variables altamente correlacionadas, se ha identificado una relación significativa entre los patrones de consumo de los clientes y la detección de fraudes. Se observa que varias de las variables seleccionadas muestran patrones de caída en ciertos momentos, lo cual sugiere comportamientos anómalos en el consumo de energía que pueden estar asociados con actividades fraudulentas. Además, se ha destacado que el impacto de los consumos de los vecinos más cercanos desempeña un papel crucial en la detección de fraudes. Esto indica que las variaciones en el consumo de energía de los hogares cercanos pueden influir significativamente en la identificación de irregularidades en el consumo.

- La limpieza de datos desempeña un papel fundamental en la preparación del conjunto de datos para el análisis. Dedicamos un tiempo considerable a esta etapa. Durante este proceso, redujimos significativamente la cantidad de variables numéricas y eliminamos duplicados y variables altamente correlacionadas, lo que contribuyó a simplificar el conjunto de datos. Además, la preparación de datos mejoró notablemente el rendimiento de los modelos que evaluamos.

8. Recomendaciones

- Dado que en este estudio se analizaron los datos recopilados durante un periodo de dos años. Aumentar la cantidad de registros en la base de datos dará una mayor robustez al modelo al proporcionar una mayor diversidad de escenarios y situaciones para entrenar y validar, lo cual es crucial para mejorar la precisión y la fiabilidad de las predicciones.

- Implementar combinaciones de características y técnicas avanzadas de feature engineering puede enriquecer significativamente los datos y mejorar el rendimiento predictivo de los modelos. En este ejercicio se imputó los valores faltantes con la mediana. Se recomienda explorar la creación de nuevas características derivadas de las existentes. Por ejemplo, considerar la combinación de características relacionadas temporalmente, como el promedio de medias móviles o diferencias porcentuales en consumo de energía. Estas nuevas características podrían capturar mejor las relaciones en los datos de entrenamiento y mejorar la capacidad de los modelos para detectar patrones complejos.

- Explorar otras alternativas para realizar el balance de las clases, como por ejemplo redes generativas adversativas, GANs, que se utilizan en la creación de nueva información.

- Dado que la red neuronal simple empleada en este estudio arrojó resultados satisfactorios en el entrenamiento y prueba de las métricas de *precisión* y *recall*, se sugiere explorar el uso de una red neuronal con una arquitectura más avanzada. Esta red podría mejorar aún más el rendimiento del modelo en términos de precisión y recall.

- Además, sería beneficioso comparar sistemáticamente las métricas de desempeño entre el Random Forest al optimizarlo y la red neuronal compleja propuesta, evaluando si la complejidad adicional del modelo de red neuronal justifica la mejora en términos de precisión y recall. Esta comparación permitirá determinar la mejor estrategia para la detección efectiva de instalaciones fraudulentas en el contexto específico del problema abordado.

Referencias

Carrillo, A. (2019, 10 5). *Sistema inteligente basado en Machine Learning para la detección de fraude de facturación de agua potable*. Repositorio Institucional de Tesis y Trabajos de Investigación. Retrieved 11 7, 2023, from https://cybertesis.unmsm.edu.pe/

Chapman, P. (n.d.). CRISP-DM 1.0. Retrieved June 13, 2024, from https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2012-13/kdd/files/CRISPWP-0800.pdf

Chawla, N., & Bowyer, K. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*.

Comisión de Regulación de Energía y Gas [CREG]. (2018). *Alejandría - Resolución 15 de 2018 CREG*. Gestor normativo CREG. Retrieved June 13, 2024, from https://gestornormativo.creg.gov.co/gestor/entorno/docs/resolucion_creg_0015_2018.htm

CREG - Comisión de Regulación de Energía y Gas. (n.d.). *Alejandría - Resolución 172 de 2011 CREG*. Gestor normativo CREG. Retrieved June 12, 2024, from https://gestornormativo.creg.gov.co/gestor/entorno/docs/resolucion_creg_0172_2011.htm#2

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly.

InteractiveChaos. (n.d.). *Método de retención*. Interactive Chaos. Retrieved June 14, 2024, from https://interactivechaos.com/es/manual/tutorial-de-machine-learning/metodo-de-retencion

Jolliffe, I.T. (2002). *Principal Component Analysis*. Springer.

KNeighborsClassifier — scikit-learn 1.5.0 documentation. (n.d.). Scikit-learn. Retrieved June 14, 2024, from

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Ley 599 de 2000 Congreso de la República de Colombia. (2000, July 24). Secretaría General de la Alcaldía Mayor de Bogotá. Retrieved June 12, 2024, from https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=6388&dt=S

LogisticRegression — scikit-learn 1.5.0 documentation. (n.d.). Scikit-learn. Retrieved June 14, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Martínez, C., & Vilalta, J. (2022, 12 10). Métodos y técnicas de Machine Learning e Inteligencia Artificial para el enfrentamiento al fraude en las Telecomunicaciones: Técnicas de minería de datos aplicadas a las gestión del fraude. *Revista Cubana de Transformación Digital*, *3*(4), 10. https://rctd.uic.cu/rctd/article/view/212/101

MLPClassifier — scikit-learn 1.5.0 documentation. (n.d.). Scikit-learn. Retrieved June 14, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Navlani, A., Fandango, A., & Idris, I. (2021). *Python Data Analysis: Perform Data Collection, Data Processing, Wrangling, Visualization, and Model Building Using Python*. Packt Publishing. https://books.google.com.co/books?id=DN4SEAAAQBAJ&pg=PA25&hl=es&source=gbs_selected_pages&cad=1#v=onepage&q&f=false

Nogueira, L. G., & K, A. C. (2017). *SMOTE — Version 0.12.3*. Imbalanced-learn. Retrieved June 13, 2024, from https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Scikit-learn -LocalOutlierFactor. (n.d.). *LocalOutlierFactor — scikit-learn 1.5.0 documentation*. Scikit-learn. Retrieved June 13, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

Scikit-learn LogisticRegression. (n.d.). *LogisticRegression — scikit-learn 1.5.0 documentation*. Scikit-learn. Retrieved June 13, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn confusion_matrix. (n.d.). *confusion_matrix — scikit-learn 1.5.0 documentation*. Scikit-learn. Retrieved June 14, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

sklearn.metrics.classification_report. (n.d.). *sklearn.metrics.classification_report — scikit-learn 0.15-git documentation*. Scikit-learn. Retrieved June 14, 2024, from https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.classification_report.html

Anexos

Repositorio notebook:

https://github.com/SusanaAlvarezC/Monografia