UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Métodos de Clasificación en Python: Aplicaciones a la Empresa

Trabajo Fin de Grado

Grado en Administración y Dirección de Empresas

AUTOR/A: Fuster Coma, Noelia

Tutor/a: Juan Pérez, Ángel Alejandro

Cotutor/a: Carracedo Garnateo, Patricia

CURSO ACADÉMICO: 2022/2023

Resumen

En los últimos años, el análisis de datos se ha convertido en un instrumento fundamental para el éxito de las empresas. Gracias a los avances tecnológicos y a la disponibilidad de enormes volúmenes de datos, las organizaciones pueden obtener información importante sobre sus operaciones y tomar decisiones fundamentadas. Python, uno de los lenguajes de programación más populares del mundo, se ha convertido en una herramienta esencial en las aplicaciones corporativas.

Este trabajo se centra en la evaluación de diversos algoritmos de clasificación de datos utilizando Jupyter Notebooks como herramienta principal. Posteriormente, mediante una aplicación de un caso real, se crearán algoritmos de aprendizaje automático y se evaluará su rendimiento en un conjunto de datos relacionado con el ámbito de la salud. El principal objetivo será verificar si los modelos son capaces de predecir el tipo de obesidad de un adolescente.

Palabras clave: Análisis de datos, modelos de clasificación, aplicaciones empresariales, obesidad, Python.

Resum

En els últims anys, l'anàlisi de dades s'ha convertit en un instrument fonamental per a l'èxit de les empreses. Gràcies als avanços tecnològics i a la disponibilitat d'enormes volums de dades, les organitzacions poden obtindre informació important sobre les seues operacions i prendre decisions fonamentades. Python, un dels llenguatges de programació més populars del món, s'ha convertit en una eina essencial en les aplicacions corporatives.

Aquest treball se centra en l'avaluació de diversos algorismes de classificació de dades utilitzant Jupyter Notebooks com a eina principal. Posteriorment, mitjançant una aplicació d'un cas real, es crearan algorismes d'aprenentatge automàtic i s'avaluarà el seu rendiment en un conjunt de dades relacionat amb l'àmbit de la salut. El principal objectiu serà verificar si els models són capaços de predir el tipus d'obesitat d'un adolescent.

Paraules clau: Anàlisi de dades, models de classificació, aplicacions empresarials, obesitat, Python.

Abstract

In recent years, data analytics has become a critical tool for business success. Thanks to technological advances and the availability of huge volumes of data, organisations can gain important insights into their operations and make informed decisions. Python, one of the world's most popular programming languages, has become an essential tool in corporate applications.

This paper focuses on the evaluation of various data sorting algorithms using Jupyter Notebooks as the main tool. Subsequently, by means of a real case application, machine learning algorithms will be created and their performance will be evaluated on a health-related dataset. The main objective will be to verify if the models are able to predict the type of obesity of an adolescent.

Keywords: Data analytics, classification models, business applications, obesity, Python.

Agradecimientos

A mi pareja, Alfonso González, por ser mi pilar y mi compañero de vida, por apoyarme durante todo el camino.

A aquellas personas que están o han estado en mi etapa por la universidad y han hecho que yo sea la persona que hoy soy.

A mi tutor, Ángel A. Juan, por darme esta oportunidad y por creer y ver algo en mí.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS:

IA: Inteligencia artificial

ML: Machine Learning

Knn: K-Nearest Neighbors (K vecinos más cercanos)

ODS: Objetivos de Desarrollo Sostenible

SVM: Support Vector Machine (Máquina de Vectores de Soporte)

SEEDO: Sociedad Española de Obesidad

TP / FP: True Positive / False Positive

TN / FN: True Negative / False Negative

1. Introducción

1.1. Contexto

Actualmente, la sociedad está inmersa en una revolución industrial caracterizada por el uso de las tecnologías como motor del cambio. La inteligencia artificial (IA) está siendo adoptada en todos los ámbitos posibles, tanto en empresas públicas como privadas, lo cual permite cambiar la forma en que las organizaciones operan.

La importancia del desarrollo de la IA radica en su potencial para impulsar el crecimiento económico y mejorar la calidad de vida. Estas tecnologías pueden resolver problemas muy complejos con mayor facilidad, rapidez y eficacia que si lo realizase un humano.

En los últimos años se ha ido perfeccionando el aprendizaje automático, es decir, el machine learning. Esta herramienta ha sido muy útil para poder analizar grandes volúmenes de datos y extraer información relevante para la toma de decisiones. Asimismo, debido a que permiten identificar patrones y tendencias, su uso ha pasado a tener un papel muy importante para la detección de oportunidades y en la predicción de los resultados.

Estas aplicaciones se están llevando a cabo incluso en el ámbito de la salud, donde el uso de la inteligencia artificial ha revolucionado la forma en la que se diagnostican y tratan enfermedades. Gracias a estas herramientas, los hospitales son capaces de analizar grandes cantidades de datos clínicos con el fin de identificar patrones y predecir enfermedades con mayor precisión. Esto permite una detección temprana de las enfermedades, una mejora en los resultados de los tratamientos y una reducción de los costes médicos.

1.2. Objetivos del trabajo

El principal objetivo del presente trabajo es analizar los diferentes modelos de aprendizaje automático supervisado con el fin de llegar a predecir el tipo de obesidad que tiene un paciente. Para ello, se tendrá en cuenta diversas variables y características como el peso, la altura y su historial familiar, entre otros.

A nivel personal, el objetivo buscado es introducirse en un campo no estudiado anteriormente, donde a través de la curiosidad por aprender y el afrontamiento de nuevos retos pueda entender, comprender y aplicar el marco teórico y la implementación de algoritmos de clasificación.

Para su consecución, se presentan además otros fines como aprender a usar una interfaz web de código abierto, en este caso Jupyter Notebooks, así como la librería “sklearn”, muy utilizada en el ámbito laboral del sector tecnológico.

1.3. ODS

En este punto se analizará la relación existente entre el presente TFG y los Objetivos de Desarrollo Sostenible (ODS).

Los ODS son un conjunto de 17 objetivos interrelacionados con 169 metas concretas, que abarcan una amplia gama de temas que son cruciales para el desarrollo sostenible; con el fin de erradicar mejorar la prosperidad de la sociedad y proteger el planeta. Estos fueron adoptados por los 193 estados miembros de las Naciones Unidas en 2015.

En la siguiente tabla se muestra los distintos ODS existentes junto con el grado de relación con el Trabajo de Fin de Grado.

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar	X			
ODS 4. Educación de calidad			X	
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminable				X
ODS 8. Trabajo decente y crecimiento económico				X
ODS 9. Industria, innovación e infraestructura	X			
ODS 10. Desigualdades				X
ODS 11. Ciudades y comunidades sostenibles				X
ODS 12. Producción y consumo responsable				X
ODS 13. Acción por el clima				X
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr los objetivos				X

Tabla 1: Relación de los ODS con el TFG. Elaboración propia.

De entre todos los objetivos de los ODS, aquellos que son más relevantes y tiene mayor relación con el presente trabajo son: ODS 3, Salud y bienestar; ODS 4, Educación de calidad; y ODS 9, Industria, innovación e infraestructura.

Uno de los objetivos con mayor grado de relación es el de la salud y bienestar. Esto es debido a que el presente documento tiene como fin aplicar algoritmos de clasificación a una base de datos de obesidad infantil, por lo que podría ayudar a mejorar la salud de los niños.

El siguiente ODS relacionado es el de Industria, innovación e infraestructura. El análisis de datos y los algoritmos están siendo una parte esencial de la tecnología actual. Debido a la gran innovación que ha surgido en los últimos años, la ciencia de datos está siendo aplicada en multitud de campos, incluyendo el ámbito empresarial y la medicinal.

Aunque este proyecto no esté directamente relacionado con el ODS 4, los resultados obtenidos pueden utilizarse para mejorar la calidad de la educación al promover prácticas de vida saludables y proporcionar oportunidades de aprendizaje en áreas importantes como la ciencia de datos y la salud.

1.4. Metodología

Para poder realizar con éxito el marco teórico de este trabajo se ha debido de hacer lecturas exhaustivas sobre libros, revistas y varios documentos científicos, además de diversas consultas a los materiales ofrecidos por el tutor de este TFG, Ángel A. Juan.

Para el entendimiento de la base de datos y sus variables, al tratarse de un tema de la salud, se ha recurrido numerosas veces la página oficial del ministerio de Sanidad, de la Sociedad Española de Obesidad (SEEDO) y de la Organización Mundial de la Salud.

Para la construcción de los algoritmos, se han realizado diversos cursos de programación del lenguaje Python a través de Microsoft y de Edx.

1.5. Orden documental

En este apartado se indica la estructura del presente documento. El orden de los capítulos es el siguiente:

Capítulo 1. Introducción

Este capítulo es la presentación del tema y de los objetivos del trabajo. También se proporciona una justificación de la relevancia del tema y una contextualización de la digitalización de las empresas en la revolución industrial 4.0.

Capítulo 2. Marco teórico

Este capítulo se enfoca en la definición de los conceptos básicos del machine learning y los diferentes tipos de aprendizaje automático. Además, se profundizará en 6 algoritmos de clasificación que serán utilizados posteriormente para el caso real de aplicación: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbour (KNN) y Naive Bayes.

Capítulo 3. Aplicación práctica: caso real

En este capítulo se describe el dataset que se va a utilizar para aplicar los algoritmos de clasificación. Se presenta el preprocesamiento de los datos para posteriormente pasar a realizar una estadística descriptiva que incluye un análisis multivariable y bivariable. Seguidamente se desarrollan los algoritmos y, para comparar su eficacia, es introducido un sujeto nuevo con datos reales. El objetivo será que los algoritmos puedan predecir el tipo de obesidad de este individuo.

Capítulo 4. Conclusiones y propuestas de mejora

En este capítulo se hace una recapitulación de los objetivos del trabajo y de los principales hallazgos obtenidos. Se hace una reflexión crítica sobre las limitaciones del estudio y se presentan posibles vías de investigación futura.

2. Marco Teórico

En este apartado se introduce la definición de Machine Learning por su creador para, posteriormente, indicar los principales tipos de aprendizaje automático, es decir, el aprendizaje reforzado, no supervisado y supervisado. Se profundizará, además, en este último mencionado, explicando mediante aplicaciones prácticas algunos de los algoritmos más usados habitualmente para la ayuda a la toma de decisiones empresariales.

2.1 Machine Learning

Según Samuel Arthur (1959), el aprendizaje automático se define como el aprendizaje sin necesidad de programación explícita. Es decir, esta rama de la inteligencia artificial pretende enseñar a los ordenadores, mediante el uso de algoritmos, a aprender y mejorar a partir de la experiencia. A diferencia de la programación tradicional, en el machine learning no es necesario programar de forma detallada cada tarea que la computadora debe realizar. En realidad, es suficiente con proporcionarles algunos conjuntos de ejemplos para que el modelo los entrene y puedan "aprender" a generalizarlo hacia nuevas situaciones.

Principalmente, su objetivo es desarrollar modelos capaces de asimilar, preparar y profundizar de forma autónoma a partir de los datos, sin necesidad de intervención humana constante.

Existen tres tipos principales de machine learning:

- Aprendizaje supervisado: Se basa en el uso de datos etiquetados para entrenar algoritmos capaces de predecir las etiquetas de nuevos datos. El modelo aprende a partir de las relaciones entre las entradas y las salidas conocidas, y luego utiliza ese conocimiento para hacer predicciones o clasificaciones de nuevos datos. Es

por esto que, según la finalidad que se le quiera dar, existen dos subtipos principales: clasificación y regresión.

- Aprendizaje no supervisado: Es una técnica de machine learning que se emplea cuando los datos no se encuentran previamente etiquetados. En esta modalidad, el algoritmo analiza los datos sin ninguna guía explícita para encontrar patrones y estructuras por sí mismo. Su principal objetivo es encontrar patrones y relaciones ocultas en los datos y aprender a clasificarlos de manera adecuada.
- Aprendizaje por refuerzo: Es un tipo de aprendizaje automático en el que un modelo aprende a través de ensayo y error en un entorno específico para maximizar una recompensa. El modelo recibe retroalimentación en forma de recompensas o penalizaciones según las decisiones tomadas. Juegos como AlphaGo (DeepMind Technologies, 2016) o Pacman (Namco, 1980) son utilizados como campos de prueba para el aprendizaje por refuerzo: el agente recibe información sobre las reglas del juego y aprende a jugar por sí mismo, empezando con comportamientos aleatorios y mejorando con el tiempo.

De los tipos mencionados anteriormente, el presente documento se centrará en el aprendizaje supervisado.

2.2. Aprendizaje supervisado

Tal y como se ha comentado, el aprendizaje supervisado es una técnica de machine learning que consiste en entrenar al algoritmo con datos históricos etiquetados para que aprenda a asignar la etiqueta adecuada a nuevos datos (Arthur Samuel, 1959).

En este tipo de aprendizaje, el algoritmo utiliza las preguntas o atributos definidos en los datos de entrenamiento y sus correspondientes respuestas o etiquetas para realizar predicciones

sobre datos futuros. Asimismo, es considerado el tipo de ML más maduro y utilizado, ya que las máquinas pueden hacer uso de sus experiencias pasadas y aplicarlas a nuevos datos.

Las aplicaciones de este método son útiles cuando los registros históricos permiten obtener predicciones fiables sobre casos futuros, como por ejemplo en la clasificación de correos electrónicos para detectar si es spam o no, las instituciones financieras para evaluar el riesgo crediticio de un solicitante o las empresas para prever la demanda en función de datos históricos de ventas.

Dentro del aprendizaje supervisado, existen dos principales algoritmos en función del tipo de datos y del problema que se quiera resolver: regresión y clasificación.

Según Galton (1886), el término "regresión" se refiere a la tendencia de los datos a "regresar" a la media en la distribución de una variable. Sin embargo, fue Arthur Samuel (1959) el quien popularizó este término en el contexto del aprendizaje supervisado al utilizarlo para describir la tarea de predecir una variable continua a partir de un conjunto de datos de entrada.

Este método es muy usado en la estadística y, tal y como se ha explicado anteriormente, consiste en entrenar a un algoritmo para que encuentre una relación entre las características de entrada y las variables de salida; con el objetivo de poder predecir un valor numérico continuo con la mayor exactitud posible. Algunos ejemplos de la regresión con aprendizaje supervisado son pronosticar el precio de una casa basado en su ubicación y tamaño, o predecir las ventas de un producto según su precio.

A diferencia de la regresión, la clasificación tiene como objetivo la búsqueda de patrones con el fin de poder clasificar los elementos en diferentes grupos, teniendo en este caso variables de salida categórica.

En la vida cotidiana es muy frecuente la utilización de estos algoritmos. Algunos ejemplos son los asistentes virtuales de Alexa o de Siri, los cuales usan la clasificación para poder reconocer la voz de sus usuarios, de tal forma que puedan identificar las palabras, entender las instrucciones y realizas las tareas encomendadas.

Otra plataforma que usa los algoritmos de clasificación es Netflix. Esta empresa utiliza técnicas de aprendizaje automático con el objetivo de recomendar series y películas según el historial de visualización y las calificaciones de cada usuario. De esta forma, pueden sugerir contenido que se ajuste a sus gustos y preferencias.

Por tanto, de forma resumida y a grandes rasgos, se podría indicar que la principal característica que diferencia estos dos métodos es que la clasificación se utiliza para predecir etiquetas categóricas, es decir, la pertenencia a una categoría determinada, mientras que la regresión se utiliza para predecir etiquetas numéricas, es decir, un valor numérico específico.

2.3. Algoritmos de clasificación.

Dentro de la clasificación, existen numerosos algoritmos y constantemente se van desarrollando más y mejorando nuevos. No obstante, hay ciertos métodos que son los más extendidos y ampliamente usados en diversos campos. Este trabajo se centrará en: Naive Bayes, Regresión logística, K-vecinos más cercanos (KNN), Árbol de decisión, Máquina de vectores de soporte (SVM), y Bosque aleatorio.

2.3.1. Naive Bayes

Naive Bayes, o también traducido como “El Ingenuo Bayes”, fue desarrollado por un matemático y filósofo llamado Thomas Bayes (1701-1761). Este algoritmo es ampliamente utilizado en la actualidad debido a su simplicidad y eficiencia en la clasificación de datos.

Se basa en el teorema de Bayes y asume que las variables predictoras son independientes entre sí (Chandra et al., 2007). La idea que sigue el algoritmo es calcular la probabilidad de que una observación pertenezca a una determinada clase dependiendo de sus características, asumiendo la independencia de las variables (Russell, 2016).

A continuación, se muestran los pasos necesarios a seguir para poder llevar a cabo este algoritmo en problemas de clasificación:

1- Transformar el conjunto de datos en una tabla que muestre la frecuencia con la que aparecen las diferentes variables.
2- Establecer una tabla de probabilidad que refleje las posibilidades de que sucedan los diferentes eventos.
3- Calcular la probabilidad posterior de cada clase.
4- El resultado de la predicción se determina por la clase que tenga la probabilidad posterior más alta.

Aunque el modelo proporciona múltiples ventajas como su facilidad y rapidez de predecir las clases, la suposición de independencia no siempre se cumple en la práctica. Esto es debido a que es casi imposible que se obtengan un conjunto de predictores completamente independientes. No obstante, el algoritmo sigue siendo muy útil en muchos casos, especialmente cuando se trata de problemas de clasificación de texto, como la detección de spam o la clasificación de noticias.

A continuación, se expondrá un breve ejemplo de clasificación de noticias. Para su ejecución, se ha utilizado la librería sklearn. Esta es una biblioteca de aprendizaje automático de código abierto que proporciona una amplia gama de algoritmos y herramientas para el aprendizaje automático, incluyendo la clasificación. De esta forma, se consigue facilitar el desarrollo y la implementación de los modelos.

Dentro de esa librería están recogidos una serie de conjuntos de datos que pueden ser utilizados para fines prácticos. La base de datos a utilizar en este ejemplo se denomina “20 Newsgroups”, la cual es una colección de aproximadamente 20,000 documentos de grupos de noticias en 20 temas diferentes. El objeto será predecir la categoría de nuevos documentos de en función de su contenido temático. Algunos de los temas incluidos son política, deportes, tecnología y religión.

En primer lugar, se importa la librería a utilizar y la base de datos (Figura 1). Posteriormente se observará el número de muestras en cada categoría para confirmar que se está utilizando una base de datos equilibrada (Figura 2).

```
import pandas as pd

# Se carga el conjunto de datos a utilizar
dataset = fetch_20newsgroups(subset="all", shuffle=True)
```

Figura 1: Importación de la base de datos. Elaboración propia.

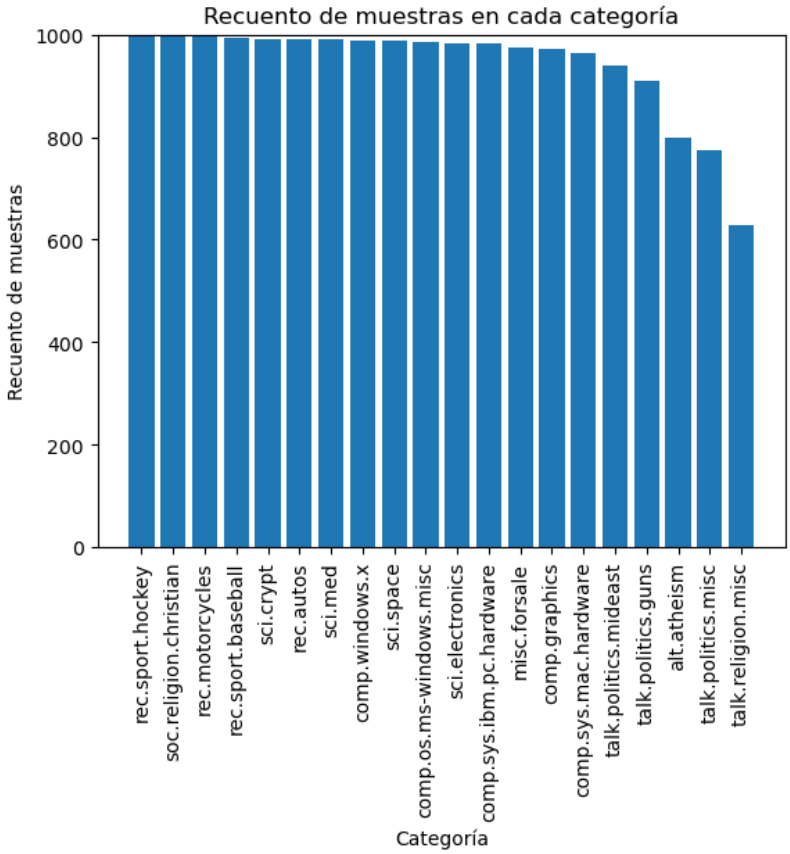

Figura 2: Recuento de muestras en cada categoría. Elaboración propia

Tal y como se puede observar en la Figura 2, la base de datos está en cierto modo balanceada, puesto que aproximadamente todas las variables tienen un número similar de muestras, donde la proporción de muestras en cada clase es de entre un 2'7% y un 4'2% respecto al total.

A continuación, se llevará a cabo en la Figura 3 el procesamiento de los datos. Para ello, se ha realizado los pasos necesarios anteriormente explicados con el fin de poder realizar el desarrollo del algoritmo y la posterior predicción en el conjunto de prueba.

Procesamiento de los datos

```
import pandas as pd

# Se carga el conjunto de datos a utilizar
dataset = fetch_20newsgroups(subset="all", shuffle=True)
```

```
# Se definen las etiquetas y características
X = dataset.data
y= dataset.target

#Se separan los datos en entrenamiento y prueba
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
#Se transforman los datos en una tabla de frecuencia utilizando CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.fit_transform(X_test)
```

Desarrollo del algoritmo

```
#Al tener más de dos etiquetas, se entrena un clasificador Naive Bayes Multinomial
from sklearn.naive_bayes import MultinomialBN
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_counts, y_train)

#Se realizan predicciones en el conjunto de prueba
y_pred = nb_classifier.predict(X_test_counts)
```

Figura 3: Procesamiento y desarrollo del algoritmo. Elaboración propia.

Tras importar los datos, estos son separados, siendo “X” el contenido del texto de las noticias, e ”y” las categorías o temas a los que pertenecen esas noticias.

Seguidamente, se utiliza la función train_test_split para dividir los datos en conjuntos de entrenamiento y de prueba. En este caso, se establece en 0’2, lo que significa que el 20% de los datos se utilizarán como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento.

Cuando se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split, se separan tanto los datos de características (X) como las etiquetas (y). X_train

y X_test contienen las características de entrenamiento y prueba, respectivamente, mientras que y_train e y_test contienen las etiquetas correspondientes a los conjuntos de entrenamiento y prueba, respectivamente.

A continuación, se desarrolla el algoritmo clasificador de Naive Bayes multinomial para poder instruirlo con los datos de entrenamiento. X_train_counts representa las características transformadas en una tabla de frecuencia utilizando CountVectorizer, y y_train son las etiquetas correspondientes a los datos de entrenamiento.

Tras ejecutar este código, el clasificador Naive Bayes multinomial estará entrenado y listo para realizar predicciones en nuevos datos. Sin embargo, se debe comprobar el funcionamiento del modelo creado. Para ello se procede a realizar el análisis de los resultados (Figura 4):

Análisis de los resultados

```
#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average='macro')
print('Sensibilidad del modelo:', sensibilidad)

#Calcular la precisión del modelo

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)

print("Precisión del modelo:", accuracy)
```

```
Sensibilidad del modelo: 0.8506632441578927
Precisión del modelo: 0.8594164456233422
```

Figura 4: Análisis del modelo creado de regresión logística. Elaboración propia.

La sensibilidad se enfoca en la capacidad del modelo para identificar correctamente las instancias positivas, mientras que la precisión se centra en la exactitud de las predicciones positivas. El resultado obtenido indica que el modelo es capaz de identificar correctamente el 85% de todas las instancias positivas reales y que el 86% de todas las instancias clasificadas como positivas por el modelo son verdaderamente positivas.

Por lo tanto, en este caso, una alta sensibilidad indica que el modelo es capaz de recuperar una gran proporción de documentos relevantes, mientras que una alta precisión indica que el modelo tiene una menor tasa de documentos irrelevantes identificados como relevantes.

2.3.2. Regresión logística

La regresión logística fue introducida por el matemático Joseph Berkson (1944) como un "método para modelar la probabilidad de un evento binario en función de una o más variables predictoras" (vol. 29, nº 227).

Este método utiliza un algoritmo de clasificación binaria fácil de implementar. Además, es usado con frecuencia como punto de partida para resolver problemas de clasificación. Su objetivo es estimar la relación entre una variable dependiente y una o más variables independientes, donde la variable dependiente toma valores binarios, es decir, solo puede tomar uno de dos valores posibles, como por ejemplo: verdadero/falso, fracaso/éxito, 1/0, positivo/negativo, etc. Asimismo, es ampliamente utilizado en problemas sobre la detección de spam, la predicción de enfermedades o la predicción de abandono en compra en línea.

El nombre de este algoritmo viene dado debido a la función utilizada: función logística o sigmoide. Esta función se representa mediante una curva en forma de S que tiene la capacidad de asignar a cualquier número real un valor entre 0 y 1 (Figura 5).

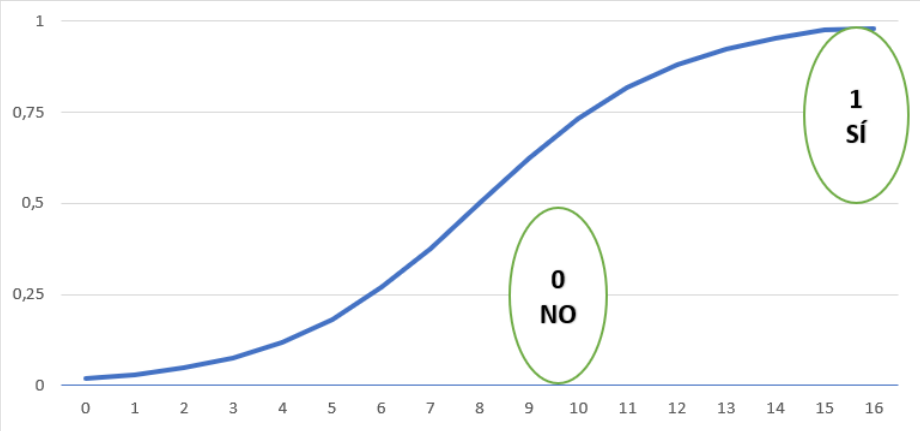

Figura 5: Ejemplo de una función logística. Fuente: Elaboración propia.

En el caso de que la curva de la función tienda a infinito positivo, la predicción se establecerá en 1, mientras que si la curva se acerca al infinito negativo, la predicción será 0. Al asignar la clasificación, si la salida de la función Sigmoide es mayor a 0’5, se puede considerar como un resultado positivo o "SI", y si es menor a 0’5, se puede clasificar como un resultado negativo o "NO". Además, se pueden expresar los resultados en términos de probabilidad, por ejemplo, si el resultado es 0’75, se puede interpretar como una probabilidad del 75% de que un correo recibido sea spam.

A continuación, se realizará este algoritmo para comprobar si un correo pertenece a la categoría de spam o no. En la Figura 6 se muestra el procedimiento llevado a cabo:

Importación de los datos

```
import pandas as pd
url = 'https://raw.githubusercontent.com/JoaquinAmatRodrigo/Estadistica-machine-learning-python/master/data/spam.csv'
dataset = pd.read_csv(url)
```

```
print('Información en el dataset:')
print(dataset.keys())
```

```
Información en el dataset:
Index(['make', 'address', 'all', 'num3d', 'our', 'over', 'remove', 'internet',
       'order', 'mail', 'receive', 'will', 'people', 'report', 'addresses',
       'free', 'business', 'email', 'you', 'credit', 'your', 'font', 'num000',
       'money', 'hp', 'hpl', 'george', 'num650', 'lab', 'labs', 'telnet',
       'num857', 'data', 'num415', 'num85', 'technology', 'num1999', 'parts',
       'pm', 'direct', 'cs', 'meeting', 'original', 'project', 're', 'edu',
       'table', 'conference', 'charSemicolon', 'charRoundbracket',
       'charSquarebracket', 'charExclamation', 'charDollar', 'charHash',
       'capitalAve', 'capitalLong', 'capitalTotal', 'type'],
      dtype='object')
```

Procesamiento de los datos

```
X = dataset.drop('type', axis=1)  # Selecciona todas las columnas excepto la columna 'type'
y = dataset['type']  # Etiquetas (spam y nonspam)

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Separar los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Escalar los datos
escalar = StandardScaler()
X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)
```

Desarrollo del algoritmo

```
#Definir el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo = LogisticRegression()

#Entrenar el modelo
algoritmo.fit(X_train, y_train)

#Realizar una predicción
y_pred = algoritmo.predict(X_test)
```

Figura 6: Código del algoritmo de regresión logística para la clasificación de correos spam. Elaboración propia.

Una vez importados los datos, se procede a su procesamiento, donde se han separado las características (columnas) del conjunto de datos en X, excluyendo la columna 'type'. La variable objetivo (spam o nonspam) se guarda en la variable y.

Seguidamente, se utiliza la función train_test_split para dividir los datos en conjuntos de entrenamiento y de prueba. En este caso, se establece en 0’2, lo que significa que el 20% de los datos se utilizarán como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento.

A continuación, se procede a desarrollar el algoritmo de regresión logística y se entrena el modelo utilizando los datos de entrenamiento. Una vez realizado este paso, se usa el modelo entrenado para predecir las etiquetas de spam o nonspam en el conjunto de prueba.

Para comprobar que la funcionalidad del modelo es correcta, se procederá a analizar los resultados de la predicción realizada con el conjunto de datos de prueba (X_test). Para ello, se puede realizar a partir de la matriz de confusión y la exactitud obtenida (Figura 7):

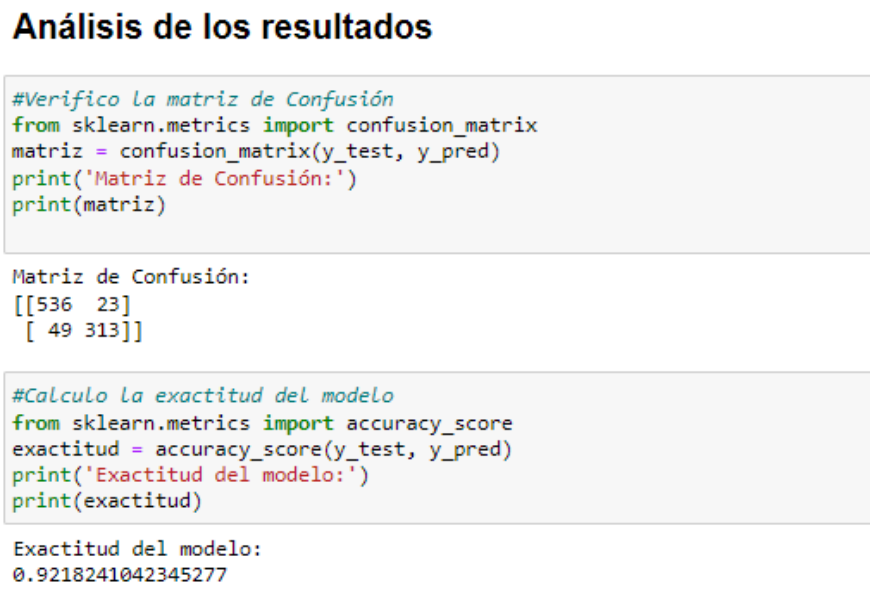

Figura 7: Análisis de los resultados del modelo. Elaboración propia.

En este caso, la matriz de confusión obtenida es la siguiente (Tabla 2):

	Predicción Negativa	Predicción Positiva
Valor Real Negativo	536 (TN)	23 (FP)
Valor Real Positivo	49 (FN)	313 (TP)

Tabla 2: Matriz de confusión para el modelo predictor de "spam". Elaboración propia.

Esto significa que el modelo clasificó correctamente 536 muestras como negativas (True Negatives), clasificó erróneamente 23 muestras negativas como positivas (False Positives), clasificó erróneamente 49 muestras positivas como negativas (False Negatives) y clasificó correctamente 313 muestras como positivas (True Positives).

Asimismo, la exactitud del modelo es una métrica que indica qué tan bien se desempeñó el modelo en general. Se calcula dividiendo el número de predicciones correctas (TP + TN) entre el número total de muestras (TP + TN + FP + FN)[1]. En este caso, la exactitud obtenida es de 0’9218, lo que indica que el modelo tuvo un rendimiento bastante bueno, con una precisión del 92.18%.

Por lo tanto, de forma resumida se podría concluir que el modelo ha logrado clasificar correctamente la mayoría de las muestras.

2.3.3. K-vecinos más cercanos

K-vecinos más cercanos (KNN por sus siglas en inglés) es un algoritmo de aprendizaje no paramétrico[2] y basado en instancias[3] comúnmente utilizado en problemas de clasificación. Alpaydin (2020) lo define como "un algoritmo de clasificación que utiliza los K ejemplos más

[1] Mirar la Tabla 2 para recordar el significado de cada abreviación y su posición en la matriz.
[2] Un algoritmo no paramétrico no asume que los datos siguen una distribución específica. Esto significa que el algoritmo no está limitado por supuestos en cuanto a la forma de distribución de los datos y puede adaptarse mejor a patrones complejos o no lineales que los algoritmos paramétricos.
[3] El aprendizaje basado en la instancia utiliza ejemplos previos para aprender a clasificar cosas nuevas en vez de seguir un conjunto de reglas.

cercanos para predecir la etiqueta de una nueva instancia" (p. 205). Es decir, este método usa las K instancias más cercanas para predecir la clase de una nueva en función de las etiquetas de clase conocidas de las instancias vecinas.

Como se ha explicado, el algoritmo asigna a un objeto la pertenencia a una clase basándose en la mayoría de los votos de sus k vecinos más cercanos. Si k es igual a 1, entonces la clase del objeto simplemente se asigna a la clase del vecino más cercano. En la siguiente figura se ha representado gráficamente el modelo con dos ejemplos: k = 3 y k = 7.

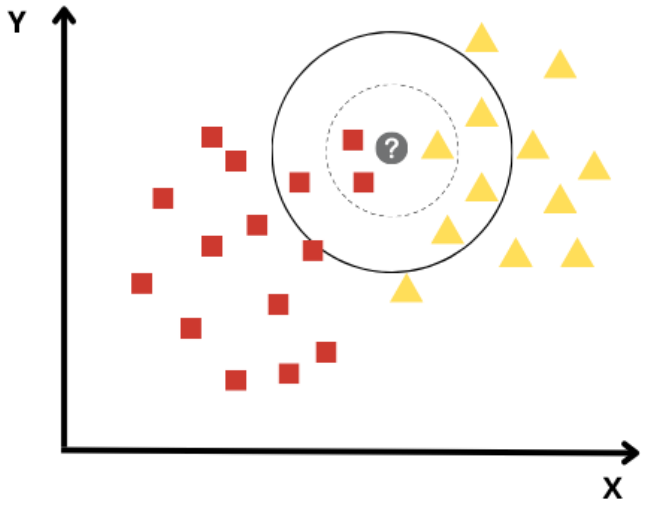

Figura 8: Ejemplo del algoritmo KNN con K = 3 y K = 7. Fuente: Elaboración propia.

En Figura 8, se quiere predecir cuál será la categoría del nuevo punto, pudiendo ser triángulo o cuadrado. Si k = 3, significará que se predecirá su categoría en función de los 3 vecinos más cercanos, ubicados dentro del círculo discontinuo. En ese caso, el nuevo punto será un cuadrado, debido a que en esa área existen 2 cuadrados y 1 triángulo. Si k = 7, en el área habrá 4 triángulos y 3 cuadrados, por lo que en este caso el nuevo punto será un triángulo.

Cabe destacar que, aunque su simplicidad y facilidad de uso lo convierten en una gran opción en problemas de clasificación, como la detección de spam o la clasificación de noticias,

es importante tener en cuenta la elección de k. En general, para garantizar la precisión del algoritmo KNN en la clasificación, la selección adecuada de k y la consideración de las características pertinentes son cruciales. Un uso de valores grandes de k en el algoritmo ayuda a reducir el efecto del ruido en la clasificación, pero también puede establecer límites entre clases similares. Como resultado, elegir el valor correcto es esencial para la precisión del modelo.

A continuación, se va a realizar un ejercicio práctico de ejemplo utilizando Jupyter Notebooks. Para ello, se ha decidido crear una base de datos con dos categorías: Azul y Rojo.

El objetivo del ejercicio será predecir de qué color será el nuevo punto que posteriormente se predecirá.

Creación de la base de datos

```
import pandas as pd
import numpy as np

# Se define las clases del conjunto de datos
Rojo = np.array([(8,4), (9,5), (9,7), (10,6), (10,8), (10,10), (11,5), (11,8), (12, 5), (12,6), (10,3), (8,9), (8,7), (8,8)])
Azul = np.array([(1,3), (2,5), (3,2), (4,1), (4,7), (3,8), (5,3), (6,5), (4,6), (7,8), (7,10), (6,7), (5,8), (5,6),])
```

Figura 9: Creación de la base de datos. Elaboración propia,

Una vez creados los datos, se visualizan en la Figura 10:

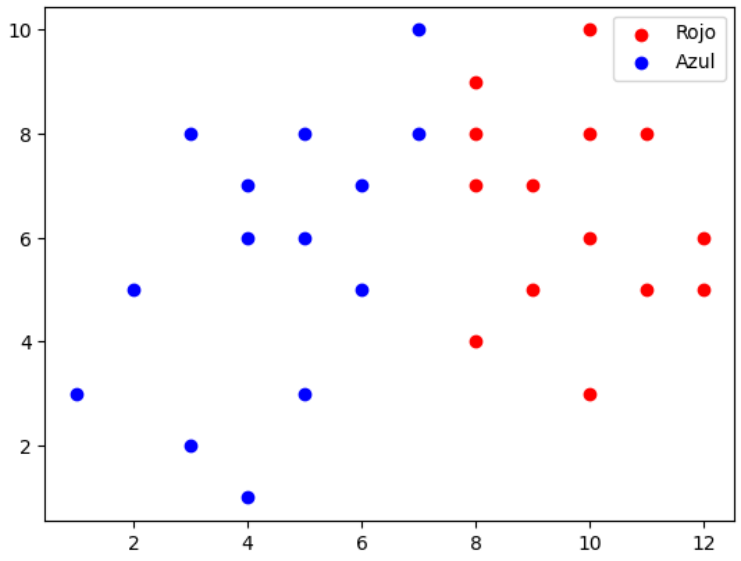

Figura 10: Gráfico de los datos generados. Elaboración propia.

Seguidamente se va a desarrollar el algoritmo. Para ello ha sido necesario que previamente se defina X (el cual contiene los datos de entrada) e y (el cual representa la etiqueta de color “Rojo” y “Azul).

Una vez concretada esta información, se procede a dividir los datos en conjunto de entrenamiento y de prueba. Siguiendo los mismos parámetros que en los ejercicios anteriores, el tamaño de la prueba será de 0’2, es decir, el 20% de los datos se utilizarán como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento (Figura 11).

Procesamiento de los datos

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

# Se combinan los datos
X = np.concatenate((Rojo, Azul))
y = np.array([0]*len(Rojo) + [1]*len(Azul))  # 0: Rojo, 1: Azul

# Se dividen los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

# Se crea el modelo KNN, se entrena el modelo
knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

# Prediciendo un nuevo punto (7,5)
new_point = np.array([[7,5]])
prediction = knn.predict(new_point)

print(f'Predicción con k=3: {"Rojo" if prediction[0] == 0 else "Azul"}')
```

```
Predicción con k=3: Rojo
```

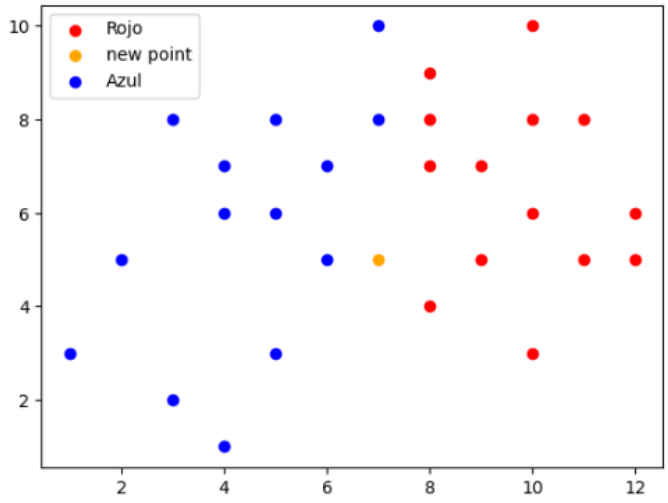

Figura 11: Procesamiento de los datos. Elaboración propia.

Así pues, se ha creado un nuevo punto para poner a prueba el algoritmo creado y verificar su eficacia. Este nuevo punto se ha representado en la figura anterior de color naranja, y se encuentra situado en las coordenadas [7,5] de la Figura 11. Con el fin de recalcar la importancia de la elección de k para el modelo, se ha llevado a cabo la predicción, en primer lugar, con k=3.

Tal y como se puede observar, el resultado que indica el algoritmo es que el nuevo punto pertenece a la categoría “Rojo”. El modelo ha podido llegar a esta conclusión debido a que, de los 3 puntos más cercanos, la mayoría son de este color (Figura 12):

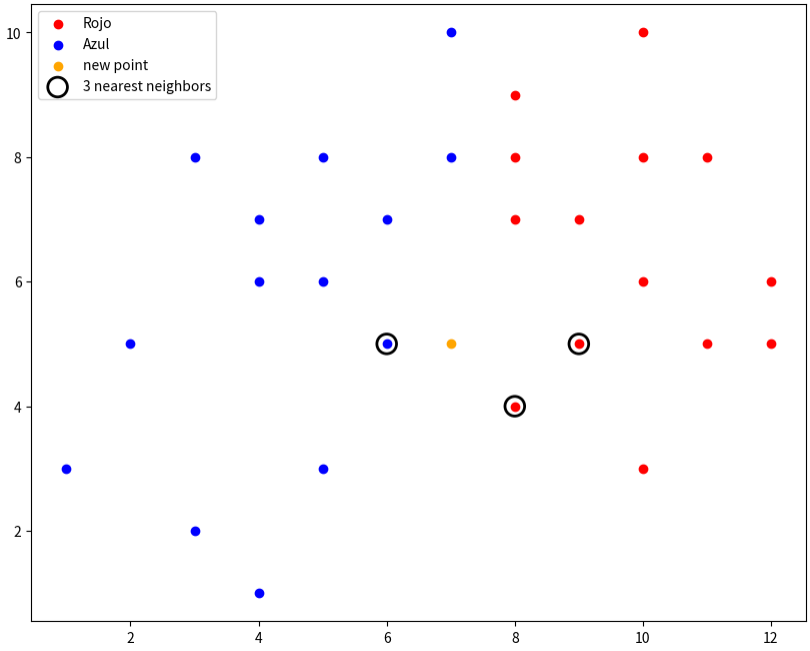

Figura 12: Indicación de los 3 vecinos más cercanos (k=3). Elaboración propia.

No obstante, si se procede a elevar el número de vecinos más cercanos a 9 (k=9), el resultado predicho por el algoritmo cambia a “Azul” (Figura 13):

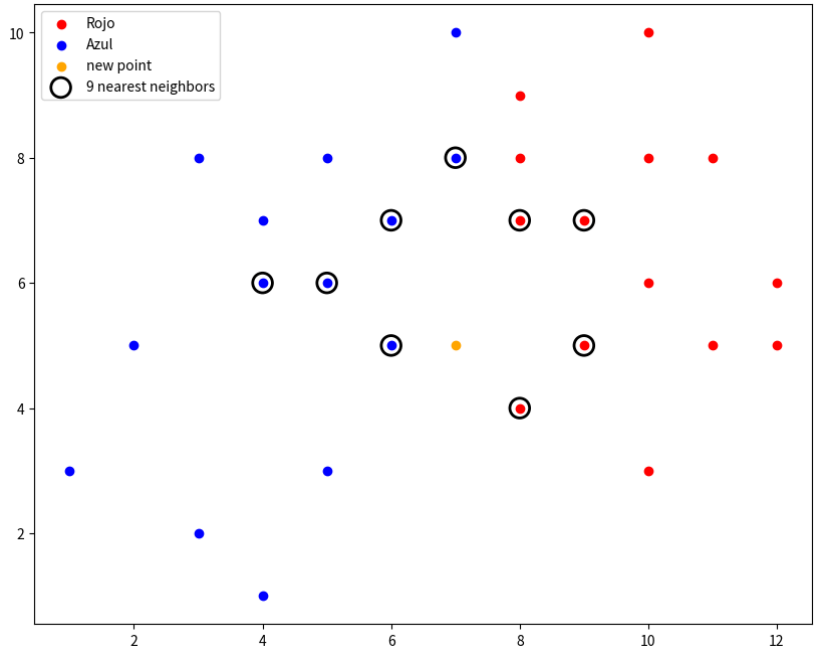

Figura 13: Indicación de los 9 vecinos más cercanos (k=9). Elaboración propia.

Por tanto, las conclusiones que se pueden llegar a obtener en este ejercicio han sido las siguientes:

- Los datos de “Rojo” y “Azul” están bastante separados, por lo que el espacio es fácilmente divisible en dos regiones.
- Podría ser suficiente tener un k pequeño para predecir un nuevo punto, ya que los vecinos más cercanos probablemente pertenecerán a la misma clase que el nuevo punto.
- En el caso de que el nuevo punto estuviese situado cerca del perímetro de separación o frontera entre las clases, un k pequeño podría dar lugar a una clasificación incorrecta. En este caso sería más conveniente aumentar k.

2.3.4. Árboles de decisión

Según Hastie et al. (2009), “un árbol de decisión es un modelo de clasificación que particiona el espacio de características en regiones rectangulares y asigna una etiqueta de clase a cada una de estas regiones”.

Su estructura de árbol se asemeja a un diagrama de flujo, donde cada nodo interno del árbol representa una característica o atributo, las ramas representan las reglas de decisión y cada nodo hoja representa un resultado. En la Figura 14 es posible observar la estructura de un árbol de decisión junto con sus componentes.

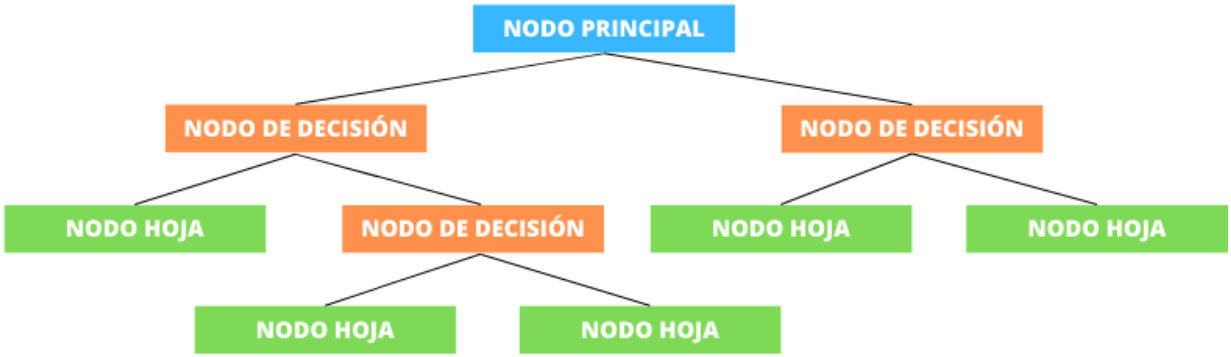

Figura 14: Estructura de un árbol de decisión. Elaboración propia.

Su objetivo en la clasificación es separar un conjunto de datos en diferentes categorías o grupos, seleccionando una serie de características o atributos importantes a través de la creación de reglas de decisión. Estas determinan qué atributos son los más relevantes y cómo se deben combinar para obtener una clasificación precisa.

Los árboles de decisión son uno de los algoritmos más utilizados por su sencillez de interpretación y comprensión. Su estructura de árbol sirve para proporcionar una representación visual clara de las decisiones y de los razonamientos utilizados, lo cual simplifica la comprensión

del proceso que hay detrás de una predicción. Además, son muy fáciles de construir y pueden utilizarse para resolver una amplia gama de problemas, lo que los convierte en una opción popular para muchas aplicaciones de aprendizaje automático.

Sin embargo, también presentan algunas limitaciones. Por ejemplo, pueden tener dificultades para manejar relaciones no lineales entre las variables de entrada y la variable objetivo. Además, pueden verse afectados por el sobreajuste[4] en conjuntos de datos pequeños o ruidosos, lo que puede resultar en una baja precisión en datos nuevos.

A continuación, se va a realizar un ejercicio para poner en práctica la teoría sobre el árbol de decisión. Para ello, también se ha utilizado un dataset ofrecido por la librería sklearn. Este conjunto de datos tiene información sobre las características de tres especies distintas de flores iris: Setosa, Versicolor y Virginica. El objetivo de este ejercicio será que, a través de la longitud y amplitud de los sépalos y pétalos, el algoritmo pueda predecir y clasificar correctamente las flores iris según sus características.

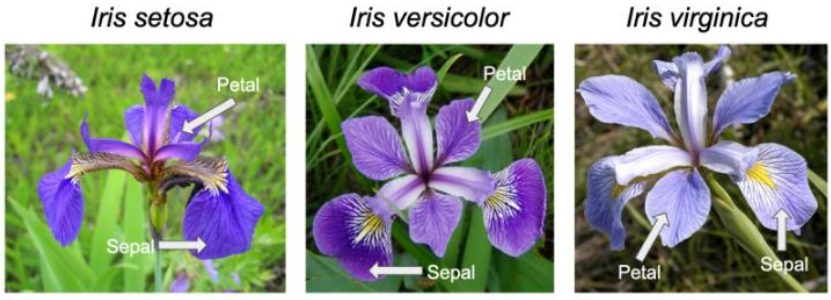

Ilustración 1: Tipos de flores iris del dataset. Fuente: Rpubs.com

[4] El sobreajuste se da cuando un modelo se ajusta demasiado a los datos de entrenamiento y pierde su capacidad de generalización a nuevos datos. Como resultado, el modelo puede tener una alta precisión en los datos de entrenamiento, pero una baja precisión en datos nuevos y no ser capaz de generalizar correctamente.

Continuando con los mismos pasos realizados en ejercicios anteriores, en primer lugar se ha importado la librería junto con el dataset y posteriormente se ha definido las variables “X” e “y” para así poder dar paso al desarrollo del algoritmo (Figura 15).

Importación de los datos

```
#Se carga el conjunto de datos

from sklearn.datasets import load_iris
iris = load_iris()
```

```
# Se definen las características (X) y la variable objetivo (Y)
X = iris.data
y = iris.target
```

Desarrollo del algoritmo

```
#Se separan los datos en entrenamiento y prueba
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

#Se crea el modelo y se entrena
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion="entropy")
clf.fit(X_train, y_train)
```

Figura 15: Importación de los datos y desarrollo del algoritmo. Elaboración propia.

Con el fin de entender mejor los datos, se procede a visualizarlos en forma de árbol (Figura 16):

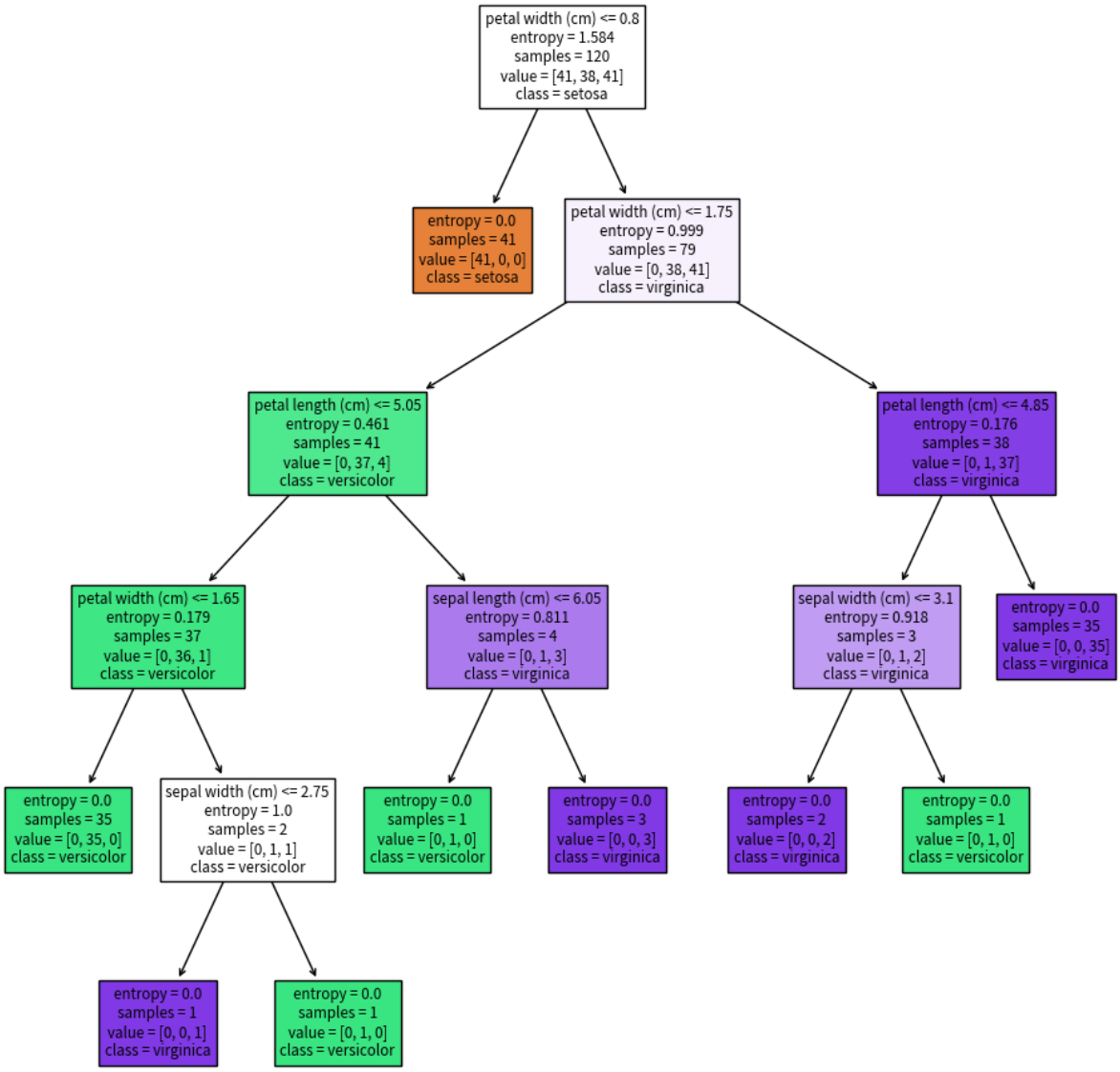

Figura 16: Árbol de los datos de "Iris". Elaboración propia.

Esta representación muestra de forma esquematizada las posibles soluciones existentes a una decisión basada en ciertas condiciones. Empieza con solo un bloque situado en la parte superior, conocido por “nodo raíz”, y se divide en posibles soluciones, cada una de las cuales lleva a otros nodos que se ramifican de la misma manera.

Los nodos de la parte superior del gráfico muestran unas características más generales y, a medida que se desciende por el árbol, las divisiones se vuelven más específicas. Cada nodo muestra:

- La condición que divide ese conjunto de datos [por ejemplo, "petal widht (cm) <= 0’8”]
- El valor de entropía. Es una medida de incertidumbre, un valor bajo significaría que el conjunto de datos es homogéneo, es decir, contiene más ejemplos de una clase específica.
- Número de muestras que cumplen con las condiciones hasta ese momento.
- Valores de la clase. Representa la cantidad de muestras de cada clase que se encuentran en el nodo. Por ejemplo, el primer nodo tiene el valor [41, 38, 41], significa que hay 41 muestras de la clase 0 (Setosa), 30 muestras de la clase 1 (Versicolor) y 41 muestras de la clase 2 (Virginica).
- La clase que se ha predicho, siendo esta la que tiene más muestras en ese nodo.

En el árbol que se ha representado, se evaluaría inicialmente el nodo raíz de la parte superior. En el caso de que la condición fuese verdadera, se seguiría la rama izquierda y; si fuese falsa, se tendría que seguir la rama derecha. De esta forma se llegaría hasta un nuevo nodo y se repetiría el proceso. Asimismo, para facilitar el entendimiento del árbol anterior, se ha relacionado el valor de entropía con el color de cada hoja. Esto quiere decir que cada clase está representada por un color: verde para Versicolor, morado para Virginica y naranja para Setosa. A menor entropía, mayor será la certeza de poder predecir la etiqueta de la flor, por lo que el color será más intenso. Los nodos blancos, por tanto, evidencia la incertidumbre.

La interpretación de las primeras hojas del árbol sería la siguiente: si la anchura del pétalo es menor o igual que 0’8 cm, entonces la flor iris pertenece a la variedad Setosa. En cambio, si

su longitud es mayor, habría que fijarse en la largura del pétalo, pudiendo ser Versicolor si esta es igual o inferior a 1’75cm; o Virginica si es mayor que esa cifra. De esta forma, se seguiría leyendo el árbol hasta finalizar en un nodo hoja.

2.3.5. Máquina de vectores de soporte (SVM)

La máquina de vectores de soporte (SVM, por sus siglas en inglés) es un modelo de aprendizaje supervisado que se utiliza para la clasificación y regresión de datos. Según sus autores, Vladimir Vapnik y Alexey Chervonenkis (1960), es un enfoque de aprendizaje que se basa en la teoría del aprendizaje estadístico y la teoría de la optimización.

La SVM es un método de clasificación que se utiliza para separar óptimamente dos categorías de datos mediante la construcción de un hiperplano en un espacio de alta dimensionalidad. Este hiperplano busca tener la máxima distancia o margen con los puntos más cercanos de ambas categorías, lo que le da el nombre de clasificador de margen máximo (Cortes y Vapnik, 1995). Originalmente, fue diseñado para resolver problemas de clasificación binaria, pero actualmente se ha adaptado para resolver problemas de regresión y clasificación multiclase en diferentes áreas como la visión artificial o el análisis de series temporales (Weston, J., 2003).

El objetivo principal de SVM es encontrar un hiperplano de separación que esté equidistante de los datos de entrenamiento más cercanos de cada categoría, con el fin de obtener el margen máximo a cada lado del hiperplano. Cabe destacar que solo se consideran los datos que se encuentran justo encima del margen máximo para definir dicho hiperplano. Estos puntos se llaman vectores soporte.

En la Figura 17, se ha creado un hiperplano con dos clases de objetos, cuadrados y triángulos:

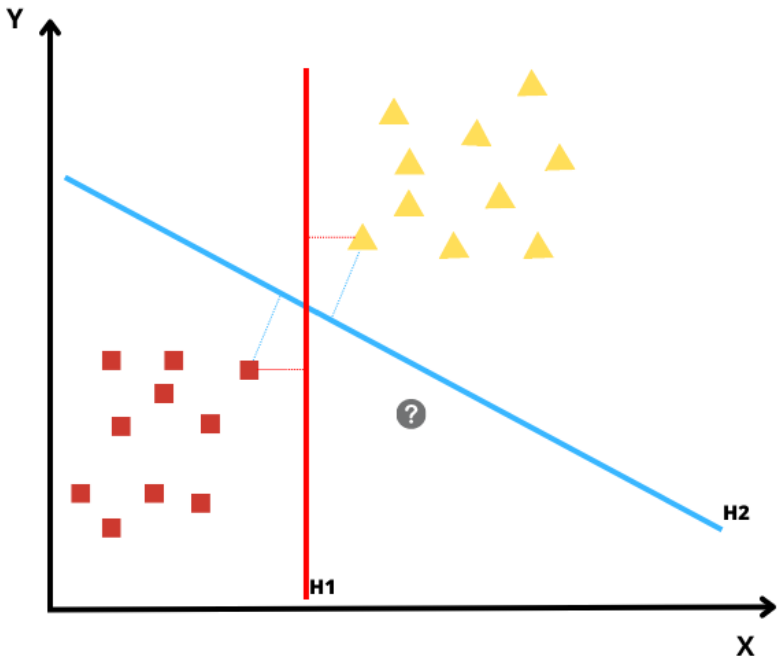

Figura 17: Hiperplano con dos posibles separaciones. Fuente: Elaboración propia.

El objetivo será encontrar una separación de margen equidistante con los vectores de soporte más cercanos, los cuales serán los más próximos a la línea de separación. En la imagen anterior, se ha creado dos posibles alternativas para separar el conjunto de datos. La primera de ellas es la línea roja llamada H1, la cual sería una forma errónea de separación debido a que el margen (línea discontinua roja) entre los vectores de soporte y la línea roja es inferior al de la línea azul. Si se intentase clasificar el nuevo punto, se identificaría como un triángulo ya que se encuentra situado en la parte derecha de la línea roja. En cambio, la forma óptima sería mediante la línea azul H2, ya que separa las dos clases con el mayor margen equidistante posible. En este caso, la categorización del nuevo punto sería de “cuadrado” al situarse en la parte izquierda de la línea de separación azul.

Cabe destacar que no siempre el hiperplano será lineal, sino que existen también otros casos más complejos. Para ello, se hace uso de una técnica llamada "kernel[5] trick" (Vapnik, V., 1998), la cual transforma el espacio de características original en uno de mayor dimensión. Por lo tanto, se pueden utilizar diferentes tipos de kernel, como el polinómico o el radial, para construir hiperplanos no lineales y así poder resolver problemas de clasificación y regresión más complejos (Cortes y Vapnik, 1995).

Con el fin de continuar explicando este método, se procederá a realizar un breve ejercicio práctico. Para este propósito, se creará una muestra de datos para posteriormente aplicar el algoritmo SVM y poder representarlo visualmente.

Creación de los datos

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Creación de un conjunto de datos con 2 características
X, y = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_classes=2, random_state=1)
```

Figura 18: Creación del conjunto de datos. Elaboración propia.

En la Figura 18 se ha generado un conjunto de datos de forma aleatoria con un total de 1000 muestras. Además, se ha especificado que cada una de estas tiene un total de 2 características y puede formar parte de alguna de las dos clases o categorías creadas. De esta forma, será más fácil poder visualizarlo correctamente en un gráfico 2D para explicar el algoritmo.

[5] Función que permite realizar transformaciones no lineales en los datos de entrada.

Desarrollo del algoritmo

```
from sklearn import svm

# División de los datos en un conjunto de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = svm.SVC(kernel='linear')

# Entrenamiento del modelo
clf.fit(X_train, y_train)

# Predicciones con los datos de prueba
y_pred = clf.predict(X_test)

# Reporte de clasificación
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.84      0.86        95
           1       0.86      0.89      0.87       105

    accuracy                           0.86       200
   macro avg       0.87      0.86      0.86       200
weighted avg       0.87      0.86      0.86       200
```

Figura 19: Desarrollo del algoritmo SVM. Elaboración propia.

Una vez definidas X e y, se procede a desarrollar la máquina de vectores de soporte y se realiza un reporte para confirmar la correcta su correcta ejecución. En las métricas obtenidas se puede observar que el modelo tiene una precisión total del 86%, es decir, de cada 100 datos puede clasificar correctamente 86. Esta cifra indica un buen rendimiento del modelo en este conjunto de datos particular.

En la Figura 20 se ha representado el modelo en una gráfica 2D:

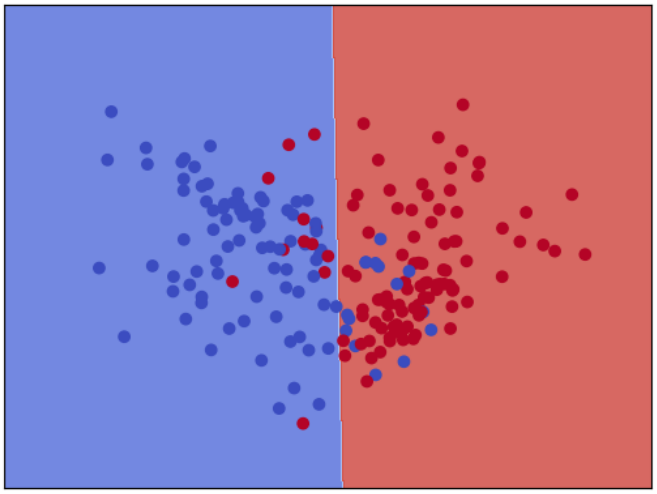

Figura 20: Visualización del hiperplano de decisión del SVM. Elaboración propia.

En la gráfica, los colores representan las diferentes clases existentes: rojo y azul. Por otro lado, la línea que separa los colores es el hiperplano de decisión del SVM. La razón por la que se observan puntos rojos en la zona azul (o viceversa) es que ningún clasificador es perfecto, y siempre habrá algún grado de error en la clasificación.

El SVM intenta maximizar el margen entre las dos clases, pero dependiendo de la complejidad de los datos y la presencia de ruido, es posible que no pueda separar perfectamente todas las instancias. Estos puntos son ejemplos de errores de clasificación. Por ejemplo, un punto rojo en la zona azul significa que esa instancia pertenece a la clase roja pero fue clasificada incorrectamente como la clase azul.

La tasa de estos errores se mide comúnmente a través de la precisión que, como se vio anteriormente, fue del 86%, es decir, existe un 14% de los datos que han sido clasificados de forma incorrecta.

2.3.6. Random Forest

El algoritmo de Random Forest fue propuesto por Leo Breiman en 2001. Según su autor, es "un conjunto de árboles de decisión aleatorios, cada uno construido a partir de una muestra aleatoria del conjunto de entrenamiento" (Breiman, 2001, p. 5).

Este método, también conocido como bosque aleatorio en castellano, es un modelo de aprendizaje automático que se utiliza para clasificar y predecir datos. Se basa en la creación de múltiples árboles de decisión, cada uno de los cuales es un modelo independiente y no correlacionado.

Cada árbol de decisión es entrenado en una muestra aleatoria de los datos y utiliza diferentes variables y reglas de división para tomar decisiones. Al combinar las decisiones de

todos los árboles de decisión, el modelo puede producir una predicción más precisa y estable que la de un solo árbol de decisión.

Los pasos del algoritmo se pueden resumir en los siguientes:

1. Seleccionar un conjunto aleatorio de observaciones del conjunto de entrenamiento.
2. Construir un árbol de decisión utilizando las observaciones seleccionadas aleatoriamente. En cada nodo, se selecciona aleatoriamente un subconjunto de características para ser consideradas para la división.
3. Repetir los pasos 1 y 2 para crear un conjunto de árboles de decisión.
4. Para realizar una predicción, se envía la observación a través de todos los árboles de decisión del conjunto y se toma la predicción final como la que tiene la mayoría de los votos (en el caso de la clasificación) o la media (en el caso de la regresión).

En la figura 21 se ha representado gráficamente un ejemplo del algoritmo para la clasificación de datos:

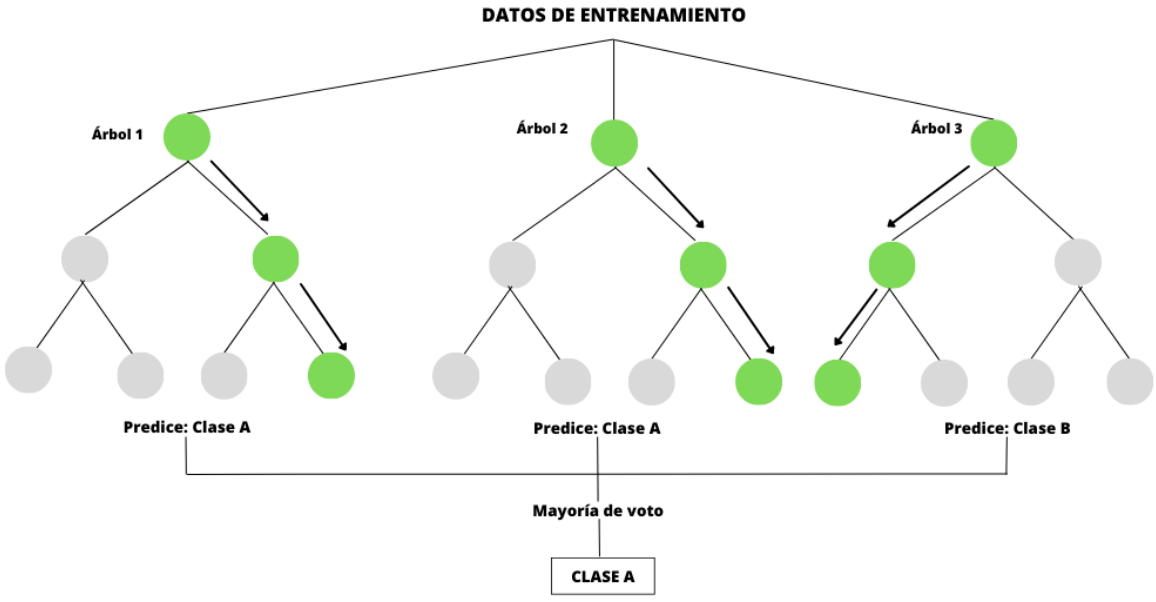

Figura 21: Ejemplo de Random Forest para la clasificación de datos. Elaboración propia.

Para este ejemplo, se ha supuesto que se tiene un conjunto de 3 árboles de decisión que conforman el bosque aleatorio. Cada árbol es entrenado con una muestra aleatoria de los datos

y utiliza diferentes variables y reglas de división para tomar decisiones. Esto significa que cada árbol puede tener una perspectiva diferente sobre los datos, lo que puede llevar a predicciones diferentes.

Generalmente, la predicción de todos los árboles nunca será la misma debido a que es posible que los datos sean complejos y difíciles de clasificar, o a que hay ciertas características o patrones en los datos que son difíciles de capturar por todos los árboles (Breiman, 1996). Sin embargo, como se toma en cuenta la decisión de todos los árboles del bosque, la predicción final es más precisa y estable que la de un solo árbol de decisión. En este caso, dos de los árboles han predicho la clase A y 1 la clase B, por tanto, por mayoría, el resultado final del modelo será "Clase A".

Es importante destacar que el número de árboles en el bosque aleatorio es un hiperparámetro que se puede ajustar para mejorar la precisión del modelo (Liaw, A., & Wiener, M., 2002). En general, cuanto mayor sea el número de árboles, mayor será la precisión del modelo, pero también aumentará el tiempo de entrenamiento y la complejidad del modelo (Cutler et al. (2007). Asimismo, este algoritmo es una mejora sobre los árboles de decisión tradicionales debido a su capacidad para reducir el sobreajuste, mejorar la estabilidad, aumentar la precisión y utilizar la selección aleatoria de características.

A diferencia de los árboles de decisión, recrear un ejemplo visual de los bosques aleatorios es mucho más complejo debido a la gran cantidad de árboles existentes en un conjunto. Por este motivo, en vez de representar visualmente el modelo, se realizará una comparación de los resultados de un mismo dataset entre el algoritmo de Árboles de Decisión y el Random Forest.

Los datos escogidos para este ejercicio serán los mismos que para el ejercicio de los árboles simples: las flores iris.

Como el proceso de importación de datos y creación del algoritmo ya se hizo con anterioridad, se procederá a mostrar los resultados de los algoritmos en las Figuras 22 y 23.

```
Resultados del modelo de  DecisionTreeClassifier(criterion='entropy')
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       0.86      0.86      0.86         7
           2       0.93      0.93      0.93        14

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30

matriz de confusión:
[[ 9  0  0]
 [ 0  6  1]
 [ 0  1 13]]
```

Figura 22: Resultados del modelo de Árbol de Decisión. Elaboración propia.

```
Resultados del modelo de  RandomForestClassifier()
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       0.88      1.00      0.93         7
           2       1.00      0.93      0.96        14

    accuracy                           0.97        30
   macro avg       0.96      0.98      0.97        30
weighted avg       0.97      0.97      0.97        30

matriz de confusión:
[[ 9  0  0]
 [ 0  7  0]
 [ 0  1 13]]
```

Figura 23: Resultados del modelo de Random Forest. Elaboración propia.

Las Figuras 22 y 23 muestran el resultado obtenido por los algoritmos creados con el fin de poder predecir la etiqueta de cada flor: Setosa (0), Versicolor (1) y Virginica (2). Los modelos creados escogieron un total de 30 muestras del conjunto de datos, perteneciendo 9 a la etiqueta Setosa, 7 a Versicolor y 14 a Virginica.

En el caso del árbol de decisión, todas las métricas estudiadas obtuvieron la máxima puntuación en la predicción de las flores Setosa, es decir, se clasificó correctamente toda la muestra. En cuanto al resto de etiquetas, las métricas obtenidas son también valores muy altos,

sin embargo, no llegan a los máximos parámetros. Esto ha sido debido a que, de los 7 casos que eran originalmente Versicolor, el modelo predijo correctamente 6 y erróneamente clasificó 1 como Virginica. Además, de los 14 datos que eran Virginica, el modelo predijo correctamente 13 y clasificó 1 de forma errónea como Versicolor.

Centrando ahora la atención en los resultados del modelo de Random Forest, estos tienen también una elevada precisión. Se pudo clasificar correctamente todas las muestras escogidas de Setosa y de Versicolor, sin embargo, al igual que en el Árbol de Decisión, se predijo de forma incorrecta una flor como Versicolor; siendo en realidad una Virginica.

Aunque, debido a la sencillez de la base de datos ambos modelos han obtenido una precisión muy alta, si se realiza una comparación entre estos, el algoritmo de Bosques Aleatorios mejora la precisión total del modelo en 4 puntos porcentuales (del 93% al 97%). La explicación de este suceso se basa en que este es un conjunto de árboles aleatorios, por lo tanto, el riesgo de sobreajuste (overlifting) es menor ya que disminuye la probabilidad de sobreajustar el modelo a los datos de entrenamiento, mejorando de esta forma su capacidad de generalización y ganando más estabilidad a la hora de enfrentarse a nuevos datos.

3. Aplicación práctica: caso real

Con el fin de poner en práctica toda la información proporcionada y explicada con anterioridad, en este capítulo se procederá a realizar un caso práctico. En los posteriores puntos se indicará cuál ha sido el dataset escogido, se explicará cómo ha sido el proceso del análisis exploratorio y se expondrá el desarrollo de los algoritmos junto con las conclusiones obtenidas tras el entrenamiento de los modelos.

Para poder realizar con éxito el proceso de la construcción de un sistema de machine learning, se ha seguido la siguiente guía (Ilustración 2):

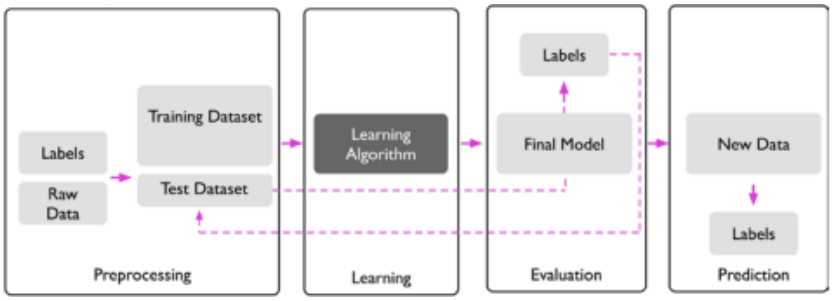

Ilustración 2: Roadmap de un modelo predictivo de machine learning. Fuente: Raschka, S., & Mirjalili, V. (2017), Python Machine Learning

En primer lugar, el preprocesamiento de los datos. Este paso incluye definir el problema, definir las características de entrada y la variable objetivo y preparar los datos para el modelo, es decir, realizar una limpieza, manejar datos restantes y transformar variables. Una vez realizado esto, se podrá dividir y entrenar los datos

En segundo lugar, se escogen los algoritmos de machine learning a utilizar, en este caso han sido de un total de 6 modelos: Naive Bayes, Regresión Logística, K-vecinos más cercanos, árboles de decisión, máquina de vectores de soporte y Random Forest.

En tercer lugar, se evalúa el rendimiento de los modelos mediante métricas. En el caso de no ser satisfactorio, se realizan mejores iterativas en los pasos anteriores con el fin de obtener rendimientos más elevados.

Por último, una vez aprobada la evaluación, se realiza predicciones con nuevos datos nunca vistos antes para posteriormente analizar sus resultados.

3.1. Introducción

El conjunto de datos con el que se trabajará a continuación se centrará en la obesidad que sufren los habitantes residentes en México, Perú y Colombia.

Este problema médico es una de las enfermedades más extendidas en la población. Según los datos ofrecidos por un estudio de la OMS en 2022, más del 52% de la población adulta mundial padece de sobrepeso u obesidad, es decir, aproximadamente 3800 millones de personas. En el caso de los jóvenes de hasta 21 años, esta cantidad se reduce a 340 millones. Además, a partir de la recopilación de datos históricos desde 1975, la obesidad se ha casi triplicado en todo el mundo.

Esta información da lugar a un futuro con esta enfermedad presente en todas partes del planeta. No obstante, mediante el análisis de múltiples casos es posible identificar cuáles son las variables que acentúan la aparición de la obesidad y, de esta forma, intentar prevenirla.

A continuación, se presentará un análisis de datos para la estimación de la obesidad en jóvenes de entre 16 y 21 años de México, Perú y Colombia.

3.2. Base de datos

El conjunto de datos ha sido obtenido a través de Science Direct[6], con un dataset original, cuenta con 2111 individuos de entre 16 y 61 años. No obstante, como el objetivo del presente TFG será entender, explicar y poder predecir la obesidad de los adolescentes, se ha reducido el tamaño de la muestra a aquellos sujetos entre 16 y 21 años.

Asimismo, para llegar con éxito a la meta propuesta, se analizará cuáles son las variables más influyentes en la obesidad y con ello se podrá identificar el nivel o tipo que posee un individuo.

3.3. Análisis exploratorio

El dataset está creado a través de una serie de preguntas recogidas mediante una encuesta. Para trabajar con más comodidad con los datos, se han convertido todas las variables categóricas a numéricas. En la Tabla 2 se indica las preguntas realizadas, las posibles respuestas y, entre paréntesis si procede, su codificación a número:

Variable	Pregunta	Respuesta
NUM_Gender	Sex	Male (1) Female (0)
Age	Age	
Height	Height	
Weight	Weight	
NUM_family_history_with_overweight	Has a family member suffered or suffers from overweight?	Yes (1) No (0)
NUM_FAVC	Frequent consumption of high caloric food	Yes (1) No (0)
NUM_FCVC	Frequency of consumption of vegetables	Never (0) Sometimes (1) Always (2)
NCP	Number of main meals	1 2 3 4

[6] Palechor, F. M., & De La Hoz Manotas, A. K. (2019b). Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Data in Brief, 25, 104344. https://doi.org/10.1016/j.dib.2019.104344

NUM_CAEC	Consumption of food between meals	No (0) Sometimes(1) Frequently(2) Always (3)
NUM_SMOKE	Do you smoke?	Yes (1) No (0)
NUM_CH2O	Consumption of water daily	Less than a litre (0) Between 1 and 2 l(1) More than 2 l (2)
NUM_SCC	Calories consumption monitoring	Yes (1) No (0)
NUM_FAF	Physical activity frequency	0 (0) 1 to 2 (1,2) 2 to 4 (2,4) 4 to 5 (4,5)
NUM_TUE	Time using technology devices	0 to 2 (0,2) 3 to 5 (3,5) >5 (5)
NUM_CALC	Consumption of alcohol	No (0) Sometimes(1) Frequently(2) Always (3)
NUM_MTRANS	Transportation used	Automobile (4) Motorbike (3), Bike (2) Public_transportation (1) Walking (0)
NUM_Nobeyesdad	Type of obesity[7]	Insufficient_weight (0) Normal_weight (1) Overweight_level_i (2) Overweight_level_ii (3) Obesity_type_i (4) Obesity_type_ii(5) Obesity_type_iii (6)
BMI[8]	Body mass index	

Tabla 3: Encuesta realizada a los individuos. Elaboración propia.

Para iniciar con el análisis de los datos, se ha cargado los datos en la hoja de Jupyter con la que se trabajará hasta el final del trabajo (Figura 24).

[7] Se utilizará la clasificación realizada por la Sociedad Española de Estudio de la Obesidad (SEEDO), sin incluir la obesidad tipo IV debido a su poca frecuencia.

[8] Esta información se calculará mediante la división del peso entre la altura elevada al cuadrado.

Importación de los datos

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Se cargan los datos
dataset = pd.read_excel('ObesityDataSet_NUM_CLEANED_1421_DEFINITIVO.xlsx')
df = dataset.drop('id', axis = 1)

# Se define la variable objetivo
features = df.drop(['NUM_Nobeyesdad'], axis=1)
target = df['NUM_Nobeyesdad']

# Mostrar las primeras filas del DataFrame ordenado
df.tail(216)
print ("Información en el dataset):")
print(df.keys())
print("Número total de individuos:", + len(df))
```

```
Información en el dataset):
Index(['NUM_Gender', 'Age', 'Height', 'Weight',
       'NUM_family_history_with_overweight', 'NUM_FAVC', 'NUM_FCVC', 'NCP',
       'NUM_CAEC', 'NUM_SMOKE', 'NUM_CH2O', 'NUM_SCC', 'NUM_FAF', 'NUM_TUE',
       'NUM_CALC', 'NUM_MTRANS', 'NUM_Nobeyesdad', 'BMI'],
      dtype='object')
Número total de individuos: 858
```

Figura 24: Importación de los datos. Elaboración propia.

Con el fin de entender la base de datos, en la Figura 24 se ha reflejado las variables que compone el dataset. Está compuesto por un total del 858 individuo, donde cada fila representa un sujeto y cada columna representa una característica específica del paciente. La variable objetivo será “NUM_Nobeyesdad”, es decir, el tipo de peso de cada individuo.

Antes de iniciar con el análisis de los datos, se comprobará si existen datos erróneos o faltantes (Figura 25). Se ha podido comprobar que el conjunto de datos contaba con individuos que posiblemente se creasen por equivocación ya que solamente había información recogida de la variable del historial familiar con obesidad (NUM_family_history_with_overweight). Por tanto, se procede a la eliminación de estos datos.

¿Datos faltantes?

```
# Encuentra las filas con valores NaN en la variable objetivo
nan_rows = df[df['NUM_Nobeyesdad'].isna()]

# Guarda las filas eliminadas en un nuevo DataFrame
rows_eliminated = nan_rows.copy()

# Elimina las filas con valores NaN en la variable objetivo
df_cleaned = df.dropna(subset=['NUM_Nobeyesdad'])

# Muestra las filas eliminadas
print("Las siguientes filas se eliminaron debido a valores NaN en la variable objetivo:")
print(rows_eliminated)
```

```
Las siguientes filas se eliminaron debido a valores NaN en la variable objetivo:
     NUM_Gender  Age  Height  Weight  NUM_family_history_with_overweight  \
856         NaN  NaN     NaN     NaN                                   1
857         NaN  NaN     NaN     NaN                                   1

     NUM_FAVC  NUM_FCVC  NCP  NUM_CAEC  NUM_SMOKE  NUM_CH2O  NUM_SCC  NUM_FAF  \
856       NaN       NaN  NaN       NaN        NaN       NaN      NaN      NaN
857       NaN       NaN  NaN       NaN        NaN       NaN      NaN      NaN

     NUM_TUE  NUM_CALC  NUM_MTRANS  NUM_Nobeyesdad  BMI
856      NaN       NaN         NaN             NaN  NaN
857      NaN       NaN         NaN             NaN  NaN
```

Figura 25: Eliminación de los datos faltantes. Elaboración propia.

3.3.1. Estadística descriptiva.

Una vez eliminado con éxito los valores faltantes, se procede al análisis de las estadísticas descriptivas. Esta información ayudará a entender la tendencia central, la dispersión y la forma de distribución de los datos.

Este análisis se ha iniciado mediante una búsqueda de posibles outliers, es decir, de observaciones anormales que se desvían significativamente del resto. La herramienta usada para esta indagación ha sido un gráfico de cajas y bigotes, también conocido como Boxplot.

De entre todas las variables de la base de datos, aquella que daba valores atípicos fue ‘BMI’ (body mass index o índice de masa corporal). Tal y como se puede observar en la Figura 26, existen 3 valores anómalos que podrían estar distorsionando las métricas del conjunto de datos.

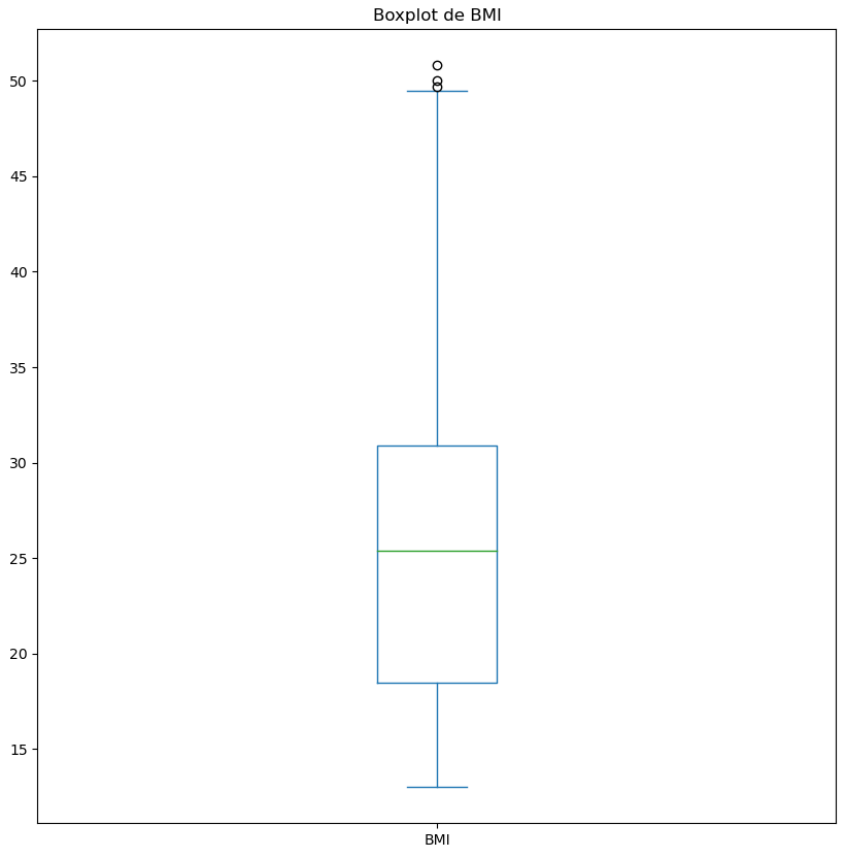

Figura 26: Box Plot de BMI. Elaboración propia.

A pesar de que se encuentran cerca del bigote superior, estos tres puntos están sesgando la media ya que un IMC excesivamente alto está relacionado también con un peso muy elevado debido a que el índice de masa corporal utiliza esa variable en su cálculo. Asimismo, estos sujetos tienen un IMC muy cercano a 50, lo cual indicarían que se encontrarían dentro de la categoría de obesidad extrema. Esta patología implica un grave problema para la salud de estas personas y su riesgo de fallecer es muy elevado ya que sus funciones básicas como respirar o caminar son interferidas. Además, son más susceptibles de padecer algunas otras enfermedades médicas como diabetes, hipertensión, ataques cerebrales u osteoartritis[9].

[9] Efectos del sobrepeso y la obesidad. (2022, 7 julio). Centers for Disease Control and Prevention. https://www.cdc.gov/healthyweight/spanish/effects.html

Es por ello que, sabiendo además que menos del 1% de los peruanos[10], mexicanos[11] y colombianos[12] sufren de obesidad extrema, se ha decidido eliminar estos 3 individuos.

Al tener ahora los datos limpios y sin influencias significativas, se procede a verificar el balance de las clases de la variable objetivo. En la Figura 27 se ha representado un gráfico de barras con la cantidad de personas que hay según su IMC.

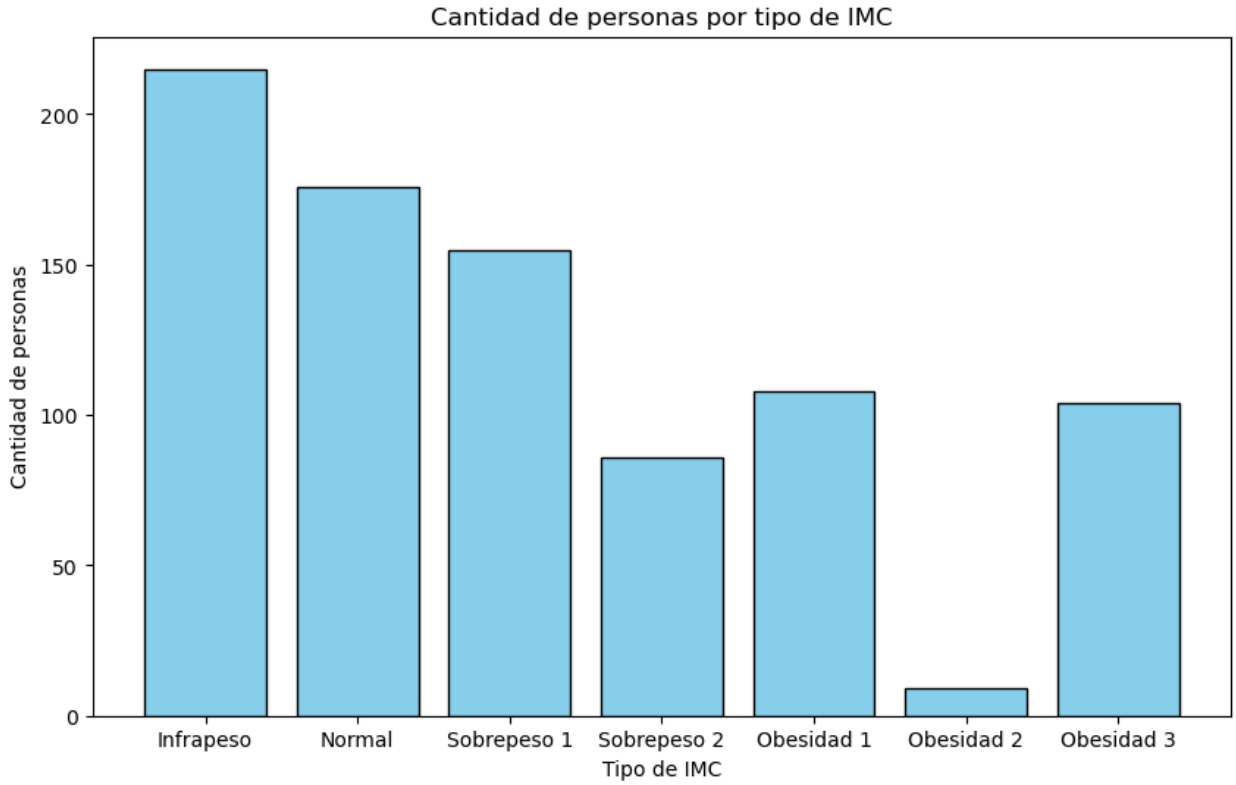

Figura 27: Cantidad de pacientes con obesidad. Elaboración propia.

[10] Ramírez, J. J. (2017, 17 julio). La obesidad en el Perú. Anales de la Facultad de medicina. https://doi.org/10.15381/anales.v78i2.13214

[11] Lugo, G. (2021, 17 noviembre). Obesidad, epidemia agudizada en México. Gaceta UNAM. https://www.gaceta.unam.mx/obesidad-epidemia-agudizada-en-mexico/

[12] Obesidad, un factor de riesgo en el covid-19. (2021). Ministerio de Salud y Protección Social - República de Colombia. https://www.minsalud.gov.co/Paginas/Obesidad-un-factor-de-riesgo-en-el-covid-19.aspx

Se puede observar que la distribución de las clases no es uniforme. El tipo de obesidad 2 tiene significativamente menos instancias que el resto, lo cual puede afectar de manera negativa al rendimiento de los modelos para predecir la obesidad.

Por lo tanto, podría ser beneficioso balancear los datos. Para ello se podrían aplicar diversos métodos:

- Sobremuestreo de las clases minoritarias: implica duplicar aleatoriamente instancias de la menor clase hasta que tenga la misma cantidad que la clase con más instancias.
- Submuestreo de las clases mayoritarias: implicar eliminar de forma aleatoria instancias de las clases con mayores datos hasta igualarlos a la misma cantidad que la clase minoritaria.
- Generación de instancias sintéticas: implica generar instancias artificiales de la clase minoritaria.

Elegir el método exacto depende de cada situación. Para este escenario, prevalecería el sobremuestreo o la generación de instancias sintéticas ya que, si se realizase el submuestreo u undersampling, el modelo perdería mucha información importante al tener que eliminar tantas instancias. No obstante, los otros dos métodos también tienen sus inconvenientes: el sobremuestreo u oversampling implica la creación de datos duplicados en el conjunto de entrenamiento y la generación de instancias sintéticas podría no representar con precisión la variabilidad real de la clase al tener que crear datos ficticios.

Debido a la posibilidad de introducir estos sesgos, se ha decidido continuar sin el balanceo de datos para observar los resultados de los modelos. En caso de que fuera necesario mejorarlos, se procederá a balancearlos.

3.3.2. Análisis multivariable

Debido a la gran cantidad de variables, se realizará en primer lugar un mapa de calor de la matriz de correlación con el fin de conocer cuáles son las variables más importantes y, tras esto, hacer un análisis bivariable y univariable de las más relevantes. La matriz es una herramienta de análisis multivariado que permite visualizar y comparar las correlaciones entre todas las parejas de variables en los datos al mismo tiempo (Figura 28).

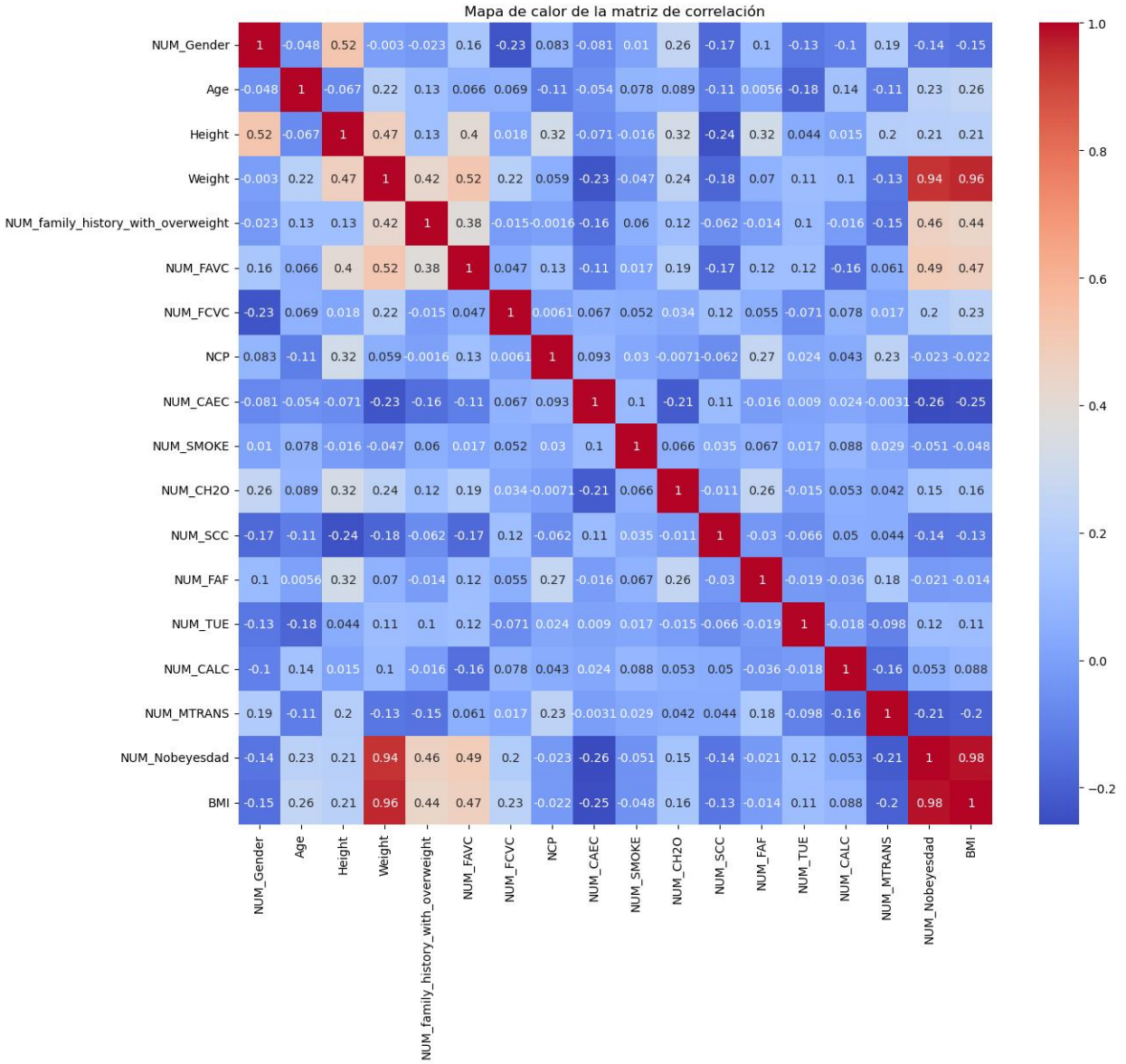

Figura 28: Mapa de calor de la matriz de correlación. Elaboración propia.

Cada celda de la matriz corresponde a la magnitud de la correlación entre dos variables. Los colores más cálidos, como el rojo, indican la presencia de una relación positiva fuerte; mientras que los colores más fríos, como el azul, indican una correlación negativa fuerte.

Las magnitudes están representadas mediante colores pero también de forma numérica, variando de -1 a 1. Un valor cercano a 1 correspondería con un color rojo muy notorio, lo cual significaría que la relación entre las variables es muy alta y, si una de ellas aumenta, la otra también tiende a hacer lo mismo. Además, la correlación entre una variable y ella misma es siempre 1, por eso la diagonal principal está llena de 1s. Por otra parte, en caso de tener una correlación cercana a -1 (color azul gélido), indicaría que si una variable aumenta, la otra tendería a disminuir. Como punto medio se sitúa el valor 0, es decir, no existe una correlación significativa entre ambas variables.

En la Figura 28 se pueden observar las siguientes relaciones:

- BMI y NUM_Obesyesdad (0’98): Tiene una magnitud prácticamente perfecta debido a que el tipo de obesidad se clasifica según el índice de masa corporal del individuo.
- BMI y Weight (0’96): El peso es usado para el cálculo del IMC o BMI ($\frac{PESO}{ALTURA^2}$) de forma directamente proporcional puesto que a medida que aumenta el peso, el índice de masa corporal también aumenta. Posteriormente se analizarán con más profundidad estas variables.
- Weight y NUM_FAVC (0’52): El consumo frecuente de alimentos altos en calorías tiene una correlación moderada con el peso y también con el tipo de IMC, por lo que podría indicar que el consumo habitual de estos alimentos puede contribuir a padecer obesidad.

- NUM_family_history_with_overweight y NUM_Obesyesdad (0’49): Parece ser que existe una correlación moderada positiva que sugiere que un historial familiar con sobrepeso favorezca a tener esta enfermedad.
- Height y Weight (0’47): Como es de esperar, mantienen una relación moderada positiva debido a que, de forma general, a mayor altura; mayor peso tendrá el sujeto.

3.3.3. Análisis bivariable

Una vez entendidas las principales correlaciones existentes, se procede a analizar con más profundidad las variables del conjunto de datos.

Algunas de las más importantes son el peso debido a su uso para el cálculo del índice de masa corporal. En las Figuras 29 y 30 se observa la relación existente entre el peso y la altura con el IMC.

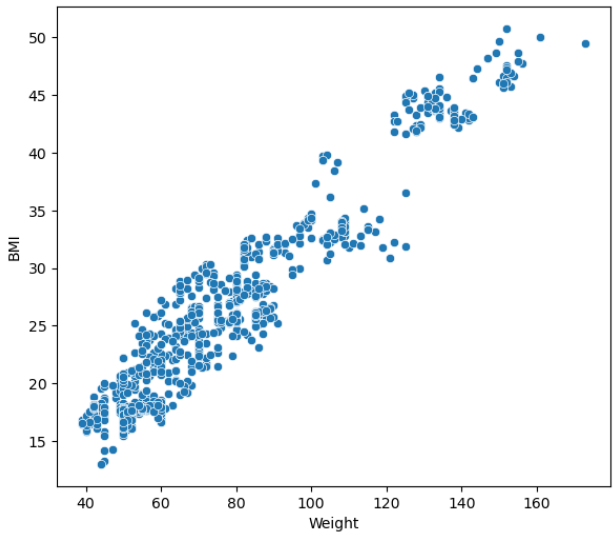

Figura 29: Relación del BMI y Weight. Elaboración propia.

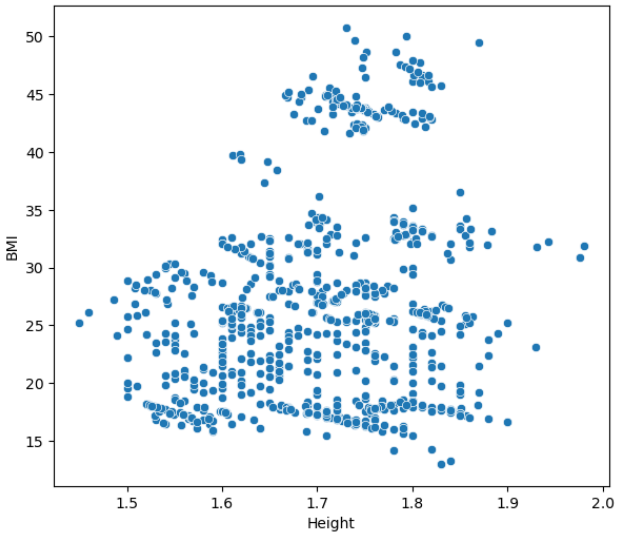

Figura 30: Relación del BMI y el Height. Elaboración propia.

Como ya se ha comentado con anterioridad, se puede observar una correlación positiva entre el índice de masa corporal y el peso. Por tanto, a medida que aumenta el peso de un individuo, también aumenta su BMI.

Sin embargo, este suceso no ocurre con la altura debido a que esta es inversamente proporcional al cuadrado del BMI, es decir, el índice de masa corporal disminuye al cuadrado de la altura. Por tanto, debido a la fórmula ($\frac{PESO}{ALTURA^2}$), la relación entre Height y BMI es más compleja y no se espera necesariamente una correlación fuerte entre estas variables, especialmente en un conjunto de datos con una amplia variedad de alturas y pesos. Además, una mayor altura puede permitir un mayor rango de pesos saludables dentro de un BMI determinado ya que esta variable no determina el índice de masa corporal.

No obstante, la relación entre el BMI, peso y altura puede variar también de factores individuales, como la composición corporal y la distribución de grasa. Todas estas variables no son tenidas en cuenta a la hora de calcular el índice de masa corporal por lo que pueden existir casos en los que un BMI alto no indique sobrepeso u obesidad. Esta casuística suele darse en atletas y deportistas federados que compiten de forma profesional.

Con tal de profundizar más en factores genéticos que puedan afectar a padecer de obesidad, en la Figura 31 se observa la cantidad de personas que, con un historial positivo de obesidad en algún individuo de su familia, tiene actualmente sobrepeso/obesidad, un peso normal o un peso bajo.

Cabe recordar que para determinar la clasificación del peso se ha usado la siguiente clasificación según el SEEDO (Sociedad Española para el Estudio de la Obesidad):

IMC	Clase
< 18’5	Peso Bajo
18’5 – 24’9	Peso Normal
25 – 26’9	Sobrepeso grado I
27 – 29’9	Sobrepeso grado II
30 – 34’9	Obesidad tipo I
35 – 39’9	Obesidad tipo II
40 – 49’9	Obesidad tipo III (mórbida)
50 <	Obesidad tipo IV (extrema)

Tabla 4: Clasificación del IMC. Fuente: SEEDO - SOCIEDAD ESPAÑOLA DE OBESIDAD

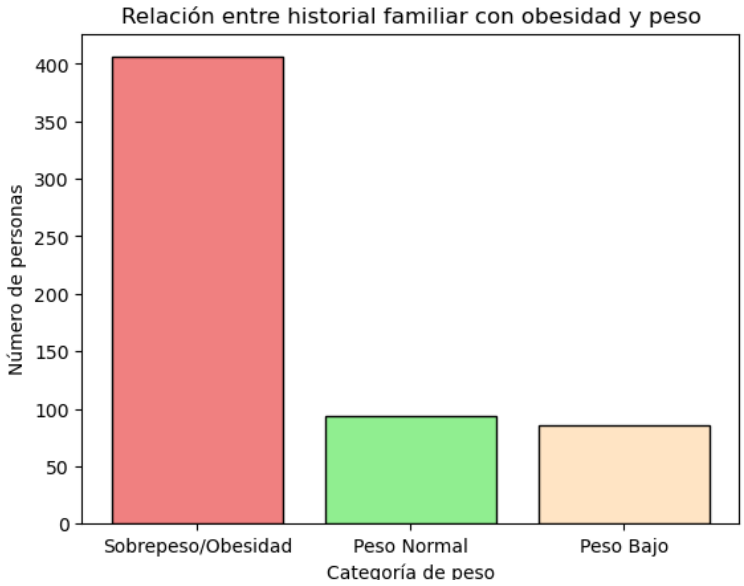

Figura 31: Relación entre un historial familiar con obesidad y el peso. Elaboración propia.

Las cantidades pertenecientes a cada colectivo son:

- Número de personas con sobrepeso/obesidad y un historial familiar de obesidad positivo: 403.
- Número de personas con peso normal y un historial familiar de obesidad positivo: 94.
- Número de personas con infrapeso y un historial familiar de obesidad positivo: 86.

Según los datos analizados, el 68% de los individuos encuestados tienen a una persona en su familia con obesidad, es decir, este porcentaje hace referencia a 583 personas de 855. Además, el 69% de los sujetos con antecedentes presentan sobrepeso u obesidad. Por lo tanto, se podría concluir que el historial familiar puede ser un factor importante en el desarrollo de esta enfermedad.

3.4. Desarrollo y resultados de los algoritmos

Con el fin de aplicar lo explicado anteriormente en el marco teórico, se procederá a crear varios modelos de clasificación. El propósito será predecir si un sujeto tendrá o no obesidad y, en caso afirmativo, a qué grupo pertenecería.

Para ello, se ha definido como objetivo el tipo de obesidad (NUM_Nobesyesdad) y el resto de las variables serán las características.

Seguidamente, se utiliza la función train_test_split para dividir los datos en conjuntos de entrenamiento y de prueba. En este caso, se establece en 0’2, lo que significa que el 20% de los datos se utilizarán como conjunto de prueba, mientras que el 80% restante se utilizará como conjunto de entrenamiento.

Cuando se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split, se separan tanto los datos de características (features) como las etiquetas (target). features_train y features_test contienen las características de entrenamiento y prueba, respectivamente, mientras que target_train y target_test contienen las etiquetas correspondientes a los conjuntos de entrenamiento y prueba, respectivamente.

A continuación, se crean los modelos a emplear: Árbol de decisión, Naive Bayes, Regresión logística, Random Forest, Sistema de máquina de vectores y K vecinos más cercanos. En la Figura 32 se puede observar el proceso de entrenamiento y evaluación de los modelos.

```
# Definir las características y y la variable objetivo
features = df_cleaned_outliers.drop(['NUM_Nobeyesdad'], axis=1)
target = df_cleaned_outliers['NUM_Nobeyesdad']

features_train, features_test, target_train, target_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Definir los modelos
models = [
    ('Árbol de decisión', DecisionTreeClassifier()),
    ('Naive Bayes', GaussianNB()),
    ('Regresión Logística', LogisticRegression()),
    ('Random Forest', RandomForestClassifier()),
    ('SVM', SVC()),
    ('KNN', KNeighborsClassifier())
]

# Crear un DataFrame para almacenar los resultados
results = pd.DataFrame(columns=['Modelo', 'R2_train', 'R2_test', 'Accuracy_train', 'Accuracy_test'])

# Entrenar y evaluar cada modelo
for name, model in models:
    # Entrenar el modelo
    model.fit(features_train, target_train)

    # Hacer predicciones en el conjunto de entrenamiento y de prueba
    target_pred_train = model.predict(features_train)
    target_pred_test = model.predict(features_test)

    # Mostrar los resultados del modelo
    print("Resultados del modelo de ", name)
    print(classification_report(target_test, target_pred_test))

    # Mostrar la matriz de confusión
    print("Matriz de confusión:")
    print(confusion_matrix(target_test, target_pred_test))

    # Calcular R2 y precisión para el conjunto de entrenamiento y de prueba
    r2_train = r2_score(target_train, target_pred_train)
    r2_test = r2_score(target_test, target_pred_test)
    accuracy_train = accuracy_score(target_train, target_pred_train)
    accuracy_test = accuracy_score(target_test, target_pred_test)

    # Almacenar los resultados
    results = results.append({'Modelo': name, 'R2_train': r2_train, 'R2_test': r2_test,
                              'Accuracy_train': accuracy_train, 'Accuracy_test': accuracy_test},
                             ignore_index=True)

# Mostrar los resultados
print(results)
```

Figura 32: Desarrollo de los algoritmos. Elaboración propia.

Tras el desarrollo de todos los algoritmos, se ha procedido a calcular las siguientes métricas de rendimiento y una matriz de confusión para evaluar la precisión y eficacia en la clasificación de los diferentes tipos de obesidad:

- Precisión: Indica cuántas de las predicciones positivas del modelo son realmente correctas. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos.
- Recall: Mide la proporción de los verdaderos positivos que el modelo es capaz de detectar

- F1-score: Es una medida que combina la precisión y el recall en una sola métrica.
- R^2-score: Indica la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes. Un valor de R^2 de 1 indica que el modelo puede explicar toda la variación en la variable objetivo, mientras que un valor de R^2 de 0 indica que el modelo no puede explicar ninguna variación.
- Matriz de confusión: Muestra los resultados de clasificación para cada tipo de obesidad. Los valores en la diagonal principal representan las clasificaciones correctas, mientras que los valores fuera de la diagonal principal representan las clasificaciones incorrectas

Los análisis de los resultados de los modelos son los siguientes:

Resultados del modelo de Árbol de decisión

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	46
1.0	0.92	0.97	0.94	35
2.0	0.97	0.85	0.90	33
3.0	0.86	1.00	0.92	12
4.0	1.00	1.00	1.00	24
5.0	1.00	1.00	1.00	3
6.0	1.00	1.00	1.00	18
accuracy			0.96	171
macro avg	0.96	0.97	0.97	171
weighted avg	0.97	0.96	0.96	171

Matriz de confusión:

```
[[46  0  0  0  0  0  0]
 [ 0 34  1  0  0  0  0]
 [ 0  3 28  2  0  0  0]
 [ 0  0  0 12  0  0  0]
 [ 0  0  0  0 24  0  0]
 [ 0  0  0  0  0  3  0]
 [ 0  0  0  0  0  0 18]]
```

Figura 33: Resultados del modelo de árbol de decisión. Elaboración propia.

```
Resultados del modelo de  Naive Bayes
              precision    recall  f1-score   support

         0.0       0.85      1.00      0.92        46
         1.0       0.83      0.71      0.77        35
         2.0       0.93      0.85      0.89        33
         3.0       0.71      0.83      0.77        12
         4.0       0.87      0.83      0.85        24
         5.0       0.50      0.33      0.40         3
         6.0       1.00      1.00      1.00        18

    accuracy                           0.87       171
   macro avg       0.81      0.79      0.80       171
weighted avg       0.87      0.87      0.86       171

Matriz de confusión:
[[46  0  0  0  0  0  0]
 [ 8 25  2  0  0  0  0]
 [ 0  4 28  1  0  0  0]
 [ 0  1  0 10  1  0  0]
 [ 0  0  0  3 20  1  0]
 [ 0  0  0  0  2  1  0]
 [ 0  0  0  0  0  0 18]]
```

Figura 34: Resultados del modelo de Naive Bayes. Elaboración propia.

```
Resultados del modelo de  Regresión Logística
              precision    recall  f1-score   support

         0.0       0.90      0.96      0.93        46
         1.0       0.76      0.80      0.78        35
         2.0       0.87      0.79      0.83        33
         3.0       0.58      0.58      0.58        12
         4.0       0.84      0.88      0.86        24
         5.0       0.00      0.00      0.00         3
         6.0       1.00      1.00      1.00        18

    accuracy                           0.84       171
   macro avg       0.71      0.71      0.71       171
weighted avg       0.83      0.84      0.83       171

Matriz de confusión:
[[44  2  0  0  0  0  0]
 [ 5 28  2  0  0  0  0]
 [ 0  5 26  2  0  0  0]
 [ 0  2  2  7  1  0  0]
 [ 0  0  0  3 21  0  0]
 [ 0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0 18]]
```

Figura 35: Resultados del modelo de Regresión Logística. Elaboración propia.

```
Resultados del modelo de  Random Forest
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99        46
         1.0       0.97      0.97      0.97        35
         2.0       1.00      0.94      0.97        33
         3.0       0.92      1.00      0.96        12
         4.0       0.96      1.00      0.98        24
         5.0       1.00      0.67      0.80         3
         6.0       1.00      1.00      1.00        18

    accuracy                           0.98       171
   macro avg       0.98      0.94      0.95       171
weighted avg       0.98      0.98      0.98       171

Matriz de confusión:
[[46  0  0  0  0  0  0]
 [ 1 34  0  0  0  0  0]
 [ 0  1 31  1  0  0  0]
 [ 0  0  0 12  0  0  0]
 [ 0  0  0  0 24  0  0]
 [ 0  0  0  0  1  2  0]
 [ 0  0  0  0  0  0 18]]
```

Figura 36: Resultado del modelo de Random Forest. Elaboración propia.

```
Resultados del modelo de  SVM
              precision    recall  f1-score   support

         0.0       0.74      1.00      0.85        46
         1.0       0.79      0.43      0.56        35
         2.0       0.58      0.88      0.70        33
         3.0       0.00      0.00      0.00        12
         4.0       0.86      0.75      0.80        24
         5.0       0.00      0.00      0.00         3
         6.0       0.95      1.00      0.97        18

    accuracy                           0.74       171
   macro avg       0.56      0.58      0.55       171
weighted avg       0.69      0.74      0.69       171

Matriz de confusión:
[[46  0  0  0  0  0  0]
 [16 15  4  0  0  0  0]
 [ 0  4 29  0  0  0  0]
 [ 0  0 11  0  1  0  0]
 [ 0  0  6  0 18  0  0]
 [ 0  0  0  0  2  0  1]
 [ 0  0  0  0  0  0 18]]
```

Figura 37: Resultado del modelo de Máquina de Vectores de Soporte (SVM). Elaboración propia.

```
Resultados del modelo de  KNN
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96        46
         1.0       0.90      0.80      0.85        35
         2.0       0.88      0.88      0.88        33
         3.0       0.91      0.83      0.87        12
         4.0       0.92      1.00      0.96        24
         5.0       1.00      0.33      0.50         3
         6.0       0.95      1.00      0.97        18

    accuracy                           0.91       171
   macro avg       0.93      0.84      0.86       171
weighted avg       0.91      0.91      0.91       171

Matriz de confusión:
[[46  0  0  0  0  0  0]
 [ 4 28  3  0  0  0  0]
 [ 0  3 29  1  0  0  0]
 [ 0  0  1 10  1  0  0]
 [ 0  0  0  0 24  0  0]
 [ 0  0  0  0  1  1  1]
 [ 0  0  0  0  0  0 18]]
```

Figura 38: Resultados del modelo de K vecinos más cercanos. Elaboración propia.

A nivel genérico, los modelos han demostrado ser eficaces para el objetivo marcado: predecir el tipo de obesidad de un individuo. No obstante, hay algunos que tienen un rendimiento más alto que otros.

El peor modelo ha sido el de Máquina de vectores de soporte, el cual tiene una precisión bastante más baja que el resto de los algoritmos, de 71’70% en el conjunto de entrenamiento y un 73’68% en el de prueba. Este rendimiento podría deberse a varias razones como, por ejemplo, el ajuste de los hiperparámetros o porque podría indicar que los datos no son fácilmente separables en un espacio de alta dimensión.

En cuanto a los resultados del modelo de Regresión Logística, este tiene una precisión bastante alta, de cerca del 84% tanto en el conjunto de prueba como en el de entrenamiento. Es decir, este algoritmo es capaz de hacer predicciones correctas aproximadamente el 84% del tiempo.

Finalmente, los otros cuatro modelos restantes (Naive Bayes, KNN, Árboles de decisión y Random Forest) muestran todos rendimientos muy elevados, con una precisión de entre el 90% y el 100%.

3.4. Comprobación de los modelos

La gran mayoría de los modelos han obtenido rendimientos excesivamente elevados, lo cual podría ser una indicación de que están sobreajustados, especialmente al tratarse de un conjunto de datos pequeños. Por tanto, se realizará una validación cruzada para confirmar la robustez.

```
from sklearn.model_selection import cross_val_score

# Crear un DataFrame para almacenar los resultados de la validación cruzada
cv_results = pd.DataFrame(columns=['Modelo', 'CV Score (train)', 'CV Score (test)', 'CV Scores (train)', 'CV Scores (test)'])

# Realizar la validación cruzada para cada modelo
for name, model in models:
    # Calcular el score de la validación cruzada para el conjunto de entrenamiento
    cv_scores_train = cross_val_score(model, features_train, target_train, cv=5)
    cv_score_train = cv_scores_train.mean()

    # Calcular el score de la validación cruzada para el conjunto de prueba
    cv_scores_test = cross_val_score(model, features_test, target_test, cv=5)
    cv_score_test = cv_scores_test.mean()

    # Almacenar los resultados
    cv_results = cv_results.append({'Modelo': name, 'CV Score (train)': cv_score_train,
                                    'CV Score (test)': cv_score_test, 'CV Scores (train)': cv_scores_train,
                                    'CV Scores (test)': cv_scores_test}, ignore_index=True)

# Mostrar los resultados de la validación cruzada
print(cv_results)
```

Figura 39: Validación cruzada de los modelos. Elaboración propia.

Esta herramienta es una técnica que se utiliza para evaluar la capacidad de generalización de un modelo, es decir, su capacidad para realizar bien en datos no vistos.

Para cada modelo, se calcula el puntaje de validación cruzada tanto para el conjunto de entrenamiento como para el conjunto de prueba. Este puntaje es una medida de qué tan bien el modelo se ajusta a los datos, donde un puntaje más alto indica un mejor ajuste. Los puntajes de validación cruzada se calculan para cada pliegue en el proceso de validación cruzada para luego hacer un promedio.

Los resultados son los siguientes (Figura 40):

	Modelo	CV Score (train)	CV Score (test)
0	Árbol de decisión	0.986797	0.964706
1	Naive Bayes	0.865135	0.754622
2	Regresión Logística	0.832890	0.830588
3	Random Forest	0.988246	0.947395
4	SVM	0.703832	0.661008
5	KNN	0.914996	0.807227

Figura 40: Resultados de la validación cruzada. Elaboración propia.

De forma general, todos los modelos, con la posible excepción del SVM, muestran una gran capacidad de generalización, por lo que podrían ser útiles para la hora de hacer predicciones en datos no vistos. Asimismo, estos resultados sugieren que los modelos basados en árboles (Árbol de decisión y Random Forest) son los más robustos y fiables.

3.5. Predicciones con nuevos sujetos

Con tal de poder probar la eficacia de los modelos a la hora de introducir un sujeto nunca visto, se procederá a crear un nuevo registro de un individuo con datos reales. De esta forma, se comprobará la predicción resultada por todos los modelos.

Manteniendo las mismas variables, se le ha proporcionado la encuesta al nuevo sujeto, llamado Sujeto 1, y la información recogida se encuentra en la Tabla 5:

Variable	Pregunta	Respuesta
NUM_Gender	Sex	Female (0)
Age	Age	20
Height	Height	1.58
Weight	Weight	59
NUM_family_history_with_overweight	Has a family member suffered or suffers from overweight?	Yes (1)
NUM_FAVC	Frequent consumption of high caloric food	No (0)
NUM_FCVC	Frequency of consumption of vegetables	Sometimes (1)

NCP	Number of main meals	2
NUM_CAEC	Consumption of food between meals	Sometimes (1)
NUM_SMOKE	Do you smoke?	No (0)
NUM_CH2O	Consumption of water daily	Less than a litre (0)
NUM_SCC	Calories consumption monitoring	No (0)
NUM_FAF	Physical activity frequency	1 to 2 (1,2)
NUM_TUE	Time using technology devices	>5 (5)
NUM_CALC	Consumption of alcohol	No (0)
NUM_MTRANS	Transportation used	Automobile (4)
NUM_Nobeyesdad	Type of obesity	-
BMI	Body mass index	23.2

Tabla 5: Datos recogidos del Sujeto 1. Elaboración propia.

Al introducir los datos, las predicciones de los modelos han sido las siguientes:

```
Árbol de decisión predice: Peso Normal
Naive Bayes predice: Peso Normal
Regresión Logística predice: Peso Normal
Random Forest predice: Peso Normal
SVM predice: Peso Normal
KNN predice: Peso Normal
```

Figura 41: Resultados de los modelos con el nuevo individuo. Elaboración propia.

Teniendo en cuenta la clasificación realizada por la Sociedad Española de Obesidad (SEEDO), este sujeto con un índice de masa corporal de 23’2, correspondería a un normopeso. Por lo tanto, se puede concluir que todos los modelos han predicho de forma correcta al individuo.

Con tal de entender mejor cómo se ha llegado a estas conclusiones, se estudiarán con más profundidad los dos modelos que demostraron tener mayor precisión: Random Forest y Árboles de decisión.

Debido a que ya se habían entrenado los modelos con anterioridad, en la Figura 42 y 43 se muestran las principales características más importantes que han tenido en cuenta estos dos algoritmos.

característica	importancia
BMI	0.963353
Weight	0.012720
NUM_SCC	0.008962
NUM_FAVC	0.007492
Height	0.002717

Figura 42: Importancia de las principales características según el árbol de decisión. Elaboración propia.

En el árbol de decisión, la característica más importante en este modelo es el BMI (Índice de masa corporal) con un porcentaje del 96’33%. El resto de las características tienen una importancia cercana al 0%, lo cual indica que no se están teniendo en cuenta para la toma de decisiones en este modelo.

característica	importancia
BMI	0.425534
Weight	0.260681
Height	0.066729
NUM_FAVC	0.028624
NUM_CAEC	0.028475
NUM_FCVC	0.025334
Age	0.022698

Figura 43: Importancia de las principales características según el Random Forest. Elaboración propia.

A diferencia del modelo anterior, el Random Forest tiene en cuenta más características para predecir el resultado. La más importante también es el índice de masa corporal, seguido del peso y la altura. Además, también valora, aunque en pequeños porcentajes, la frecuencia de consumición de alimentos calóricos (NUM_FAVC), el consumo de snacks entre las principales comidas (NUM_CAEC), la frecuencia del consumo de vegetales (NUM_FCVC) y la edad, entre otros.

No obstante, estos modelos no pueden predecir con total certeza el tipo de obesidad que tiene un sujeto. Como se ha observado, la característica más importante que han tenido en cuenta ha sido el índice de masa muscular. El problema del IMC es que no tiene en cuenta otras características importantes como la masa muscular o el porcentaje de grasa.

Para demostrar esta teoría, se han recogido datos reales sobre un joven culturista que compite de forma profesional, llamado a partir de ahora Sujeto Culturista. Los datos se encuentran en la Tabla 6:

Variable	Pregunta	Respuesta
NUM_Gender	Sex	Male (1)
Age	Age	23
Height	Height	1.85
Weight	Weight	102.5
NUM_family_history_with_overweight	Has a family member suffered or suffers from overweight?	No (0)
NUM_FAVC	Frequent consumption of high caloric food	Yes (1)
NUM_FCVC	Frequency of consumption of vegetables	Always (2)
NCP	Number of main meals	4
NUM_CAEC	Consumption of food between meals	Sometimes (1)
NUM_SMOKE	Do you smoke?	Yes (1)
NUM_CH2O	Consumption of water daily	More than 2L (2)
NUM_SCC	Calories consumption monitoring	Yes (1)
NUM_FAF	Physical activity frequency	2 to 4 (2.4)
NUM_TUE	Time using technology devices	>5 (5)
NUM_CALC	Consumption of alcohol	No (0)
NUM_MTRANS	Transportation used	Walking (0)
NUM_Nobeyesdad	Type of obesity	-
BMI	Body mass index	29,95

Tabla 6: Datos del Sujeto Culturista. Elaboración propia.

Los resultados de los algoritmos son los siguientes:

```
Árbol de decisión predice: Sobrepeso grado II
Naive Bayes predice: Peso Normal
Regresión Logística predice: Sobrepeso grado I
Random Forest predice: Sobrepeso grado II
SVM predice: Obesidad tipo I
KNN predice: Obesidad tipo I
```

Figura 44: Resultados de las predicciones para un culturista. Elaboración propia.

En la Figura 44 se puede observar que existe una discrepancia en los resultados de los algoritmos.

El modelo de Árbol de Decisión y el de Random Forest predicen que el individuo tiene "Sobrepeso grado II". Estos modelos funcionan creando una serie de preguntas binarias para llegar a una predicción. Estos modelos pueden ser muy efectivos, pero también pueden ser sensibles a los cambios en los datos de entrenamiento y pueden sobreajustar si hay muchas características.

Los culturistas suelen tener un índice de masa corporal más alto debido a su mayor masa muscular y no necesariamente a un mayor nivel de grasa corporal. Estas condiciones no han sido introducidas en las variables de los modelos, por lo que no han podido tenerlas en cuenta y han fallado en las predicciones.

Sorprendentemente, Naive Bayes, el cual obtuvo la peor precisión en el conjunto de entrenamiento, ha sido el único algoritmo capaz de predecir que el sujeto no padece de obesidad.

Por tanto, las diferencias en las predicciones reflejan las diferentes formas en que estos modelos aprenden de los datos. Cada modelo tiene sus propias fortalezas y debilidades y puede ser más o menos adecuado dependiendo del problema del conjunto de datos disponible.

3.5. Conclusiones del caso práctico

Tras el análisis de los algoritmos, se puede llegar a la conclusión de que los modelos han podido lograr el resultado esperado: predecir la clase de obesidad de un sujeto.

No obstante, se ha podido demostrar las limitaciones que presentan los algoritmos con las variables actuales, puesto que, en el caso de deportistas de élite, la precisión de estos se ve gravemente afectado.

Asimismo, no todos los modelos han tenido el mismo rendimiento. En Regresión Logística y SVM, la precisión y el valor R^2 en el conjunto de entrenamiento han sido ligeramente inferiores a los del conjunto de prueba. Esto puede parecer contra-intuitivo, ya que normalmente se espera que un modelo tenga un rendimiento mejor o al menos similar en el conjunto de entrenamiento en comparación con el conjunto de prueba.

En el caso del SVM, una posible explicación de este suceso podría ser que el modelo está realizando las predicciones con un menor margen, es decir, una menor distancia entre la línea de decisión y los ejemplos de entramientos más cercanos. Esto conduce a que a que algunos ejemplos de entramiento se clasifiquen incorrectamente debido a la complejidad de separación de los datos en un espacio de alta dimensión.

En cuanto al caso de la Regresión Logística, al ser un modelo que estima la probabilidad de que un ejemplo pertenezca a una clase particular, si el conjunto de prueba es más "fácil" de predecir que el conjunto de entrenamiento, entonces el modelo puede tener un rendimiento mejor en el conjunto de prueba.

Asimismo, es interesante notar que los modelos basados en árboles (Árbol de decisión y Random Forest) parecen tener un rendimiento superior en este conjunto de datos en

comparación con los otros modelos. Esto podría sugerir que las características tienen relaciones no lineales y complejas con la variable objetivo y que estos modelos son capaces de capturarlas.

Aunque ambos modelos han demostrado ser efectivos, el bosque aleatorio suele dar resultados más robustos y puede ser preferible cuando se desee maximizar la precisión. En cambio, el árbol de decisión podría ser útil cuando la interpretabilidad sea un factor importante o el conjunto de datos sea pequeño.

En definitiva, no existe un único modelo adecuado para lograr clasificar un conjunto de datos, sino que todos tienen sus ventajas e inconvenientes que les hacen únicos y es por ello por lo que se debe de comprobar el rendimiento y la precisión de cada uno atendiendo a las características de los datos.

4. Conclusiones

Una vez desarrollado este Trabajo de Fin de Grado, se puede concluir que la inteligencia artificial tiene un papel fundamental para la toma de decisiones en cualquier ámbito del mundo actual.

Como se ha demostrado, los algoritmos de clasificación han sido capaces de predecir desde la pertenencia de una categoría de una flor hasta identificar si un correo es considerado spam o si un individuo padece de obesidad. El uso de estas herramientas ha llegado al ámbito profesional con el fin de facilitar a las empresas a tomar decisiones más informadas y basadas en los datos.

Existen numerosos beneficios que pueden ser extraídos de la inteligencia artificial, logrando una mayor eficiencia operativa y ahorro de costes. Asimismo, su aplicación no está exenta de desafíos. Es muy importante que las organizaciones tengan una sólida comprensión de los datos y que estos sean recogidos y almacenados de la forma correcta; pues unos datos de mala calidad, irrelevantes o una interpretación incorrecta de los resultados del modelo pueden llevar a tomar decisiones erróneas.

Este problema se ha visto reflejado como una limitación en el presente estudio. Debido a que no se ha logrado obtener la suficiente muestra de datos, la calidad de los modelos de aprendizaje automático ha sido muy reducida a la hora de intentar predecir la obesidad en un deportista de élite con gran masa muscular.

Algunas posibles vías de investigación futura para abordar este problema serían ampliar el conjunto de datos o la exploración de otros algoritmos, como las redes neuronales, con el fin de intentar mejorar los resultados.

Por tanto, se concluye este TFG confirmando que la inteligencia artificial tiene un papel crucial en la actualidad y continuará desarrollándose y aportando más beneficios a la sociedad en los próximos años, llegando así a la Revolución Industrial 5.0.

5. Bibliografía.

Aizerman, M. A., Braverman, E. M., & Rozonoer, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25(6), 821-837.

Alaminos-Fernández, A. F. (2022). Árboles de decisión en R con Random Forest.

Alpaydin, E. (2010). Introduction to machine learning (2nd ed.). MIT Press.

Berkson, J. (1944). Application of the logistic function to bioassay. Journal of the American Statistical Association, 39(227), 357-365.

Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140.

Carles Vega, J. (2023). Modelización de los factores que inciden en el rendimiento académico de los estudiantes universitarios con técnicas de estadística multivariante y de machine learning (Doctoral dissertation, Universitat Politècnica de València).

Chandra, B., Gupta, M. y Gupta, M.P. Robust Approach for Estimating Probabilities in Naive-Bayes Classifier. In International Conference on Pattern Recognition and Machine Intelligence (pp. 11-16), Springer, Berlin, Heidelberg, December (2007).

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273–297.

Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. Ecology, 88(11), 2783-2792.

Datos.gob.es. (2020). ¿Cómo aprenden las máquinas? Machine Learning y sus tipos. https://datos.gob.es/es/blog/como-aprenden-las-maquinas-machine-learning-y-sus-tipos

DeepMind Technologies. (2016). AlphaGo [Software]. Disponible en http://deepmind.com/research/alphago/

Floyd Pérez, J. D. (2022). Efecto de un programa de ejercicios funcionales en el medio acuático en los niveles de obesidad de la población adulta de la clínica de rehabilitación de la ciudad de Guadalajara de Buga (Bachelor's thesis, Licenciatura en Educación Física, Recreación y Deporte).

Galton, F. (1886). Regression towards mediocrity in hereditary stature. The Journal of the Anthropological Institute of Great Britain and Ireland, 15, 246-263.

Juan, A. A. (2020). Data Science with Python: NumPy and Pandas. Hot Topics in Analytics.

Juan, A. A. (2023) Supervised Machine Learning (II). Hot Topics in Analytics. Statistical Tools for Business Data Analytics

Samuel, A. L. (1959). "Some studies in machine learning using the game of checkers." (“Some Studies in Machine Learning Using the Game of Checkers”) IBM Journal of Research and Development, 3(3), 210-229.

Simeone, O. (2018). "A Very Brief Introduction to Machine Learning with Applications to Communication Systems." (“A Very Brief Introduction to Machine Learning with Applications to ...”) Repositorio Universidad de Cornell.

Río Navarro, B. E. D., & Sienra Monge, J. J. L. (2011). Relación de la obesidad con el asma y la función pulmonar. Boletín médico del Hospital Infantil de México, 68(3), 171-183.

Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd ed.). Pearson.

Vapnik, V. (1995). The nature of statistical learning theory. New York: Springer. "SVM is a new approach to classification based on the statistical learning theory and the theory of optimization."

Weston, J. (2003). Support Vector Regression Machines. En D. Michael (Ed.), Advances in Kernel Methods: Support Vector Learning (pp. 73-92). The MIT Press.