APLICACIÓN DE UN MODELO DE CLASIFICACIÓN DE CATEGORÍAS FISCALES DE UNA EMPRESA DE MANUFACTURACIÓN Y COMERCIALIZACIÓN DE DISPOSITIVOS BIOMÉDICOS

Equipo de trabajo:

Karolina Arrieta Salgado
Ingeniera Química

Viviana Idárraga Ojeda
Química

Matías Camelo Valera
Ingeniero Mecánico – Aeronáutico

Stevens Restrepo Vallejo
Psicólogo

Andrés Puerta González
Biólogo

Maestría en Ciencia de Datos y Analítica
Universidad EAFIT
2023-II

Tabla de contenido

1. Introducción

Los impuestos indirectos son según el Internal Revenue Service of the United States (IRS)[1] "un impuesto que se puede dirigir hacia los otros, tal como el impuesto de propiedad sobre el negocio". Esto implica, la asignación de impuestos sobre cosas como propiedad raíz, automóviles, inventario de la compañía y maquinaria o servicios de software propios de la empresa. Según EY, una de las más grandes consultoras a nivel mundial en la materia. esto generaría otros impuestos como: impuesto al valor añadido (IVA), Impuesto sobre bienes y servicios, impuesto sobre las ventas, impuestos sobre el uso de inventario y maquinaria propiedad de la compañía[2]. En este contexto, en aras de proporcionar una categoría fiscal a cada uno de los dispositivos biomédicos contenidos en el conjunto de datos de este proyecto, se propone evaluar modelos de clasificación para definir el clasificador que mejor asigne la clase o categoría de impuesto, a partir de la comprensión de patrones presentes en un conjunto de características asignadas a cada dispositivo.

En el ámbito del Machine Learning (ML), los problemas de clasificación son esenciales y tienen un impacto significativo en diversas disciplinas, desde la automatización de decisiones hasta la medicina. La clasificación implica prever respuestas cualitativas asignando observaciones a categorías específicas. Diversas estrategias de ML abordan problemas de clasificación, como tareas binarias, de múltiples clases, de múltiples etiquetas y jerárquicas. Entre las técnicas de clasificación, el uso de PCA como preprocesamiento destaca por su capacidad para reducir la dimensionalidad y eliminar la correlación entre atributos. Además, la clasificación de conjuntos de datos no balanceados presenta desafíos, abordados por enfoques a nivel de datos, como SMOTE, que genera instancias sintéticas para contrarrestar la falta de representación de la clase minoritaria. A nivel de algoritmo, métodos de ensamble como Random Forest y Boosting se destacan por mejorar la precisión y generalización en conjuntos de datos no balanceados. La técnica de Bagging, mediante el muestreo con reemplazo, también contribuye a reducir la varianza y mejorar la estabilidad de los modelos. En general, la diversidad de estrategias y enfoques en el ámbito de la clasificación en ML enriquece la capacidad para abordar con éxito la complejidad de los desafíos en el análisis de datos y la toma de decisiones.

De esta manera, el objetivo principal de este proyecto es desarrollar un modelo de clasificación efectivo en el ámbito del Machine Learning para asignar categorías fiscales a dispositivos biomédicos. Este objetivo surge de la necesidad de proporcionar una clasificación fiscal a cada dispositivo biomédico en el conjunto de datos del proyecto. Se propone evaluar modelos de clasificación que utilicen técnicas como PCA para reducir la dimensionalidad y abordar la correlación entre atributos. Además, se abordarán desafíos específicos, como conjuntos de datos no balanceados, mediante enfoques como SMOTE. Se considerarán métodos de ensamble como Random Forest y Boosting para mejorar la precisión y generalización en conjuntos de datos desbalanceados. En resumen, el objetivo es utilizar estrategias y enfoques avanzados en el ámbito de la clasificación en Machine Learning para lograr una asignación precisa de categorías fiscales a dispositivos biomédicos, contribuyendo así a la eficacia en el análisis de datos y la toma de decisiones en este contexto, con el fin de abolir el error humano y por ende los riesgos fiscales, legales y financieros.

[1] https://apps.irs.gov/app/understandingTaxes/student/glossary.jsp
[2] https://www.ey.com/en_gl/tax/indirect-tax-compliance

2. Marco teórico

Los problemas de clasificación en el ámbito de Machine Learning (ML) son una tarea fundamental y tiene una importancia significativa en diversos campos. Como lo son, automatización de decisiones, identificación de patrones, procesamiento de grandes volúmenes de datos, medicina y diagnóstico, entre otros. En general las estrategias de ML puede dividir los problemas de clasificación en tareas binarias, de múltiples clases, de múltiples etiquetas y jerárquicas (Sokolova & Lapalme, 2009). El objetivo de la clasificación es predecir una respuesta cualitativa asignando una observación a una categoría o clase. Hay muchas técnicas de clasificación o clasificadores, que implementan diferentes estrategias para resolver los diferentes problemas de clasificación que se pueden presentar en el mundo de la ciencia de datos y analítica. (James, 2023).

En algunos problemas de clasificación se requiere un preprocesamiento de los datos como el uso de Análisis de Componentes Principales (PCA). PCA en sí mismo no es un método de clasificación, su principal objetivo es transformar el conjunto de datos en un nuevo conjunto de variables no correlacionadas llamadas componentes principales, que capturan la mayor cantidad posible de la varianza en los datos (Hastie et al., 2009). Por lo tanto, PCA puede utilizarse indirectamente en tareas de clasificación como una etapa de preprocesamiento, como por ejemplo: la reducción de dimensionalidad manteniendo la mayor parte de la varianza, lo que puede ser beneficioso para la clasificación, ya que reduce la complejidad del modelo y puede ayudar a prevenir el sobreajuste de este; eliminación de correlación entre atributos, es decir, PCA transforma las características originales en componentes no correlacionados, esto puede ser útil cuando las características originales están altamente correlacionadas, ya que la multicolinealidad puede ser problemática para algunos algoritmos de clasificación; selección de características a partir de una evaluación de los pesos de las características en los componentes principales, se puede identificar qué características contribuyen más a la variabilidad en los datos; entre otras (Greenacre et al., 2023).

Por otra parte, dentro de los diferentes problemas de clasificación es frecuente enfrentarse a la clasificación de datos no balanceados. Es decir, cuando la cantidad de observaciones se distribuyen de manera desigual entre las clases, cuya diferencia es bastante significativa (Brownlee, 2020). Lo cual, es un problema para la mayoría de los algoritmos ML utilizados en la clasificación, ya que estos asumen que los datos están distribuidos de manera equitativa entre las clases. Lo que conlleva a sesgos en los procesos de clasificación y un rendimiento deficiente al favorecer las muestras de la clase mayoritaria (Brownlee, 2020; Tanha et al., 2020). Existen dos aproximaciones que buscan dar solución al problema de tratar con datos no balanceados. La primera aproximación es a nivel de datos, estas buscan balancearlos mediante métodos de re-muestreo, aumentando la clase que está en menor proporción o disminuyendo la clase que está en mayor proporción. Dentro de esta aproximación están "over-sampling" (Chawla et al., 2002) y "under-sampling" (Tahir et al., 2009), la técnica de sobremuestreo de minoría sintética (SMOTE) (N V Chawla et al., 2002), entre otros (Ganganwar, 2012).

El algoritmo de preprocesamiento "Synthetic Minority Oversampling Technique" (SMOTE) tiene aborda el problema de la falta de representación de la clase minoritaria en un conjunto de datos. Este algoritmo hace parte de la aproximación a nivel de datos, para el manejo de datos no balanceados por métodos de re-muestreo (Fernández, García, Herrera, et al., 2018). La base de SMOTE es preprocesar los datos a partir de la creación de nuevas instancias minoritarias para la interpolación entre vecinas de clases minoritarias, de allí viene el nombre de "sobre muestreo de minorías sintéticas". Este procedimiento genera un aumento en el número de instancias de clases minoritarias en el vecindario (Nitesh V. Chawla et al., 2002). SMOTE ayuda a evitar el sobreajuste (overfitting) al generar ejemplos sintéticos, en lugar de simplemente duplicar instancias de la clase minoritaria existente. Lo

cual, puede mejorar el rendimiento de los modelos de aprendizaje automático en conjuntos de datos desbalanceados (Nitesh V. Chawla et al., 2002; Fernández, García, Herrera, et al., 2018).

Por otra parte, la segunda aproximación es a nivel de algoritmo, la cual busca adaptar el código de ML a los datos no balanceados. Lo cual, puede implicar la penalización de clasificaciones erróneas por cada clase, minimizando el error de costo en vez de maximizar la tasa de precisión (Kotsiantis et al., 2006). En general, se han propuestos diferentes enfoques para esta aproximación que en últimas busca darle mayor peso a las muestras de clases minoritarias como los son los métodos de ensamble. Los métodos de ensamble en general se basan en la idea de combinar las predicciones de varios modelos base para obtener una predicción más robusta y precisa (Hastie et al., 2009; James, 2023)

El método de ensamble conocido como Random Forest (Bosque Aleatorio) es una técnica que combina múltiples modelos de árboles de decisión para mejorar la precisión y el rendimiento predictivo (Kullarni & Sinha, 2013). Random Forest, utiliza árboles de decisión como base, y múltiples árboles se combinan para formar un "bosque". Cada árbol se entrena en una muestra aleatoria del conjunto de datos, y, además, en cada nodo de decisión, se elige la característica o atributos más relevantes entre un subconjunto aleatorio de características. Esto ayuda a reducir la correlación entre los árboles individuales y mejora la capacidad del modelo para generalizar a datos nuevos. Luego, las predicciones de todos los árboles se combinan, ya sea por votación o promedio, para obtener una predicción final (Ziegler & König, 2014). De esta manera, al construir múltiples árboles de correlaciones y combinar sus predicciones, se reduce la varianza y se mejora la capacidad de generalización del modelo. Esto lo hace más robusto y menos propenso al sobreajuste en comparación con un solo árbol de decisión. Este enfoque ayuda a mejorar la precisión y la generalización del modelo, haciendo que Random Forest sea una técnica poderosa y popular en problemas de clasificación supervisada (James, 2023).

Boosting es un método de ensamble ampliamente utilizado que se fundamenta en la generación de versiones ponderadas de los datos de entrenamiento para construir un agregado secuencial del clasificador base (Freund & Schapire, 1996). En este proceso, cada árbol se genera de manera secuencial, tomando como punto de partida árboles previos. Cada uno de estos árboles se ajusta a una versión modificada del conjunto de datos original. A diferencia de la aproximación de ajustar un único árbol de decisión grande a los datos, el enfoque distintivo de Boosting radica en su ajuste estricto a los datos. En lugar de enfocarse directamente en el resultado Y, el algoritmo de Boosting ajusta cada árbol de decisión a los residuos del modelo, lo que representa las discrepancias entre las predicciones actuales y los valores reales. Posteriormente, cada nuevo árbol de decisión generado se agrega a la función ajustada para actualizar los residuos y mejorar la precisión global del modelo (Ferreira & Figueiredo, 2012). Es relevante destacar que cada árbol individual resultante de este proceso puede ser bastante pequeño, con solo unos pocos nodos terminales. La determinación del tamaño de estos árboles se rige por los parámetros del algoritmo de Boosting, como la profundidad máxima y la cantidad de nodos terminales permitidos. Al colocar árboles pequeños en los residuos, el algoritmo busca mejorar gradualmente en áreas donde el modelo no funciona bien, permitiendo una adaptación precisa a las complejidades del conjunto de datos. Este enfoque iterativo y secuencial distingue a Boosting como un método efectivo para mejorar la precisión de los modelos predictivos, especialmente en situaciones en las que un solo árbol de decisión podría ser insuficiente (James, 2023).

El método de clasificación de ensamble Bagging (Bootstrap Aggregating) es una técnica que se utiliza para mejorar la precisión y estabilidad de los modelos predictivos, especialmente en problemas de clasificación. Esta metodología busca reducir la varianza y mejorar la generalización del modelo al combinar múltiples clasificadores (Brownlee, 2020). El proceso de Bagging comienza con un muestro con reemplazo (Bootstrap), el cual, generan múltiples conjuntos de datos de entrenamiento mediante

el muestreo con reemplazo del conjunto de datos original. Cada conjunto de datos bootstrap es esencialmente una versión única y aleatoria del conjunto de datos original, con algunas instancias repetidas y otras ausentes. Posteriormente, con cada conjunto de datos bootstrap, se entrena un clasificador base que realiza predicciones sobre el conjunto de datos de prueba o sobre nuevos datos. Las predicciones de se combinan para obtener una predicción final. Esta combinación suele realizarse mediante votación (en problemas de clasificación binaria, por ejemplo, se elige la clase más votada) o mediante el promedio de las probabilidades de clasificación (Fernández, García, Galar, et al., 2018). La idea clave detrás del Bagging es que al introducir variabilidad a través del muestreo con reemplazo y al entrenar modelos independientes, se reduce la probabilidad de sobreajuste a datos específicos y se mejora la capacidad del modelo para generalizar a nuevos datos. Además, al promediar o votar sobre las predicciones de varios modelos, se obtiene un modelo conjunto más robusto y resistente a ruido o variaciones en los datos. El Bagging es utilizado comúnmente con algoritmos de alto sesgo o alta varianza, como árboles de decisión profundos, para mejorar su rendimiento y hacerlos más adecuados para aplicaciones del mundo real. Ejemplos populares de algoritmos de Bagging incluyen Random Forest, que es una extensión específica de Bagging para árboles de decisión (James, 2023).

En resumen, los problemas de clasificación en Machine Learning desempeñan un papel crucial en diversas áreas, desde la automatización de decisiones hasta la medicina. La diversidad de estrategias y clasificadores disponibles refleja la complejidad de estos desafíos. El preprocesamiento de datos, como el uso de PCA, agrega una capa adicional de eficacia al abordar la multicolinealidad y reducir la complejidad del modelo. La problemática de datos no balanceados destaca la necesidad de estrategias como SMOTE, que generan instancias sintéticas para contrarrestar la falta de representación de la clase minoritaria. Además, las aproximaciones a nivel de algoritmo, como Boosting y Random Forest, ofrecen soluciones efectivas al adaptar el modelo a conjuntos de datos no balanceados. La aplicación de técnicas como Bagging contribuye a mejorar la estabilidad y precisión de los modelos, abordando la sensibilidad a la variabilidad en los datos de entrenamiento. En conjunto, estas estrategias y enfoques enriquecen el campo de la clasificación en Machine Learning, permitiendo enfrentar con éxito la diversidad y complejidad de los problemas en el análisis de datos y la toma de decisiones.

3. Desarrollo metodológico

3.1. Entendimiento del problema, pregunta de negocio o hipótesis

El área de impuestos indirectos de una empresa anónima dedicada a la manufacturación y comercialización de dispositivos médicos necesita crear un modelo de clasificación que prediga las categorías fiscales pertenecientes a cada uno de sus sesenta y tres mil productos como instrumentos quirúrgicos, prótesis, implantes, tejidos sintéticos, entre otros. En la actualidad este proceso se ha venido realizando de manera manual mediante el filtrado de datos sobre las variables, haciendo uso de la clasificación por familias y tipo de producto definidos dentro de la compañía. Este procedimiento inicia con la descarga de una tabla categorizada separada por comas (.CSV) que se encuentra en la interfaz de SAP online. Esta metodología de asignación de categorías fiscales está sujeta al error humano asociado al proceso mecánico y repetitivo que conllevan al agotamiento mental por el manejo de este volumen de datos. Además, de requerir tiempo y un gran esfuerzo por parte del personal encargado, estos errores conllevan a una clasificación inadecuada que podrían conducir a riesgos fiscales, legales y financieros.

Por lo tanto, el objetivo del presente proyecto integrador es la implementación de un algoritmo que optimice y estandarice la categorización o asignación de clases obteniendo una perspectiva global basada en datos con el fin de abolir el error humano y por ende los riesgos fiscales, legales y financieros.

3.2. Análisis exploratorio de datos y transformación de los datos

Los datos corresponden a una lista de aproximadamente sesenta y tres mil dispositivos biomédicos de una empresa anónima dedicada a su manufacturación y comercialización. A cada uno de estos productos se le asigna un conjunto de características que describen la línea, familia del producto, lugares de manufactura, almacenamiento y codificación interna (Tabla 1).

Dentro de este conjunto de características se encuentra la variable objetivo VERTEX PROD ID, la cual, hace referencia a las categorías fiscales que se deben aplicar a cada producto para que puedan ser grabados fiscalmente de forma apropiada. A la fecha existen 144 categorías VERTEX PROD ID, de las cuales, las categorías 48Y, 38N, 37N, 65Y, 14N, 47Y, 41Y y 06N son asignadas cada una a más de 2000 productos, el resto de los productos son asignados a otras categorías en una menor proporción, siendo algunas de estas asignadas a un producto solamente (Figura 1). Este panorama muestra unos datos claramente desbalanceados. Por ejemplo, la naturaleza de un implante requiere el uso de varios instrumentos quirúrgicos para el procedimiento médico, por lo cual el desbalance de datos también es inherente al tipo de negocio.

Nº	Variable	Descripción
1	ITEM NO	Colección de IDs únicos del producto
2	DOM RLS CODE	Códigos únicos de despliegue del producto en las plataformas de ventas, no todos los productos requieren de este código.
3	MANUFACT WHSE	Locación en donde fue fabricado
4	PRODUCT CLASS	Catalogación de uso del producto
5	PROD CLASS DESC	Catalogación de uso del producto en formato texto
6	ITEM HSE PROD LINE	Línea de producción de un artículo agrupado por categorías de producto. Cada línea de producción se encarga de un grupo de productos con características similares
7	ITEM HSE PROD LINE DESC	Esta es la descripción abreviada de ITEM HSE PROD LINE. Es una cadena de texto de cada línea de producción.
8	BUSINESS SEGMENT CDE	Segmento de negocio al que pertenece el producto. Consiste en un código numérico asignado a cada uno.
9.	BUSINESS SEGMENT DESC	Descripción en texto del segmento de negocio BUSINESS SEGMENT CDE
10	APPLICATION CDE	El código de aplicación del producto explica la usabilidad del artículo con fines médicos.
11	APPLICATION DESC	La descripción del código de aplicación se refiere a la explicación en formato textual de aplicación del artículo.
12	PRODUCT TYPE	Tipo de producto haciendo referencia a su clasificación. Está en formato de texto abreviado.
13	PRODUCT TYPE DESC	Describe el tipo de producto o PRODUCT TYPE de manera más precisa.
14	PRODUCT GROUP	Hace referencia al grupo al que pertenece el artículo y está descrito como una categoría numérica.
15	PRODUCT GROUP DESC	Descripción en formato textual de PRODUCT GROUP.
16	PROD SYSTEM	Describe de forma abreviada el sistema al que está asignado el producto.
17	PROD SYSTEM DESC	Descripción detallada en formato texto del sistema de distribución del producto.
18	PROD SUBSYSTEM	Al igual que en el sistema en el que se asigna el producto, hay un subsistema para darle mayor división a los artículos y este también es una versión de texto reducida.
19	PROD SUBSYSTEM DESC	Descripción más específica en formato texto de PROD SUBSYSTEM.
20	PROD FAMILY	La familia del producto es una forma de clasificarlos por su tipo de uso, ya que están todos asociados a grupos de uso como instrumentos quirúrgicos, software, piezas de mantenimiento, etc.
21	PROD FAMILY DESC	Descripción en formato texto de PROD FAMILY.
22	VERTEX PROD ID	Variable objetivo, se refiere a las categorías fiscales que se deben aplicar a cada producto para que puedan ser grabados fiscalmente de forma apropiada.
23	MANUFACT TYPE CODE	Tipo de manufactura del producto en formato texto abreviado.
24	INSTRUMENT DESC	Esta variable menciona si un artículo es o no un instrumento. En formato texto.

Tabla 1. Conjunto de características que describen la línea, familia del producto, lugares de manufactura, almacenamiento y codificación interna.

Figura 1. Distribución de valores únicos asignados a 144 categorías VERTEX PROD ID existentes.

Antes de abordar el problema de desbalanceo de los datos, se realizó la preparación de estos. En primer lugar, se realizó la eliminación de los espacios en blanco. En segundo, lugar se realizó asignaciones de representaciones numéricas a cada categoría por cada columna, lo cual, fue validado con la cantidad de categorías que existen por columnas. Dichas asignaciones se realizaron con la librería *sklearn.preprocessing.LabelEncoder.*

Una vez realizada la limpieza y la asignación numérica de los datos, se calcularon los coeficientes de correlación entre todas las posibles combinaciones de variables del conjunto de datos. La matriz de correlación construida permitió evaluar posibles relaciones lineales positivas o negativas entre las variables (ver Figura 2 y Figura 3). Dentro de las correlaciones positivas encontradas se puede observar una correlación fuerte entre el Vertex Prod ID y el Product Class de los artículos, quiere esto decir que la clase del producto tiene una fuerte influencia sobre la categoría fiscal en la que se asigna el producto, sin embargo, no puede determinarse que con solo esta variable es suficiente para una asignación apropiada de la categoría fiscal (Figura 2).

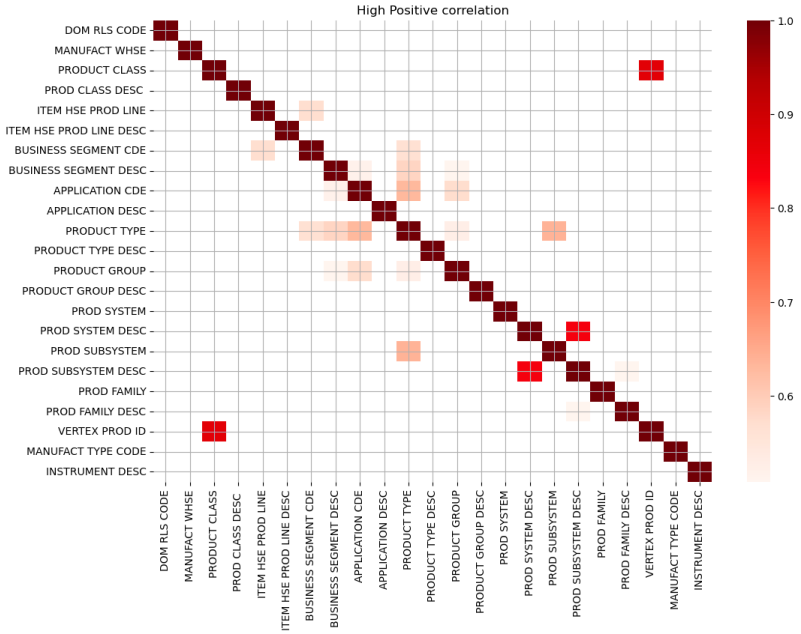


Figura 2. Mapa de calor de las correlaciones positivas mayores a 0.5.

Por otro lado, dentro de las correlaciones negativas identificadas (Figura 3) podemos observar que no existen correlaciones fuertes relevantes para analizar.

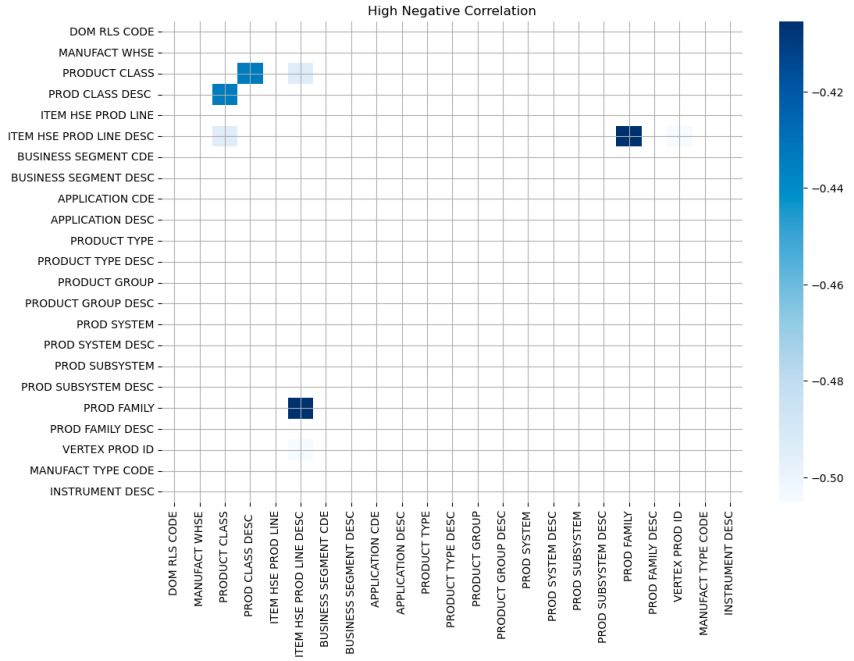


Figura 3. Mapa de calor de las correlaciones negativas menores a -0.4.

En el análisis exploratorio de los datos se implementó una estrategia de clasificación de Random Forest y como resultado de esta primera estrategia de clasificación se obtuvieron los siguientes resultados. Una precisión de clasificación de 0.79 (figura 4).

Reporte de clasificación:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	5
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	13
4	0.93	0.94	0.93	252
5	1.00	0.50	0.67	4
6	0.70	0.21	0.32	34
7	1.00	1.00	1.00	7
8	0.00	0.00	0.00	0
9	1.00	0.99	0.99	423
10	0.97	0.99	0.98	176
11	0.82	0.98	0.89	127
12	0.95	0.95	0.95	20
14	1.00	1.00	1.00	1
15	0.00	0.00	0.00	0
16	1.00	0.75	0.86	4
17	1.00	1.00	1.00	1
18	0.98	0.98	0.98	131
19	0.99	0.83	0.90	302
20	1.00	0.94	0.97	33
...				
accuracy			0.98	12485
macro avg	0.76	0.76	0.74	12485
weighted avg	0.98	0.98	0.98	12485

Figura 4. Reporte de clasificación de análisis de Random Forest de base en el análisis exploratorio.

3.3. Selección de modelos (supervisados y no supervisados), Ingeniería de Características, Entrenamiento, Evaluación

Los modelos evaluados para resolver el problema de clasificación del presente trabajo fueron del tipo supervisados. Los hiper-parámetros utilizados en los modelos fueron por defecto. Con respecto a la ingeniería de características se realizaron los procedimientos enunciados en la tabla 2.

Selección de características	Se eliminó Item Desc por tener información redundante
Transformación de características	Se realizó asignación de representaciones numérica sobre las variables categóricas usando ordinal_encoding. Redistribución de la variable objetivo.
Creación de nuevas características	Se creo una nueva categoría o clase fiscal MA DIY
Manejo de datos faltantes	Se remplazaron los datos faltantes con el valor MISS para las columnas DOM RLS COD, MANUFACT TYPE CODE y INSTRUMENT DESC.
Ingeniería temporal	No aplica
Reducción de dimensionalidad	No aplica
Ingeniería de texto	No aplica
Validación de características	No aplica
Adaptación de modelos específicos	Ninguno de los modelos utilizados requiere de datos estandarizados para mejorar su funcionamiento puesto que su estructura está basada en árboles de decisiones. La única adaptación que se realizó fue con *ordinal_encoder.*

Tabla 2. Procedimientos realizados en la ingeniería de características.

Las métricas evaluadas por cada modelo fueron: Precisión, Recall y F1-Score. La precisión describe la proporción de predicciones positivas que fueron correctamente identificadas, es decir, tiene una baja tasa de falsos positivos.

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ positivos + Falsos\ Positivos}$$

Recall, también conocido como sensibilidad, es la proporción de predicciones positivas que fueron correctamente identificadas, pero teniendo en cuenta a los falsos negativos a diferencia de la precisión. Por lo tanto, un alto recall indica que tiene una baja tasa de falsos negativos.

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

F1-Score es la media armónica de precisión y recall, toma valores entre 0 y 1, donde 1 indica que se tiene una precisión y un recall perfecto.

$$F1 - Score = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

Estás métricas permitieron evaluar y comparar los modelos de clasificación utilizados para predecir correctamente las clases objetivo.

3.3.1. Redistribución de los datos y transformación

Como se pudo observar en la figura 1 la distribución de los datos entre las clases no es uniforme. Lo que implica un sesgo significativo en la distribución de clases, es decir, hay un desbalanceo significativo. La clasificación bajo estas condiciones requiere el empleo de técnicas especializadas, incluyendo la preparación de datos, algoritmos de aprendizaje y métricas de rendimiento específicas. En esta primera aproximación ser realizó una redistribución de los datos, siguiendo el siguiente criterio. Todas las categorías que tuvieron menos de 50 productos se unificaron en una clase llamada MA DIY, donde se reunieron 895 productos (Figura 5).

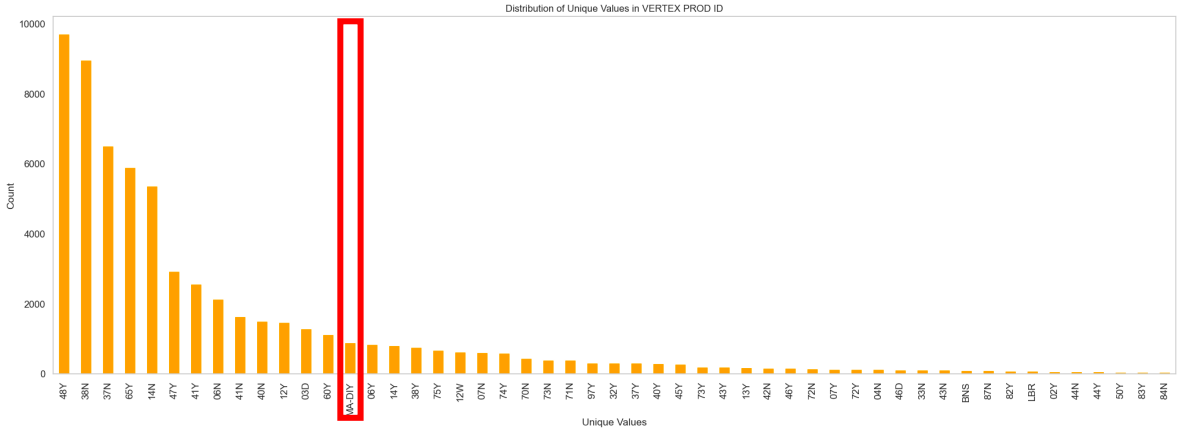


Figura 5. Distribución de valores únicos en VERTEX PROD ID, donde se resalta la nueva categoría MA DIY que reúne 895 productos que tuvieron un conteo menor a 50 productos.

Posterior a la redistribución de los datos se realizó la asignación de representaciones numérica sobre las variables categóricas usando *ordinal_encoder*, el cual, tiene manejador de errores, busca valores desconocidos para el *encode* asignando un valor predeterminado. A continuación, se realizó la implementación, entrenamiento y evaluación de los seleccionados.

3.3.2. Análisis de componentes principales (PCA)

El Análisis de Componentes Principales (PCA) reduce la dimensionalidad de los datos al encontrar nuevas direcciones en las cuales los datos conservan la mayor variabilidad. Estas nuevas direcciones son los componentes principales, y el conjunto transformado de datos permite una representación más eficiente y compacta del conjunto original. La elección del número de componentes principales a retener dependerá del objetivo del análisis y la cantidad de varianza que se desee conservar (Greenacre et al., 2023). El funcionamiento general de PCA es:

1. Cálculo de la matriz de covarianza: dado un conjunto de datos X con n observaciones y p características, se calcula la matriz de covarianza C que muestra las relaciones entre todas las posibles parejas de características.

$$C = \frac{1}{n-1}(X - \bar{X})^T(X - \bar{X})$$

2. Descomposición de la matriz de covarianza: se realiza la descomposición de valores propios sobre la matriz de covarianza *C*. Esto proporciona los vectores y los valores propios.
3. Selección de componentes principales: los vectores propios se ordenan según el orden descendente. Estos representan las direcciones en las que los datos tienen la mayor variabilidad.
4. Los componentes principales se eligen tomando los primeros *k* valores propios, donde *k* es la cantidad de componentes deseados. Generalmente, se seleccionan los primeros *k* vectores propios que explican la mayor parte de la varianza en los datos.
5. Transformación de los datos: los datos originales *X* se proyectan en el nuevo espacio definido por los *k* componentes principales. La transformación se realiza multiplicando la matriz de datos original por la matriz de componentes principales.

   $$Y = X \times V$$

   Donde *Y* es la matriz transformada de datos y *V* es la matriz formada por los *k* vectores propios seleccionados.
6. Varianza explicada: cada componente explica una cierta cantidad de varianza en los datos. La suma de las varianzas explicadas por todos los componentes principales da la varianza total de los datos.
7. Elección del número de componentes principales: se puede decidir cuántos componentes principales retener con base en la cantidad de varianza que se desea conservar. Una regla común es seleccionar suficientes componentes para explicar, por ejemplo, el 95% o el 99% de la varianza total.

Se realizó un Análisis de Componentes Principales (PCA) para entender cómo cada componente contribuye a la explicación total de la variabilidad en los datos, mediante la reducción de la dimensionalidad de los datos mientras se retiene la mayor cantidad posible de información. En la figura 6 se muestra la gráfica de varianza acumulada por componente, la cual, muestra cómo la varianza total se acumula a medida que se añaden más componentes principales. Teniendo en cuenta que cada componente principal captura cierta cantidad de varianza de los datos, en la gráfica podemos observar que a partir de los diecisiete componentes se explica aproximadamente el 95% de la varianza total en los datos. Al considerar estos componentes, retenemos una cantidad significativa de información mientras reducimos la dimensionalidad. Además, la curva de varianza acumulada se tiende a aplanarse después de estos componentes, indicando que agregar más componentes no contribuiría de manera significativa a explicar la variabilidad en los datos.

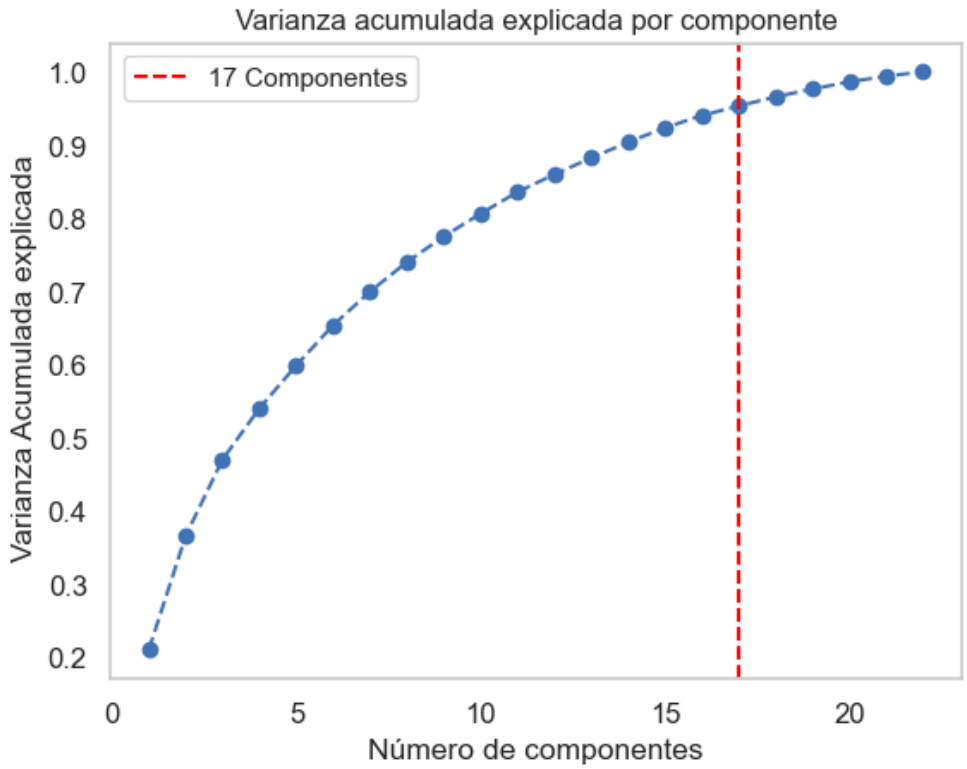


Figura 6. Varianza acumulada explicada por componente. Los valores en el eje x representan los componentes y en el eje y, la varianza acumulada.

Reporte de clasificación:

	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.92	0.89	0.91	261
2	0.67	0.23	0.34	26
3	1.00	0.98	0.99	428
4	0.94	0.98	0.96	170
5	0.85	0.97	0.90	123
6	0.93	1.00	0.97	28
7	0.98	0.97	0.98	128
8	0.98	0.85	0.91	296
9	0.97	0.95	0.96	38
10	0.99	1.00	0.99	1075
11	0.97	0.92	0.95	163
12	0.91	0.95	0.93	63
13	1.00	0.91	0.95	23
14	0.99	0.98	0.98	1303
15	1.00	0.94	0.97	63
16	0.98	1.00	0.99	1793
17	0.92	0.95	0.94	152
18	1.00	1.00	1.00	301
19	0.98	0.98	0.98	59
...				
accuracy			0.98	12485
macro avg	0.95	0.94	0.94	12485
weighted avg	0.98	0.98	0.98	12485

Figura 7. Reporte de clasificación del modelo de Random Forest después de la reducción de dimensionalidad con PCA.

Una vez realizada el preprocesamiento de datos, reduciendo la dimensionalidad con PCA, se implementó la técnica de clasificación con Random Forest, con lo que se obtuvo un balanced accuracy de 0.94 (Figura 7). Sin embargo, para obtener esta puntuación es necesario mantener 17 componentes, lo cual no hace una gran diferencia a nivel computacional en el tiempo de predicción o en el tiempo de ejecución del modelo. De igual manera un PCA con este conjunto de datos con solo 24 variables no es necesario. Y emplearlo en este caso de negocio implicaría hacer una transformación adicional sobre los datos cada vez que se quieran hacer predicciones. Dado todo lo anterior, concluimos que para el caso de negocio y en vista de la reducida cantidad de variables y el bajo beneficio que se obtendría de un proceso de reducción de dimensionalidad como este, aplicar PCA es innecesario y decidimos prescindir del uso de este.

3.3.3. Balanceo de datos con SMOTE

Descripción básica de cómo funciona SMOTE (Fernández, García, Herrera, et al., 2018; James, 2023):

1. Identificación de la clase minoritaria: Primero, se identifica la clase minoritaria en el conjunto de datos. En un problema de desequilibrio de clases, la clase minoritaria es la que tiene menos instancias.
2. Selección de un ejemplo de la clase minoritaria: Se elige aleatoriamente un ejemplo de la clase minoritaria.
3. Selección de k vecinos más cercanos: Se seleccionan k ejemplos cercanos (vecinos) de la clase minoritaria. La elección de k depende del diseño de la técnica y es un parámetro ajustable.
4. Generación de ejemplos sintéticos: Se generan nuevos ejemplos sintéticos entre el ejemplo seleccionado y sus k vecinos más cercanos. Estos nuevos ejemplos se crean tomando diferencias ponderadas entre las características del ejemplo seleccionado y sus vecinos. Donde la proporción es un número entre 0 y 1 que determina cuánto se deben ponderar las diferencias.
5. Incorporación de ejemplos sintéticos al conjunto de datos: Los nuevos ejemplos sintéticos se agregan al conjunto de datos original, y ahora el conjunto de datos está más equilibrado en términos de clases.

Los resultados generados con Random Forest a partir de los datos balanceados con SMOTE, utilizando hiper-parámetros por defecto fue una precisión de 0.954. El *f1-score* nos indica que el balance entre *precision* y *recall* en la mayoría de las clases predichas versus los valores reales de *y* es bueno, por lo que se aproximan a 1 (Figura 8).

Reporte de clasificación:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.93	0.93	0.93	252
2	0.38	0.62	0.47	34
3	1.00	0.99	0.99	423
4	0.96	0.99	0.97	176
5	0.88	0.72	0.79	127
6	0.95	0.95	0.95	20
7	0.98	0.98	0.98	131
8	1.00	0.83	0.91	302
9	0.74	0.97	0.84	33
10	1.00	1.00	1.00	1049
11	0.99	1.00	1.00	150
12	0.91	0.95	0.93	61
13	1.00	1.00	1.00	21
14	1.00	0.98	0.99	1309
15	0.95	0.97	0.96	64
16	0.99	1.00	0.99	1752
17	0.96	0.96	0.96	155
18	1.00	1.00	1.00	283
19	0.98	0.92	0.95	63
...				
accuracy			0.98	12485
macro avg	0.94	0.95	0.94	12485
weighted avg	0.98	0.98	0.98	12485

Figura 8. Reporte de clasificación del análisis con Random Forest a partir de los datos balanceados con SMOTE.

Luego de aplicar el modelo, de los 62424 datos iniciales, resultaron 393159 (Figura 9).

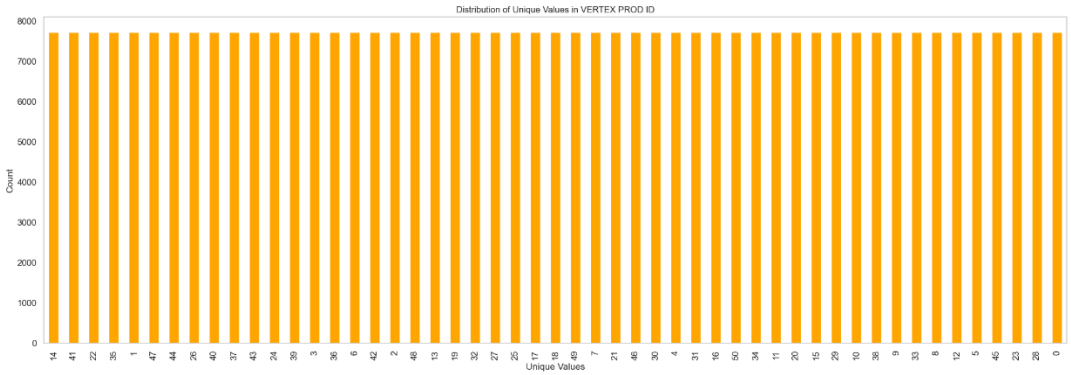


Figura 9. Distribución de datos con SMOTE

3.3.4. Random Forest

Random Forest es un algoritmo de aprendizaje supervisado que pertenece a la categoría de métodos de ensamblaje (Ziegler & König, 2014). Funciona de la siguiente manera:

1. Construcción de Múltiples Árboles de Decisión: Random Forest construye un conjunto de árboles de decisión durante el entrenamiento. Cada árbol se entrena con un subconjunto aleatorio del conjunto de datos de entrenamiento.
2. Muestreo con Reemplazo y Características Aleatorias: Para cada árbol, se realiza muestreo con reemplazo (bootstrap) en el conjunto de datos. Esto implica seleccionar aleatoriamente muestras del conjunto de datos, permitiendo que algunas muestras se seleccionen más de una

vez y otras no se seleccionen en absoluto. Además, en cada división de un árbol, se considera solo un subconjunto aleatorio de características. Esto introduce más aleatoriedad y desliga la correlación los árboles, ayudando a mejorar la generalización del modelo.

3. Entrenamiento de Árboles Independientes: Cada árbol se entrena de manera independiente utilizando su conjunto de datos de entrenamiento único. Durante el entrenamiento de cada árbol, se realizan divisiones en los nodos de acuerdo con reglas que maximizan la ganancia de información o reducen el error.
4. Combina Predicciones: Una vez que todos los árboles están entrenados, las predicciones de cada árbol se combinan para obtener una predicción final. En problemas de clasificación, se puede utilizar una votación mayoritaria entre los árboles para determinar la clase final. En problemas de regresión, se puede realizar un promedio de las predicciones.

Los resultados generados con Random Forest a partir de hiper-parámetros por defecto fue una precisión de 0.947. El *f1-score* nos indica que el balance entre *precision* y *recall* en la mayoría de las clases predichas versus los valores reales de *y* es bueno, por lo que se aproximan a 1 (Figura 10).

Reporte de clasificación:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.94	0.92	0.93	252
2	1.00	0.15	0.26	34
3	1.00	0.99	0.99	423
4	0.96	0.99	0.97	176
5	0.81	1.00	0.90	127
6	0.95	0.95	0.95	20
7	0.98	0.98	0.98	131
8	0.99	0.83	0.90	302
9	1.00	0.97	0.98	33
10	1.00	1.00	1.00	1049
11	0.99	1.00	1.00	150
12	0.89	0.95	0.92	61
13	1.00	1.00	1.00	21
14	1.00	0.98	0.99	1309
15	0.97	0.95	0.96	64
16	0.99	1.00	0.99	1752
17	0.96	0.96	0.96	155
18	1.00	1.00	1.00	283
19	0.98	0.92	0.95	63
...				
accuracy			0.98	12485
macro avg	0.95	0.95	0.94	12485
weighted avg	0.98	0.98	0.98	12485

Figura 10. Reporte de clasificación del análisis con Random Forest.

3.3.5. Boosting - XGBOOST

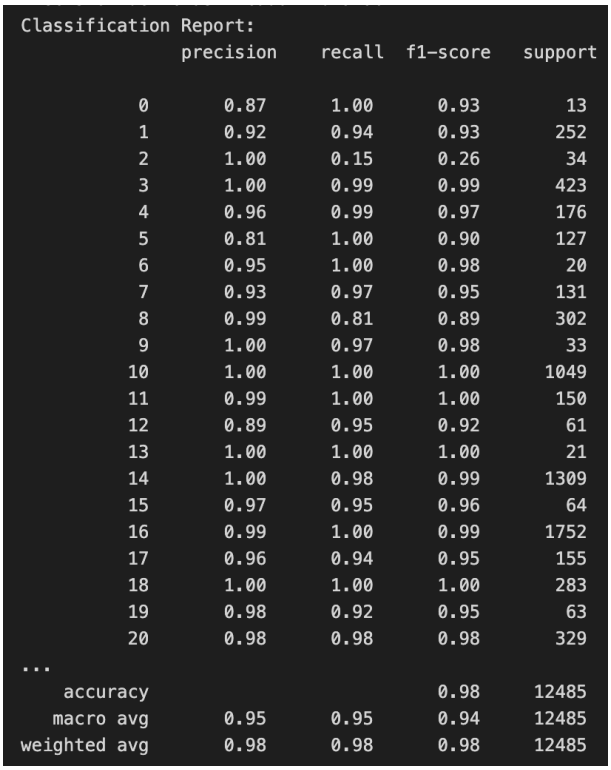
El Boosting tiene tres parámetros de ajuste:

1. El número de árboles *B*. A diferencia del Bagging y Random Forest, el modelo puede sobreajustarse si *B* es demasiado grande, aunque este sobreajuste tiende a ocurrir lentamente, si es que ocurre. Usamos validación cruzada para seleccionar *B*.

2. El parámetro de contracción λ, un pequeño número positivo. Esto controla la velocidad a la que el modelo de Boosting aprende. Los valores típicos son 0,01 o 0,001 y la elección correcta puede depender del problema. Un λ muy pequeño puede requerir el uso de un valor muy grande de *B* para lograr un buen rendimiento.

3. El número *d* de divisiones en cada árbol, controla la complejidad del conjunto potenciado. A menudo, *d* =1 funciona bien, en cuyo caso cada árbol consta de una única división. De manera más general, *d* es la profundidad de interacción y controla el orden de interacción del modelo potenciado, donde *d* divisiones pueden involucrar como máximo d variables.

Los resultados generados con XGBOOST a partir de hiper-parámetros por defecto fue una precisión de 0.9496. El *f1-score* nos indica que el balance entre *precision* y *recall* en la mayoría de las clases predichas versus los valores reales de *y* es bueno, dado que se aproximan a 1 (Figura 11).



Classification Report:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	0.92	0.94	0.93	252
2	1.00	0.15	0.26	34
3	1.00	0.99	0.99	423
4	0.96	0.99	0.97	176
5	0.81	1.00	0.90	127
6	0.95	1.00	0.98	20
7	0.93	0.97	0.95	131
8	0.99	0.81	0.89	302
9	1.00	0.97	0.98	33
10	1.00	1.00	1.00	1049
11	0.99	1.00	1.00	150
12	0.89	0.95	0.92	61
13	1.00	1.00	1.00	21
14	1.00	0.98	0.99	1309
15	0.97	0.95	0.96	64
16	0.99	1.00	0.99	1752
17	0.96	0.94	0.95	155
18	1.00	1.00	1.00	283
19	0.98	0.92	0.95	63
20	0.98	0.98	0.98	329
...				
accuracy			0.98	12485
macro avg	0.95	0.95	0.94	12485
weighted avg	0.98	0.98	0.98	12485

Figura 11. Reporte de clasificación de análisis con XGBOOST.

3.3.6. Bagging con árboles de decisión

Bagging (Bootstrap Aggregating) es una técnica de ensamble que se utiliza para mejorar la estabilidad y el rendimiento de los modelos de Machine Learning, y puede aplicarse a diversos algoritmos, incluidos los árboles de decisión. Aquí se da una breve descripción de cómo funciona Bagging con árboles de decisión (James, 2023):

1. Bootstrap Sampling: El proceso comienza mediante la creación de múltiples conjuntos de datos de entrenamiento a partir del conjunto de datos original utilizando el re-muestreo con bootstrap, se seleccionan aleatoriamente muestras con reemplazo del conjunto de datos original para formar conjuntos de datos más pequeños, pero de tamaño similar al conjunto de datos original. Algunas instancias pueden aparecer en el conjunto de datos de entrenamiento múltiples veces, mientras que otras pueden no aparecer en absoluto.
2. Entrenamiento de Árboles de Decisión: Para cada conjunto de datos de entrenamiento obtenido mediante el muestreo bootstrap, se entrena un árbol de decisión. Cada árbol se

entrena de manera independiente y puede capturar diferentes patrones en los datos debido a las variaciones introducidas por el proceso de bootstrap.

3. Votación o Promedio: Una vez que se han entrenado todos los árboles, se utiliza un proceso de votación (en el caso de problemas de clasificación) para combinar las predicciones de los árboles individuales y generar una predicción final. En el caso de clasificación, se puede utilizar una mayoría de votos para tomar la decisión final.
4. Reducción de la Variabilidad: La principal ventaja de Bagging es que reduce la varianza del modelo, lo que ayuda a prevenir el sobreajuste. Cada árbol se entrena en un conjunto de datos ligeramente diferente debido al muestreo bootstrap, lo que significa que cada árbol tiene una perspectiva única sobre los datos. Al combinar estas perspectivas, el modelo de ensamble tiende a generalizar mejor nuevos datos.

Los resultados generados con Bagging con árboles de decisión a partir de hiper-parámetros por defecto fue una precisión de 0.9494. El *f1-score* nos indica que el balance entre *precision* y *recall* en la mayoría de las clases predichas versus los valores reales de *y* es bueno, por lo que se aproximan a 1 (Figura 12).

Accuracy: 0.9494
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.92	0.92	0.92	252
2	0.70	0.21	0.32	34
3	1.00	0.99	0.99	423
4	0.96	0.99	0.98	176
5	0.82	0.98	0.89	127
6	0.95	1.00	0.98	20
7	0.97	0.98	0.98	131
8	0.99	0.82	0.90	302
9	1.00	0.94	0.97	33
10	1.00	1.00	1.00	1049
11	0.97	1.00	0.99	150
12	0.89	0.95	0.92	61
13	1.00	1.00	1.00	21
14	1.00	0.98	0.99	1309
15	0.97	0.97	0.97	64
16	0.99	1.00	0.99	1752
17	0.96	0.96	0.96	155
18	0.99	0.99	0.99	283
19	0.94	0.92	0.93	63
20	0.98	0.99	0.98	329
...				
accuracy			0.98	12485
macro avg	0.95	0.95	0.94	12485
weighted avg	0.98	0.98	0.98	12485

Figura 12. Reporte de clasificación de análisis con Bagging con árboles de decisión.

3.3.7. Evaluación cruzada

La validación cruzada, permite evaluar el rendimiento de un modelo predictivo, como los utilizados en métodos de clasificación. El objetivo principal es estimar el comportamiento de un modelo con datos no vistos, aquellos que no se utilizaron durante el entrenamiento del modelo. Lo cual, implica dividir el conjunto de datos en subconjuntos separados para entrenar y probar el modelo de manera iterativa. Esta etapa del análisis es esencial para evaluar cómo se generaliza un modelo a nuevos datos

y para identificar posibles problemas de sobreajuste o subajuste. También pueden ayudar a ajustar los hiper-parámetros del modelo para obtener un rendimiento óptimo (James, 2023).

En el proceso de evaluación de los modelos se realizó un proceso de optimización en el que se evaluaron diferentes hiper-parámetros con *GridSearchCV* (Tabla 3). De esta manera, se realizó la validación cruzada de Random Forest y Bagging de los cuales se definieron los mejores hiper-parámetros (Tabla 4). Adicionalmente se tiene en cuenta la precisión y la velocidad de procesamiento de cada uno de los modelos en la tabla 5. Posteriormente en la figuras 13-15 y las tablas 6-8 se hace la comparación de cada uno de los modelos utilizados en aras de seleccionar un modelo final que tenga el mejor desempeño.

Hiperparámetros evaluados en RandomForest		**Hiperparámetros evaluados en Bagging**	
n_estimators	10, 50, 100, 250, 300	*n_estimators*	10, 50, 100, 250, 300
max_depth	None, 5, 10, 15, 20, 30	*max_samples*	0.5, 0.7, 0.3, 1.0
min_samples_split	2, 5, 7 ,10, 13	*max_features*	0.5, 0.7, 0.3, 1.0
min_samples_leaf	1, 2, 4, 8	*bootstrap*	True, False
		bootstrap_features	True, False

Tabla 3. Hiper-parámetros evaluados para los modelos Random Forest y Bagging.

Hiper-parámetros definidos para Random Forest		**Hiper-parámetros definidos para Bagging**	
n_estimators	250	*n_estimators*	250
max_depth	20	*max_samples*	1.0
min_samples_split	2	*max_features*	1.0
min_samples_leaf	1	*bootstrap*	True
		bootstrap_features	True

Tabla 4. Mejores Hiper-parámetros establecidos para los modelos Random Forest y Bagging.

Métricas	**Random Forest**	**Bagging**	**Boosting**
Precisión del conjunto de prueba	0.9475	0.9487	0.9496
Velocidad de procesamiento	8 segundos	3.5 segundos	13.3 segundos

Tabla 5. Métricas evaluadas por modelos.

	RF	**RF Bal**	**XC Boost**	**Bagging**	**PCA_RF**	**SMOTE_RF**	**SMOTE_Bagging**
Conteo	42	42	42	42	42	42	42
Media	0.838	0.952	0.961	0.958	0.950	0.931	0.940
Des Std	0.314	0.092	0.074	0.081	0.090	0.136	0.103
Min	0.000	0.580	0.580	0.570	0.580	0.380	0.560
25%	0.900	0.952	0.952	0.952	0.940	0.942	0.940
50%	0.995	0.99	0.990	0.990	0.985	0.990	0.975
75%	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabla 6. Comparación de precisión por modelos.

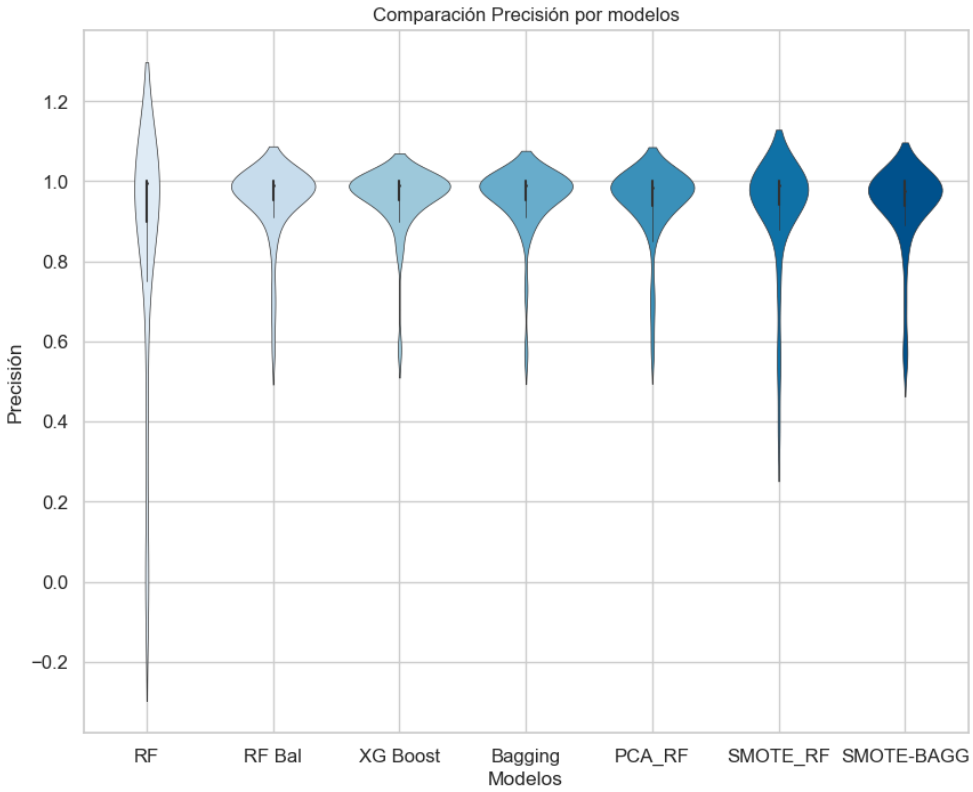


Figura 13. Comparación de precisión por modelos.

	RF	**RF Bal**	**XC Boost**	**Bagging**	**PCA_RF**	**SMOTE_RF**	**SMOTE_Bagging**
Conteo	47	47	47	47	47	47	47
Media	0.721	0.952	0.941	0.950	0.943	0.956	0.951
Des Std	0.397	0.115	0.149	0.117	0.115	0.073	0.120
Min	0.000	0.230	0.040	0.230	0.230	0.620	0.210
25%	0.500	0.940	0.945	0.940	0.930	0.945	0.945
50%	0.950	0.990	0.990	0.990	0.970	0.980	0.990
75%	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabla 7. Comparación de Recall-score por modelos.

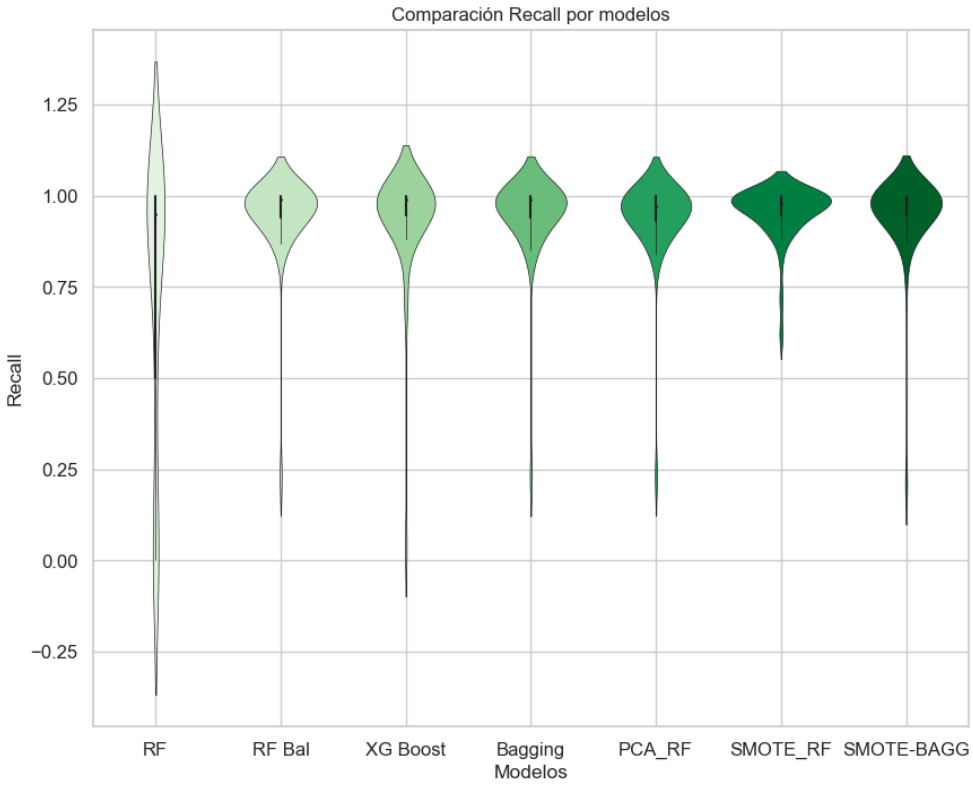


Figura 14. Comparación de Recall-score por modelos.

	RF	**RF Bal**	**XC Boost**	**Bagging**	**PCA_RF**	**SMOTE_RF**	**SMOTE_Bagging**
Conteo	38	38	38	38	38	38	38
Media	0.891	0.945	0.933	0.945	0.939	0.936	0.952
Des Std	0.177	0.117	0.155	0.111	0.115	0.111	0.128
Min	0.320	0.340	0.070	0.380	0.340	0.470	0.210
25%	0.892	0.942	0.940	0.950	0.950	0.935	0.953
50%	0.980	0.985	0.980	0.980	0.970	0.980	0.985
75%	1.000	0.997	0.990	0.990	0.990	0.997	1.000

Tabla 8. Comparación del estadístico *F1-Score* por modelos.

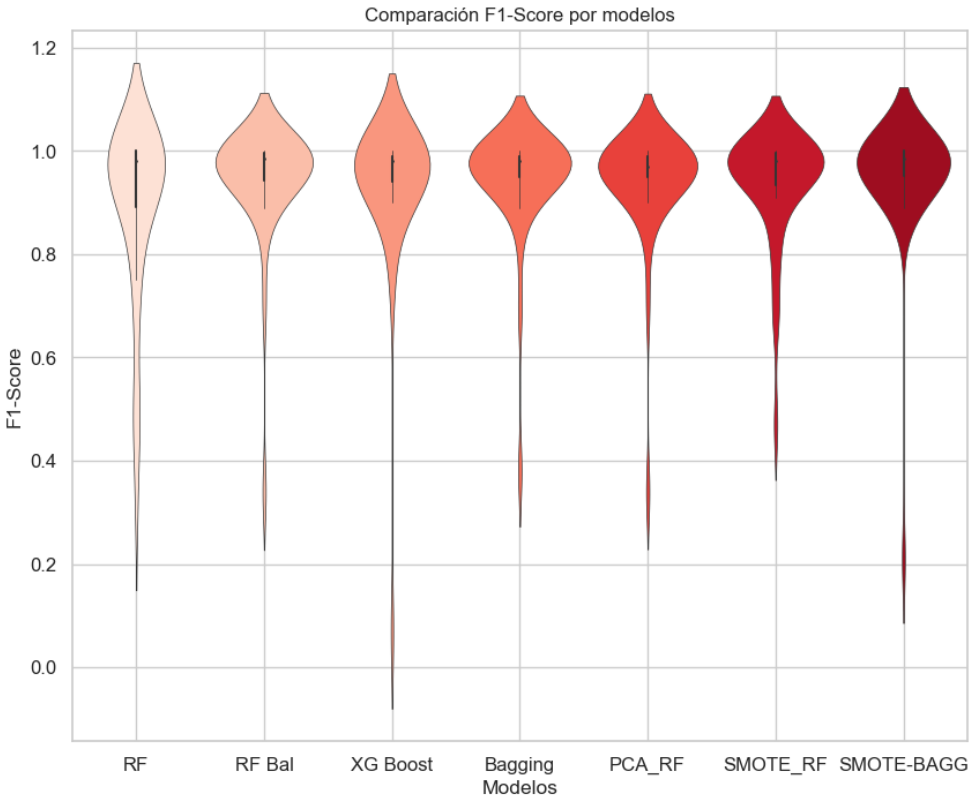


Figura 15. Comparación de F1-score por modelos.

Para el caso de Boosting no se realizó validación cruzada porque el F1-score para algunas clases presentaba un desempeño regular y no estaba prediciendo de forma aceptable algunas clases representativas.

3.4. Análisis y Conclusiones

- El análisis se centra en aproximadamente sesenta y tres mil dispositivos biomédicos de una empresa dedicada a su manufactura y venta, cada uno con asignaciones de características y una variable clave, "VERTEX PROD ID", relacionada con categorías fiscales. Se observa un desbalance en estas asignaciones, atribuido a la diversidad de productos.
- Se evidencia un desbalance significativo en la distribución de datos entre clases, lo que requiere el uso de técnicas especializadas para la clasificación. En esta primera aproximación, se opta por redistribuir los datos, consolidando categorías con menos de 50 productos en una clase denominada MA-DIY, totalizando 895 productos. Posteriormente, se aplicó la asignación de representaciones numéricas a variables categóricas utilizando el *ordinal_encoder*, que maneja posibles errores de etiquetado asignando valores predeterminados para valores desconocidos. Finalmente, se llevó a cabo la implementación, entrenamiento y evaluación de los algoritmos seleccionados en este contexto de datos ajustado.
- Se llevó a cabo un Análisis de Componentes Principales (PCA) con el objetivo de comprender cómo cada componente contribuye a explicar la variabilidad en los datos al reducir su dimensionalidad. La gráfica de varianza acumulada reveló que aproximadamente el 95% de la variabilidad se explicaba con diecisiete componentes principales. A pesar de obtener una precisión balanceada de 0.94 al aplicar la clasificación con Random Forest utilizando estos componentes, se determinó que mantener 17 componentes no generaba una diferencia

significativa en términos computacionales. Dado que el conjunto de datos tenía solo 22 variables y el beneficio de la reducción de dimensionalidad era limitado, se concluyó que el uso de PCA resultaba innecesario para este caso de negocio. Además, se señaló que aplicar PCA implicaría una transformación adicional en los datos durante su procesamiento, lo cual no aportaría un beneficio significativo. Por lo tanto, se decidió prescindir del uso de PCA en este contexto.

- Se implementó el algoritmo Random Forest sobre datos balanceados mediante la técnica de SMOTE, utilizando los hiper-parámetros por defecto, lo que condujo a una notable precisión del 95.4%. El análisis del f1-score revela un equilibrio satisfactorio entre precisión y recall en la mayoría de las clases predichas en comparación con los valores reales de y, indicando una aproximación a 1. Tras la aplicación del modelo, se observa un aumento significativo en la cantidad de datos, pasando de 62,424 a 393,159 registros. Estos resultados destacan la eficacia de la combinación de Random Forest y SMOTE para mejorar la capacidad predictiva del modelo y resaltar la importancia de gestionar el desbalance de datos en problemas de clasificación.
- En el análisis de los resultados obtenidos mediante Random Forest, XGBOOST y Bagging con hiper-parámetros por defecto, se observó que los modelos demostraron un rendimiento destacado con precisiones de 0.947, 0.9496 y 0.9494, respectivamente. El f1-score revela un equilibrio efectivo entre la precisión y la exhaustividad (*recall*) en la mayoría de las clases predichas en comparación con los valores reales de *y*, mostrando valores cercanos a 1. Estos resultados sugieren que los modelos son capaces de realizar predicciones precisas y capturar adecuadamente tanto los verdaderos positivos como los verdaderos negativos, indicando un rendimiento robusto en la clasificación de datos.
- En este estudio, se empleó *GridSearchCV* para optimizar hiper-parámetros en la evaluación de modelos Random Forest y Bagging mediante validación cruzada, resultando en la identificación de los mejores hiper-parámetros. Por otro lado, se prescindió del uso del modelo de Boosting ya que se observó un bajo desempeño en las métricas que indicaban dificultades en la predicción de las clases representativas.

4. TECNOLOGÍA: Ingeniería de Datos y uso de tecnología para:

4.1. Desarrollo del proyecto

El proyecto consta de dos momentos, el de Entrenamiento del Modelo, y Limpieza, Transformación de Datos y Aplicación del Modelo.

4.1.1. Entrenamiento del modelo

Este proyecto se centra en la implementación de una estrategia de aprendizaje automático para la clasificación de categorías fiscales en una empresa de dispositivos biomédicos. El proceso de entrenamiento del modelo abarca varias etapas, desde la carga de datos desde un depósito de AWS S3 hasta la comparación de modelos como Bosques Aleatorios, XGBoost y Bagging. Se lleva a cabo un exhaustivo análisis exploratorio de datos, preprocesamiento, balanceo de datos con SMOTE y evaluación de modelos mediante métricas como precisión, recall y F1-score. El modelo final, basado en Bosques Aleatorios, se entrena en todo el conjunto de datos, incluyendo datos balanceados, y se guarda para uso futuro. La organización del proyecto incluye informes de clasificación, almacenamiento del modelo final y visualizaciones que proporcionan una visión detallada del rendimiento del modelo y la distribución de datos (figura 16).

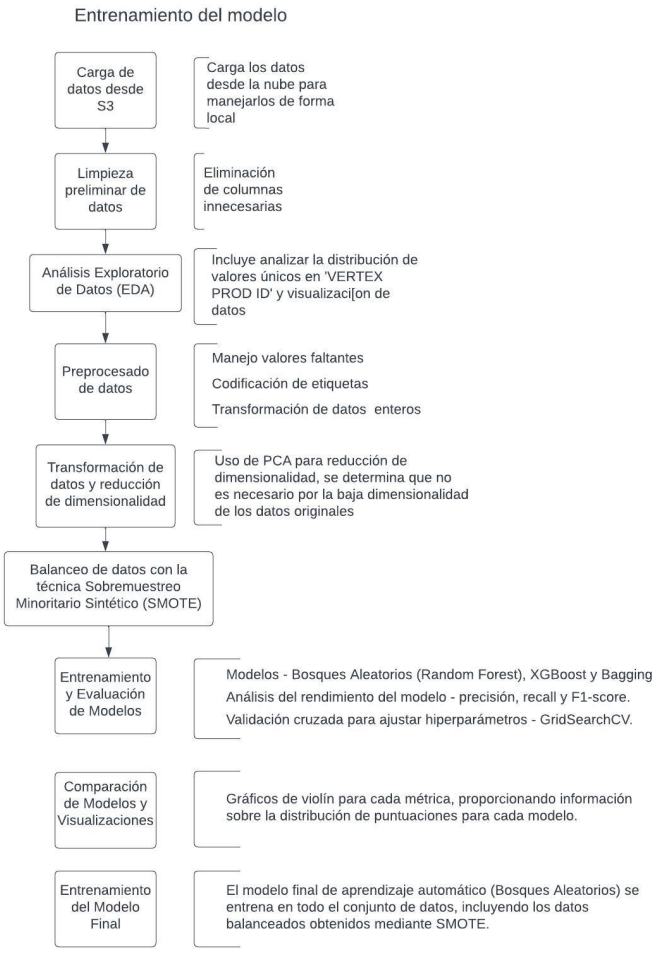


Figura 16. Entrenamiento del modelo.

4.1.2. Limpieza, Transformación de Datos y Aplicación del Modelo

Este proyecto se centra en la implementación de una estrategia de aprendizaje automático para la clasificación de categorías fiscales en una empresa de dispositivos biomédicos. El proceso de entrenamiento del modelo abarca varias etapas, desde la carga de datos desde un depósito de AWS S3 hasta la comparación de modelos como Bosques Aleatorios, XGBoost y Bagging. Se lleva a cabo un exhaustivo análisis exploratorio de datos, preprocesamiento, balanceo de datos con SMOTE y evaluación de modelos mediante métricas como precisión, recall y F1-score. El modelo final, basado en Bosques Aleatorios, se entrena en todo el conjunto de datos, incluyendo datos balanceados, y se guarda para uso futuro. La organización del proyecto incluye informes de clasificación, almacenamiento del modelo final y visualizaciones que proporcionan una visión detallada del rendimiento del modelo y la distribución de datos (Figura 17).

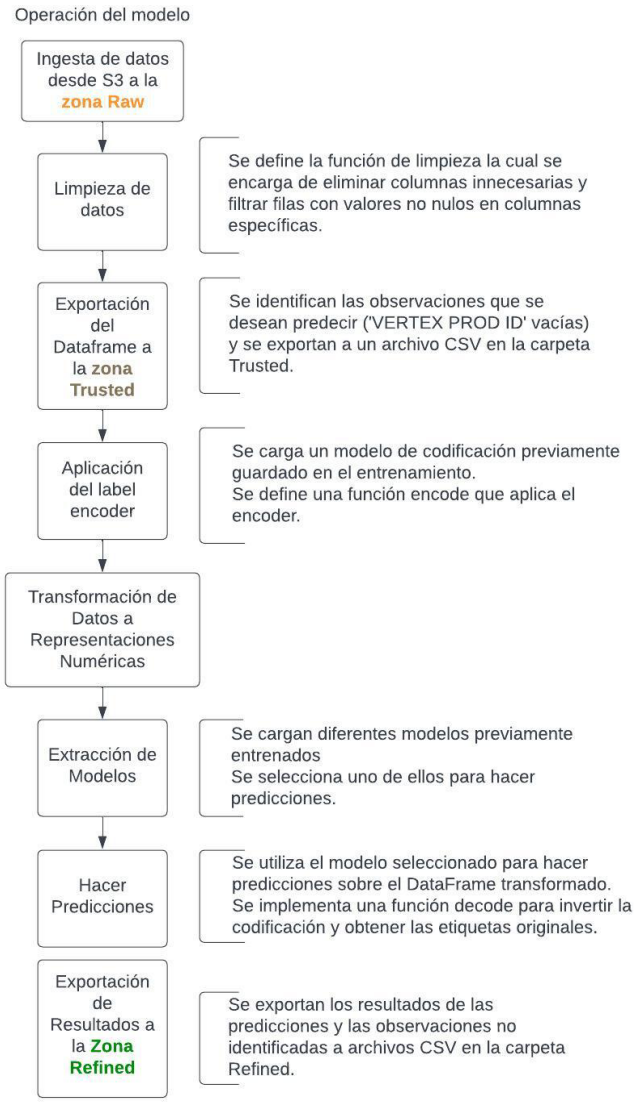


Figura 17. Operación del modelo. Limpieza, transformación de datos y aplicación del modelo.

4.1.3. Visualización de datos con Power BI

Mediante Power BI, se hace la visualización de las predicciones del modelo utilizado. Se crea un dashboard que contiene dos tarjetas, una con el número de predicciones totales y otra con el número de productos dentro de dichas predicciones. Gráficamente es posible visualizar la distribución de las categorías predichas, las cuales fueron 41N (Implantes y prótesis), 48Y (Instrumentos quirúrgicos), 03D (Insumos de un solo uno), y 12Y (Instrumentos quirúrgicos). Se observa que el mayor número de predicciones fue para la categoría 41N. Además, se evidencia de manera jerárquica por medio de un gráfico de anillos la representación de las clases de producto siendo "OTHER TOTALS JOINTS" la que obtiene un mayor porcentaje. Así mismo, por medio de un treemap se muestran organizados los tipos de productos y el peso que toman dentro de las predicciones. La zona de datos nuevos hace referencia a las características que no fueron identificadas y es necesario más información para hacer la predicción de ellas (figura 18).

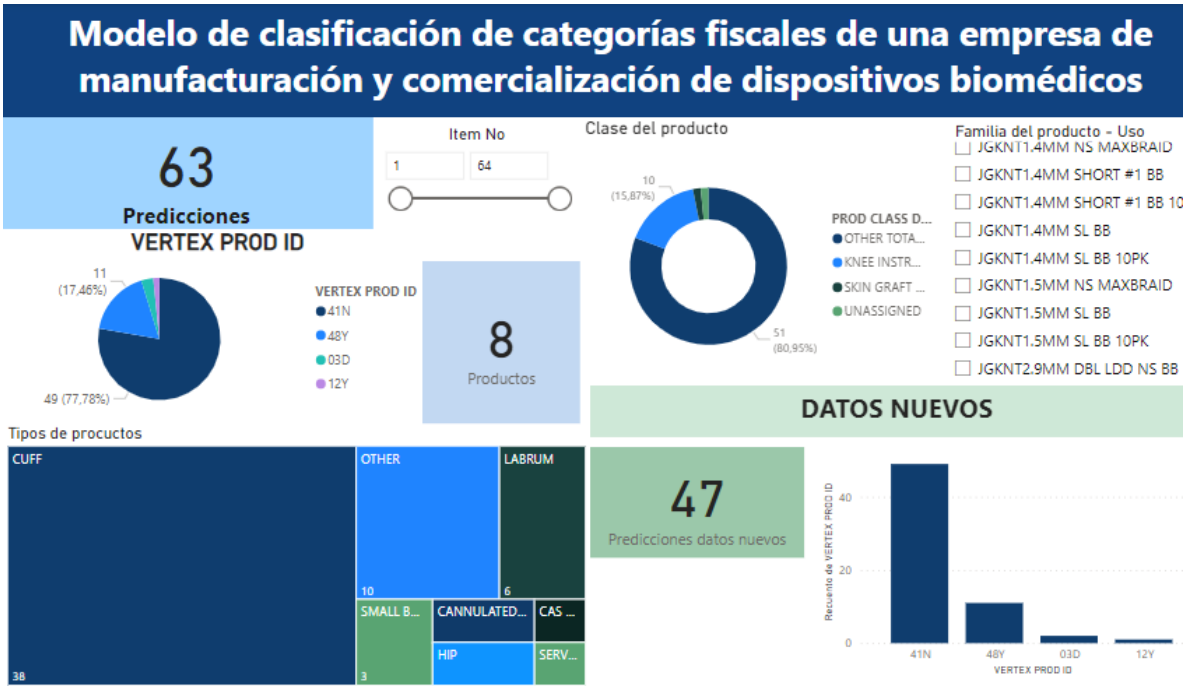


Figura 18. Visualización de predicciones en Power BI

4.1.4. Arquitectura

El procedimiento inicia con la ingesta de datos desde un Bucket en AWS S3 mediante Power Shell. Posteriormente, los datos se almacenan en Batch en un Datalake con AWS S3. El procesamiento y transformación de datos se llevan a cabo utilizando Python 3.12, mientras que el análisis y modelado se realizan mediante herramientas como Jupyter Notebooks y PowerBI. En cuanto a Machine Learning y Modelado, se emplean diversas bibliotecas de Python, como pandas, numpy, seaborn, matplotlib, joblib, scipy, Scikit-learn e Inbalanced-Learn, junto con modelos como Random Forest, Bagging, XGBoost, SMOTE, cross-validation y PCA. El despliegue de modelos se realiza mediante la implementación de archivos .bat, y el monitoreo se efectúa mediante la obtención de archivos .csv con observaciones no clasificables. La gestión de proyectos y colaboración se lleva a cabo con el uso de sistemas de control de versiones como Git. La automatización se logra mediante archivos .bat para nuevas predicciones y el reentrenamiento del modelo según las necesidades de la empresa. En cuanto a la escalabilidad y rendimiento, no se aplica un diseño escalable debido al crecimiento esperado

reducido, y la optimización de rendimiento se centra en la mejora del proceso de ingesta de datos desde S3 a la carpeta local, con una posible oportunidad de mejora teórica al obtener datos preprocesados desde servicios de AWS como S3, Glue y Athena, aunque la implementación real está limitada por la falta de servicios de nube en la empresa (Tabla 9).

Procedimiento	Descripción
Ingesta de datos	- Se recupera la información desde un Bucket en AWS S3 con ayuda de Power shell (comando del sistema).
Almacenamiento de datos	- Almacenamiento en Batch: Datalake con AWS S3.
Procesamiento de datos	- Procesamiento y transformación de datos realizado con Python 3.12.
Análisis y modelado	- Herramientas de Análisis y Exploración de Datos: Empleo de Jupyter Notebooks y PowerBI para la visualización de datos. - Machine Learning y Modelado: Uso de la librería pandas, numpy, seaborn, matplotlib, joblib, scipy, Scikit-learne Inbalanced-Learn de Python, entre los modelos usados se encuentran: Random Forest, Bagging, XGBoost, SMOTE, cross-validation y PCA.
Despliegue y monitoreo de modelos	- Despliegue de modelos: Implementación del modelo de clasificación desde archivos .bat - Monitoreo de Modelo: Obtención de archivo .csv con observaciones no clasificables al tener características nuevas, desconocidas por el modelo.
Gestión de proyectos y colaboración	- Control de versiones: Utilización de sistemas de control de versiones como Git para rastrear cambios en el código y en los modelos.
Automatización	- Uso de archivos .bat para usar el modelo para un nuevo set de predicciones y para reentrenamiento del modelo a necesidad de la empresa a partir de la observación de nuevos datos, no conocidos.
Escalabilidad y rendimiento	- Diseño Escalable: No aplica, debido a que el crecimiento esperado de los datos es reducido dentro del corto y mediano plazo del proyecto. - Optimización de Rendimiento: Para este proyecto, el proceso de mayor demora es la ingesta de datos desde S3 a la carpeta local, una oportunidad de mejora es poder obtener los datos preprocesados desde los servicios de AWS como: (S3 – Glue – Atena), sin embargo, en el caso real, la empresa actualmente no cuenta con servicios de nube como Azure y AWS por lo que lograr esta mejora es meramente teórico.

Tabla 9. Arquitectura implementada en el proyecto.

4.2. Despliegue del proyecto (en un escenario hipotético de implementación real)

Para el despliegue del proyecto se usaría almacenamiento en nube para preservar el archivo más reciente de la base de datos de la empresa. Sobre la matriz de productos y características se realizaría un proceso de ingesta a partir de PowerShell hacia el almacenamiento local y en donde posteriormente se aplicará el flujo de limpieza de datos en un archivo *.py* que hará la ejecución de la transformación de las observaciones que se deben predecir y posterior aplicación del modelo. Posteriormente, se obtendrá un archivo con las predicciones de cada observación y en caso de que existiesen características desconocidas para el modelo. Estas irían de manera separada a otro archivo en donde se almacenarían sugerencias de clases fiscales, pero que finalmente le dirán al usuario final que hay presencia de nuevas características para una revisión más manual y posterior toma de decisión sobre la necesidad de reentrenar el modelo. Finalmente, se visualizan las predicciones realizadas y las observaciones con características nuevas en un tablero interactivo de Power BI en el que se puede ver el resumen detallado de la ejecución (figura 19).

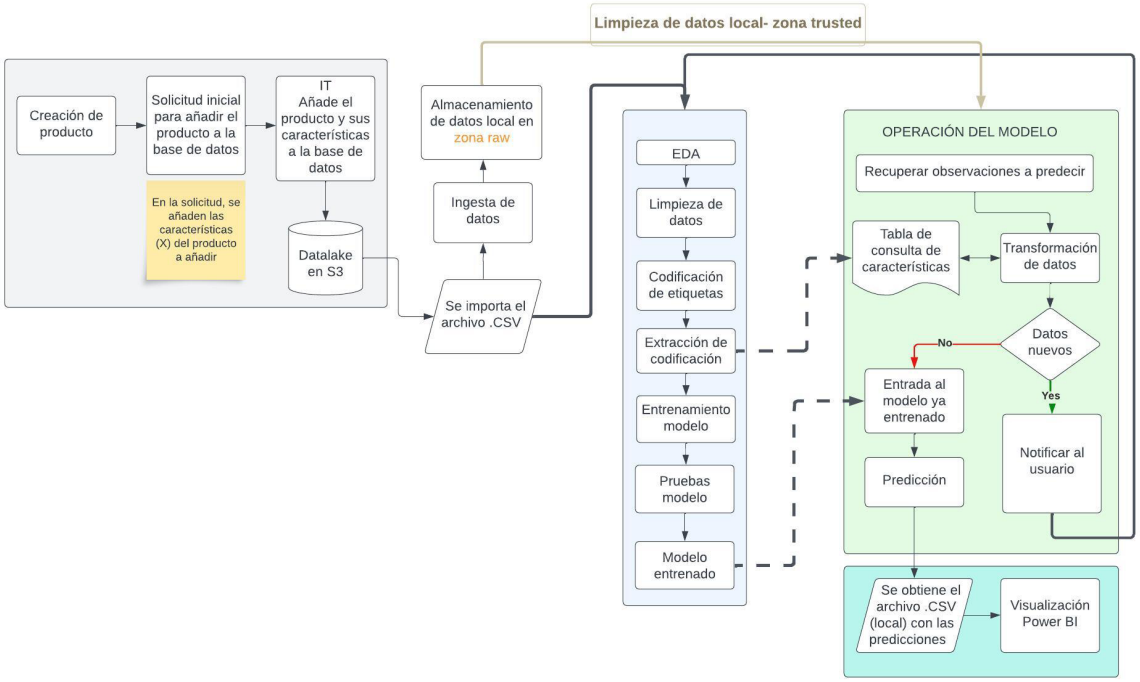


Figura 19. Ciclo de vida de los datos.

5. Conclusiones generales del proyecto

- En este análisis de aproximadamente 63,000 dispositivos biomédicos de una empresa, se identificó un desbalance en las asignaciones de características debido a la diversidad de productos. Ante este desequilibrio en la distribución de datos entre clases, se consolidaron categorías con menos de 50 productos en una nueva clase llamada MA-DIY, totalizando 895 productos. La implementación, entrenamiento y evaluación de algoritmos se llevaron a cabo en este nuevo contexto de datos ajustado, destacando la importancia de manejar el desbalance en la clasificación de dispositivos biomédicos.

- En el estudio, se empleó un Análisis de Componentes Principales (PCA) para reducir la dimensionalidad de un conjunto de datos con 22 variables, pero se determinó que mantener 17 componentes no aportaba beneficios significativos. En lugar de utilizar PCA, se optó por aplicar el algoritmo Random Forest con datos balanceados mediante SMOTE, logrando una precisión del 95.4%. El análisis del f1-score mostró un equilibrio satisfactorio entre precisión y recall, destacando la eficacia de la combinación de Random Forest y SMOTE para mejorar la capacidad predictiva del modelo y resaltando la importancia de abordar el desbalance de datos en problemas de clasificación.

- En resumen, al analizar los resultados de Random Forest, XGBOOST y Bagging con hiper-parámetros por defecto, se encontró un rendimiento destacado con precisiones de 0.947, 0.9496 y 0.9494, respectivamente, y un equilibrio efectivo entre precisión y exhaustividad en el f1-score. Los modelos demostraron capacidad para realizar predicciones precisas y capturar tanto verdaderos positivos como verdaderos negativos, indicando un rendimiento robusto en la clasificación de datos. Además, se utilizó GridSearchCV para optimizar hiperparámetros en Random Forest y Bagging, identificando los mejores ajustes, mientras que se descartó el modelo de Boosting debido a su bajo rendimiento en las métricas, señalando dificultades en la predicción de clases representativas.

- El proyecto se centra en implementar aprendizaje automático para clasificar categorías fiscales en una empresa de dispositivos biomédicos. Consta de dos fases: Entrenamiento del Modelo y Aplicación del Modelo. El entrenamiento incluye la carga de datos desde AWS S3, comparación de modelos como Bosques Aleatorios, XGBoost y Bagging, análisis exploratorio, preprocesamiento, balanceo con SMOTE y evaluación. El modelo final (Bosques Aleatorios) se entrena en todo el conjunto de datos y se guarda. Se utilizan Power BI para visualizaciones, destacando la categoría 41N. El proceso de datos abarca Python, Jupyter Notebooks y PowerBI, con despliegue mediante archivos .bat, monitoreo con archivos .csv y gestión de proyectos con Git. Se automatiza con archivos .bat para nuevas predicciones y reentrenamiento. La escalabilidad no es prioridad debido al crecimiento esperado reducido, y se sugiere mejorar la ingesta de datos desde AWS, limitada por la falta de servicios de nube en la empresa.

- El proyecto implica el uso de almacenamiento en la nube para preservar la base de datos de la empresa. La matriz de productos se procesa mediante PowerShell y limpieza de datos en Python. Se aplican predicciones con un modelo, y los resultados se almacenan de forma local en las diferentes zonas de datos. Las características desconocidas se registran con sugerencias manuales de clases fiscales. La visualización final se presenta en un tablero interactivo de Power BI, proporcionando un resumen detallado de la ejecución.

- En resumen, el proyecto demuestra eficacia en la clasificación de categorías fiscales de dispositivos biomédicos, con énfasis en la gestión de datos, visualización y automatización.

6. Referencias

Aggarwal, C. (2014). Chapter 1. An introduction to cluester analysis. In *Data clustering algorithms and applications* (pp. 1–27). CRC Press Taylor & Francis Group. https://doi.org/10.1201/9781315373515-1

Brownlee, J. (2020). Imbalanced Classification with Python - Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning. *Machine Learning Mastery*, 463.

Chawla, Nitesh 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. https://doi.org/10.1613/jair.953

Chawla, N V, Bowyer, K. W., Hall, L. O., & Philip Kegelmeyer, W. S. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *J Artif Intell Res*, *16*, 321–357.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. In *Learning from Imbalanced Data Sets*. https://doi.org/10.1007/978-3-319-98074-4

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905. https://doi.org/10.1613/jair.1.11192

Ferreira, A. J., & Figueiredo, M. A. T. (2012). Boosting algorithms: A review of methods, theory, and applications. In *Ensemble Machine Learning: Methods and Applications* (pp. 35–85). https://doi.org/10.1007/9781441993267_2

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, *2*(4), 42–47. http://www.ijetae.com/files/Volume2Issue4/IJETAE_0412_07.pdf

Greenacre, M., Groenen, P. J. F., Hastie, T., Iodice D'enza, A., Markos, A., Tuzhilina, E., & Iodice D'enza, A. (2023). Principal Component Analysis. *Nature Reviews Methods*, *1856*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*.

James, G. (2023). *An introduction to Statistical Learning with Application in Python*.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Handling imbalanced datasets : A review. *Science*, *30*(1), 25–36.

Kullarni, V. Y., & Sinha, P. K. (2013). Random Forest Classifier: A Survey and Future Research Directions. *International Journal of Advanced Computing*, *36*(1), 1144–1156.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00349-y

Ziegler, A., & König, I. R. (2014). Mining data with random forests: Current options for real-world applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(1), 55–63. https://doi.org/10.1002/widm.1114