Gradient Boosting

Walys Vera Herrera, Javier Enrique Aguilar

Resumen. Gradient Boosting es una poderosa técnica de aprendizaje automático que ha ganado popularidad debido a su capacidad para crear modelos predictivos robustos y precisos. Este método de aprendizaje combina las fortalezas de los árboles de decisión y algoritmos de optimización para mejorar de manera iterativa el rendimiento del modelo. Gradient Boosting funciona ajustando una sucesión de aprendices débiles, generalmente árboles de decisión, a los residuos de los modelos previos, reduciendo gradualmente los errores de predicción. Este proceso le permite manejar eficazmente varios tipos de datos, incluyendo características numéricas y categóricas, y se utiliza ampliamente en tareas de regresión y clasificación. Implementaciones destacadas como XGBoost, LightGBM y AdaBoost han convertido a Gradient Boosting en una herramienta indispensable para resolver problemas complejos del mundo real en campos como finanzas, atención médica y procesamiento de lenguaje natural.

Presentamos un resumen de teoría de aprendizaje estadístico, luego árboles de decisión y por último Gradient Boosting. Para desarrollar este proyecto hemos utilizado como guía principal el texto "An Introduction to Statistical Learning with Applications in Python"James et al. (2023).

Keywords: Machine learning, Statistical Learning, Boosting.

1. Introducción

El boosting, un metaalgoritmo de ensamblaje en machine learning, tiene como objetivo principal potenciar un conjunto de regresores o clasificadores débiles para convertirlos en un modelo fuerte y robusto. En el ámbito del aprendizaje automático, un clasificador débil se define como aquel que apenas supera el desempeño aleatorio. Este enfoque surge como respuesta a la pregunta planteada por Kearns y Valiant en 1988: "¿Puede un conjunto de clasificadores deficientes generar un clasificador fuerte?". En 1990, Robert Schapire responde afirmativamente a esta interrogante.

La esencia del boosting radica en la motivación de combinar las salidas de numerosos clasificadores débiles para crear un comité fuerte y preciso. El algoritmo de boosting más destacado, conocido como .ªdaboostz desarrollado por Freund y Schapire en 1997, ha sido fundamental en este ámbito. No obstante, en los últimos años, una variante de boosting denominada XGBoost, presentada por Chen y Guestrin, ha ganado una notable popularidad al destacar constantemente en competiciones de renombre, como Kaggle o la KDDCup.

*Universidad de Antioquía
walys.vera@udea.edu.co
**TU Dortmund
javier.aguilar@tu-dortmund.de url: https://jear2412.github.io

2. Aprendizaje Estadístico

En este capítulo, nos sumergimos en el vasto universo del aprendizaje estadístico, una disciplina esencial en la ciencia de datos y análisis. La misión central es adentrarnos en el análisis de datos, desentrañando patrones, relaciones y regularidades cruciales para comprender fenómenos complejos.

La esencia del aprendizaje estadístico se materializa en la aplicación práctica de la información extraída de los datos. Nos dirigimos al desarrollo de modelos precisos, habilidad fundamental para prever resultados y fundamentar decisiones. Este enfoque cobra especial relevancia en entornos donde los datos son observados y la tarea es descifrar las interconexiones entre variables o anticipar futuros resultados.

Consideremos la observación de una respuesta cuantitativa y y p predictores diferentes, $x = (x_1, x_2, \ldots, x_p)'$. Suponemos que existe alguna relación entre y y x, expresada de manera general como:

$$y = f(x) + \varepsilon,$$

donde f es una función suficientemente general, pero desconocida. Destacamos que estamos observando una respuesta cuantitativa y y múltiples predictores x, aspecto esencial para comprender y modelar la relación subyacente entre estas variables. En esta ecuación, f representa la información sistemática que x proporciona sobre y, mientras que ε encapsula la variabilidad no sistemática o aleatoria en esta relación, con una media de cero. Este equilibrio entre la información sistemática y la variabilidad no sistemática sienta las bases para la modelización y predicción en el aprendizaje estadístico, es decir, que el objetivo del aprendizaje estadístico es conocer quién es f.

2.1. ¿Por qué estimar f?

Hay dos razones principales por las que podríamos querer estimar f: la predicción y la inferencia.

Predicción La predicción es el proceso de estimar o calcular un valor desconocido o resultado futuro y basado en información disponible x, incluso cuando no podemos obtener directamente ese valor futuro. En el contexto del aprendizaje estadístico, esta predicción se logra mediante la estimación de una función $\hat{f}$ que relaciona las variables de entrada x con la variable de salida y. En situaciones donde las entradas x son conocidas, pero la salida y no es fácilmente obtenible, la predicción se vuelve invaluable. Utilizando modelos estadísticos, como el propuesto por la ecuación $\hat{y} = \hat{f}(x)$, donde $\hat{f}$ es nuestra estimación para la función desconocida f, podemos realizar predicciones para y basadas en la información proporcionada por x.

La precisión de nuestras predicciones $\hat{y}$ está influenciada por dos tipos de errores: el error reducible, asociado con la inexactitud en nuestra estimación $\hat{f}$, y el error irreducible, proveniente de la variabilidad no predecible de y debido a factores incontrolables.

Aunque podemos trabajar para reducir el error reducible mediante técnicas estadísticas avanzadas, el error irreducible sigue siendo inherente, ya que ε no puede preverse utilizando x.

- **Error Reducible:** Imagina que estás tratando de predecir la altura de una persona y en función de la cantidad de comida que consume al día x. Si utilizas un modelo simple que considera solo la cantidad de comida, tu predicción tendrá un error. Sin embargo, si mejoras tu modelo y consideras no solo la cantidad de comida, sino también otros factores como la cantidad de ejercicio que hace la persona, la precisión de tu predicción mejorará. Este error se llama reducible"porque puedes reducirlo al mejorar tu modelo. En pocas palabras, el error reducible puede reducirse mejorando el modelo y considerando más variables, mientras que el error irreducible siempre estará presente debido a factores impredecibles e incontrolables

- **Error Irreducible:** Ahora, supongamos que has mejorado tu modelo al máximo y consideras todos los factores posibles que podrían influir en la altura de una persona. Aun así, siempre habrá una variabilidad inherente en la altura de las personas que no puedes eliminar. Por ejemplo, factores genéticos, cambios naturales en el crecimiento y otras influencias impredecibles pueden afectar la altura de una persona, y no puedes controlarlos ni preverlos con precisión. Este error se llama ïrreducible"porque no importa cuánto mejores tu modelo, siempre habrá una parte de la variabilidad que no puedes eliminar.

Inferencia La inferencia se refiere a la búsqueda y comprensión de las relaciones entre variables para responder preguntas sobre cómo se relacionan y qué significado tienen en un conjunto de datos, sin necesariamente enfocarse en hacer predicciones.

¿Cómo estimamos f?

En esta sección, abordaremos la estimación de una función desconocida que relaciona variables de entrada y salida. Exploraremos únicamente el enfoque paramétricos. Este enfoque se basa en dos pasos clave. Primero suponemos la forma funcional de la relación entre la variable de respuesta y y las variables predictoras x. Por ejemplo, en el caso de suponer una relacion lineal entre y y x, podemos colcar $f(x_1, ..., x_p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$ en donde β_0 es el intercepto y $\beta_1, \beta_2, \ldots, \beta_p$ son los coeficientes de regresión. $x_1, x_2, \ldots, x_p$ son las variables predictoras. Nuestro objetivo en este caso seria conocer quienes son $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$.

Una vez que se ha elegido la forma de f, se procede a estimar los parámetros del modelo. En el caso anterior, esto implica encontrar valores específicos para los parámetros $(\beta_0, \beta_1, \ldots, \beta_p)$ que mejor se ajusten a los datos observados. El método de estimación más común es el de "Mínimos Cuadrados,"que busca minimizar el cuadrado de la diferencia entre los valores predichos por el modelo y los valores reales observados.

2.2. El tradeoff entre interpretabilidad y precisión

El tradeoff entre interpretabilidad y precisión en modelos estadísticos implica la elección estratégica entre dos aspectos cruciales al desarrollar un modelo. Por un lado, la interpretabilidad se refiere a la capacidad de entender y explicar el modelo de manera clara y sencilla. Los modelos interpretables son especialmente valiosos cuando el objetivo principal es realizar inferencias sobre la relación entre variables. Por otro lado, la precisión hace referencia a la capacidad del modelo para hacer predicciones precisas y exactas, siendo esencial en situaciones donde la prioridad es obtener predicciones altamente confiables sin necesariamente entender completamente la estructura interna del modelo. En el contexto de aprendizaje estadístico, donde el objetivo principal es comprender las relaciones y los efectos de las variables, se prefieren modelos más simples y restrictivos. Ejemplos incluyen la regresión lineal, que ofrece una interpretación clara, pero puede ser menos flexible en la representación de patrones complejos.

En contraste, en situaciones donde solo se busca predecir de manera precisa sin una necesidad inmediata de entender el modelo, se podría favorecer la utilización de modelos altamente flexibles. Sin embargo, paradójicamente, modelos extremadamente flexibles pueden conducir al sobreajuste, comprometiendo la precisión al adaptarse en exceso a los datos de entrenamiento.

En pocas palabras, el tradeoff entre interpretabilidad y precisión implica tomar decisiones estratégicas según los objetivos específicos del análisis. Si bien modelos más complejos pueden ofrecer mayor precisión en predicciones, a menudo a expensas de la interpretabilidad, es esencial encontrar un equilibrio que se alinee con las necesidades particulares de la tarea en cuestión. Este compromiso se hace evidente al considerar cuidadosamente el contexto y los requisitos específicos de cada situación.

2.3. Aprendizaje Supervisado VS Aprendizaje No Supervisado

Dentro del panorama del aprendizaje automático, es esencial comprender las dos categorías principales en las que se dividen los problemas: el aprendizaje supervisado y el no supervisado. Cada uno de estos enfoques tiene sus propias características distintivas y aplicaciones.

El aprendizaje supervisado es un enfoque dentro del aprendizaje automático donde se entrena un modelo utilizando un conjunto de datos etiquetado. En este conjunto de datos, cada observación se acompaña de una etiqueta o resultado conocido. El objetivo fundamental del modelo consiste en aprender una relación significativa entre las características de entrada y las etiquetas de salida, con el propósito de realizar predicciones precisas sobre datos no etiquetados.

En otras palabras, el proceso implica que el modelo asimile patrones y regularidades presentes en los datos de entrenamiento para luego aplicar ese conocimiento adquirido a nuevas instancias sin etiquetas. Este método se revela especialmente útil en situaciones donde se desea predecir o clasificar datos futuros basándose en la información disponible en el conjunto de entrenamiento. El aprendizaje supervisado abarca una amplia variedad

de aplicaciones prácticas, desde la clasificación de imágenes hasta la predicción de precios en mercados financieros, destacando su versatilidad y utilidad en distintos campos.

El aprendizaje no supervisado es un enfoque en el aprendizaje automático donde se trabaja con datos no etiquetados o sin respuestas conocidas. El objetivo principal es descubrir patrones, estructuras o agrupaciones en los datos sin la guía de etiquetas predefinidas. Esto puede incluir técnicas como el análisis de clústeres, la reducción de dimensionalidad y la detección de anomalías.

2.4. Medición de la Calidad del Ajuste

En esta sección, abordamos la evaluación del rendimiento de un método de aprendizaje estadístico mediante la introducción de una función de pérdida L. Esta función cuantifica la relación entre nuestras predicciones $\hat{y}$ y las observaciones reales y.

Por ejemplo, la función de pérdida cuadrática se define como:

$$L(y, \hat{y}) = (y - \hat{y})^2,$$

donde y representa las observaciones reales y $\hat{y}$ denota nuestras predicciones. En general, la elección de la función de pérdida es fundamental para el proceso de aprendijaze del modelo y determina directamente que tan complejo puede ser ajustar un modelo.

Para la función anterior, podemos evaluar el rendimiento de un modelo en un conjunto de datos dado utilizando el Error Cuadrático Medio (MSE), que se define como:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(x_i)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2.$$

Donde n es el número de observaciones en el conjunto de datos, y_i son las observaciones reales, y $\hat{f}(x_i)$ son las predicciones del modelo para la i-ésima observación.

Es importante destacar que el MSE se calcula utilizando los datos de entrenamiento utilizados para ajustar el modelo, y se le conoce más adecuadamente como MSE de entrenamiento. Sin embargo, nuestro interés principal radica en la precisión de las predicciones cuando se aplican a datos no vistos previamente. Por lo tanto, buscamos minimizar el MSE en un conjunto de prueba para seleccionar un modelo robusto que generalice bien a datos nuevos. Este proceso se complica cuando no se dispone de un conjunto de prueba independiente. La elección de la función de pérdida y la cuidadosa consideración del MSE son aspectos esenciales para evaluar y comparar modelos de aprendizaje estadístico.

2.5. Trade-off entre sesgo y varianza

El trade-off entre sesgo y varianza se refiere a un equilibrio crítico que debe gestionarse al desarrollar modelos predictivos. Este concepto ilustra la compensación necesaria entre dos aspectos clave de un modelo:

1. **Sesgo (*Bias*):** El sesgo se refiere a las simplificaciones que realiza un modelo sobre la relación subyacente entre las variables de entrada y la variable objetivo. Un modelo con alto sesgo tiende a hacer suposiciones excesivamente simples sobre la estructura del conjunto de datos y puede no capturar todas sus complejidades.

2. **Varianza:** La varianza mide la sensibilidad del modelo a pequeñas variaciones en los datos de entrenamiento. Un modelo con alta varianza se ajusta muy bien a los datos de entrenamiento, pero puede ser sensible al ruido y tener dificultades para generalizar a nuevos datos que no forman parte del conjunto de entrenamiento.

El equilibrio se busca porque reducir el sesgo a menudo aumenta la varianza y viceversa. Aquí hay algunas consideraciones clave:

- Modelos más simples, con alto sesgo, tienden a ser más estables pero pueden no adaptarse bien a datos complejos.
- Modelos más complejos, con baja varianza, pueden adaptarse mejor a datos complejos pero pueden ser propensos a sobreajustar el ruido en los datos de entrenamiento.

Para la perdida cuadrática, el trade-off entre sesgo y varianza se describe mediante la ecuación

$$E\left[(y_0 - \hat{f}(x_0))^2\right] = \text{Var}(\hat{f}(x_0)) + \left[\text{Bias}(\hat{f}(x_0))\right]^2 + \text{Var}(\varepsilon).$$

En esta ecuación, el término $E\left[(y_0 - \hat{f}(x_0))^2\right]$ representa el error cuadrático medio esperado en x_0. El error ε se supone usualmente centrado en cero con varianza desconocida σ^2. Este error se descompone en tres componentes principales:

1. **Varianza de $\hat{f}(x_0)$:** Mide cuánto variarían las predicciones de $\hat{f}(x_0)$ si entrenáramos el modelo con diferentes conjuntos de datos. Una varianza alta implica que pequeñas variaciones en los datos de entrenamiento generan grandes cambios en las predicciones.

2. **Sesgo cuadrático de $\hat{f}(x_0)$:** Representa la diferencia entre la predicción promedio de $\hat{f}(x_0)$ y el valor real y_0. Un sesgo alto indica que el modelo simplifica demasiado las relaciones subyacentes en los datos.

3. **Varianza de ε:** Es la varianza del término de error, que refleja la variabilidad inherente e irreducible en el proceso generador de datos.

El objetivo es encontrar un equilibrio óptimo entre estos tres componentes. Modelos con alto sesgo tienden a subajustar los datos, mientras que modelos con alta varianza tienden a sobreajustarlos. El trade-off implica que reducir el sesgo puede aumentar la

varianza y viceversa. La elección de la complejidad del modelo afecta este trade-off, y se busca un punto donde el error total sea mínimo. En resumen, el sesgo-varianza trade-off es esencial para entender cómo la elección de modelos impacta en la capacidad de generalización y ajuste a los datos de entrenamiento.

3. Árboles

Los árboles son modelos de decisión que se utilizan para abordar problemas de regresión y clasificación. Estos métodos implican la división del espacio de predictores en J regiones simples R_j, facilitando así la toma de decisiones basada en las características de entrada. En esencia, un árbol de decisión representa un conjunto de reglas que guían la asignación de una observación a una categoría o la predicción de un valor numérico. Este procedimiento se puede visualizar en la figura 1.

La simplicidad y capacidad de interpretación hacen que los árboles sean herramientas valiosas en estadística. Al realizar split, es decir donde se hace la decisión,[1] en el espacio de variables predictoras de manera jerárquica, los árboles permiten visualizar y entender las relaciones complejas entre las variables.

Un árbol se expresa formalmente de la siguiente manera

$$T(x;\Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j), \tag{3.1}$$

en donde los parámetros se representan mediante $\Theta = \{R_j, \gamma_j\}_{j=1}^{J}$ e $I(\cdot)$ es la función indicadora. Aquí, J suele tratarse como un meta-parámetro fijado por el usuario. Los parámetros se encuentran minimizando el riesgo empírico:

$$\Theta = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j), \tag{3.2}$$

en donde L es la función de pérdida especificada por el usuario.

Este es un problema de optimización combinatoria formidable, y generalmente nos conformamos con soluciones subóptimas aproximadas. Es útil dividir el problema de optimización en dos partes:

1. Encontrar γ_j dado R_j: Dados los R_j, estimar los γ_j suele ser trivial. En el caso de regresión y con una perdida cuadrática se tiene $\hat{\gamma}_j = \bar{y}_j$, la media de los y_i que caen en la región R_j. En el caso de clasificación cuando se utiliza la perdida por clasificación incorrecta, $\hat{\gamma}_j$ es la clase modal de las observaciones que caen en la región R_j.

[1] Para detalles sobre como se realiza el split, ver la referencia.

2. Encontrar R_j: Esta es la parte difícil, para la cual se encuentran soluciones aproximadas. También es necesario tener en cuenta que encontrar los R_j implica estimar los γ_j también. Una estrategia típica es usar un algoritmo de partición recursivo, descendente y codicioso para encontrarlos R_j. Además, a veces es necesario aproximar el riesgo empírico con un criterio más suave y conveniente para optimizar los R_j.

Este tipo de modelo se utiliza comúnmente en métodos de ensamblado, como Gradient Boosting, donde varios árboles se combinan para formar un modelo más complejo y preciso.

3.1. Tipos de árboles

Dependiendo de la tarea, podemos clasificar los árboles en árboles de regresión y clasificación cuando y es cuantitativa y cualitativa respectivamente.

Árboles de regresión

En este caso nos interesa predecir un valor numérico. Estos árboles representan un enfoque gráfico y jerárquico para dividir el espacio de las variables predictoras en regiones que se ajusten a la variabilidad de los datos y permitan realizar predicciones precisas.

El proceso de construcción de un árbol de regresión implica dividir recursivamente el conjunto de datos en subconjuntos más pequeños basándose en las variables predictoras. Cada división se elige de manera que minimice la varianza del objetivo dentro de cada región resultante. Al final del proceso, cada hoja del árbol contiene la predicción de regresión para las observaciones que caen en esa región.

Como mencionamos anteriormente, el objetivo al construir un árbol de regresión es encontrar cajas $R_1, ..., R_J$. En este caso queremos minimizar la perdida cuadratica, por lo que buscamos que minimicen la Suma Residual de Cuadrados (RSS), dada por:

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Donde $\hat{y}_{R_j}$ representa la respuesta promedio para las observaciones de entrenamiento dentro de la j-ésima caja. La elección de divisiones se realiza de forma recursiva, buscando minimizar la RSS en cada paso. Este proceso implica dividir recursivamente el espacio de los predictores en regiones, tomando decisiones que minimizan el RSS en cada paso. La estructura resultante del árbol, con su segmentación jerárquica, se construye para capturar de la mejor manera posible las relaciones subyacentes en los datos, proporcionando un modelo claro e interpretable para el análisis de regresión. La ecuación (RSS) cuantifica la discrepancia entre las respuestas observadas y las predichas, y el objetivo es minimizar esta discrepancia al construir el árbol.

Árboles de clasificación

Los árboles de clasificación son modelos de aprendizaje supervisado diseñados para resolver problemas de clasificación, donde el objetivo es asignar instancias de entrada a categorías o clases discretas. La representación de estos modelos adopta la forma de un árbol jerárquico, donde cada nodo interno representa una decisión basada en una característica específica, y las hojas del árbol identifican las clases a las que se asignan las instancias. Este procedimiento se ilustra en la figura 2.

Un árbol de clasificación es muy similar a un árbol de regresión, excepto que se utiliza para predecir una respuesta cualitativa en lugar de una cuantitativa. Para un árbol de clasificación, se predice que cada observación pertenece a la Clase más común de las observaciones de entrenamiento en la región a la que pertenece.

El proceso de crecimiento de un árbol de clasificación es similar al de un árbol de regresión. Se utiliza la partición binaria recursiva para hacer crecer el árbol, pero en lugar de utilizar RSS, se recurre a alternativas como la tasa de error de clasificación, el Índice Gini y la entropía para evaluar la calidad de las divisiones.

La tasa de error de clasificación se define como $E = 1 - \max(p_{mk})$ en donde p_{mk} representa la proporción de observaciones de entrenamiento en la región m que pertenecen a la clase k.

El Índice Gini se define como $G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$. Este es una medida de la variabilidad total entre las K clases. Por último, la entropía se define como $D = -\sum_{k=1}^{K} p_{mk} \log(p_{mk})$. Ambos enfoques son más sensibles a la pureza del nodo que la tasa de error de clasificación, siendo preferibles para evaluar la calidad de las particiones.

3.2. Ventajas y desventajas de los árboles

Los árboles de decisión destacan por su capacidad para capturar relaciones no lineales y patrones complejos en los datos. Esta flexibilidad los hace particularmente útiles cuando las relaciones entre las variables no son lineales. Además, su estructura jerárquica brinda una interpretación intuitiva, facilitando la comprensión de las decisiones tomadas por el modelo.

Un usuario puede optar por árboles de decisión cuando los datos presenten relaciones complejas, sobre todo si se sospechan relaciones no lineales. Tambien si se desea un modelo que sea robusto ante valores atípicos y variables categóricas.

Uno de las mayores desventajas de los arboles es la tendencia al sobreajuste, la cual se refiere a la propensión del modelo a adaptarse demasiado a los detalles específicos de los datos de entrenamiento, lo que puede llevar a un rendimiento deficiente en nuevos datos no vistos. Este problema puede ser mitigado mediante estrategias adicionales, como la poda del árbol (La poda del árbol es una técnica utilizada para reducir la complejidad de un árbol de regresión al eliminar nodos (ramas y hojas) que no proporcionan beneficios significativos en términos de mejora en la capacidad de generalización del modelo.), que ayuda a simplificar la estructura del modelo y mejorar su capacidad para generalizar a

Aspecto	Ventajas	Desventajas
Interpretación intuitiva	Los árboles son fácilmente interpretables y permiten comprender visualmente cómo se toman las decisiones en cada nivel del árbol.	Pueden ser propensos al sobreajuste (overfitting) si no se controlan adecuadamente, especialmente en árboles profundos.
Eficiencia	Una vez construido, el árbol puede predecir nuevas instancias de manera rápida y eficiente, ya que simplemente sigue el camino desde la raíz hasta una hoja.	Pueden ser inestables, es decir, pequeños cambios en los datos pueden llevar a grandes cambios en la estructura del árbol.
Manejo automático de características	No es necesario escalar o normalizar características antes de ajustar un árbol de clasificación, ya que manejan automáticamente diferentes escalas y tipos de características.	Sensibles a pequeñas variaciones en los datos, lo que puede llevar a la generación de árboles subóptimos.
Versatilidad en tipos de clases	Pueden manejar problemas de clasificación con múltiples clases (clases multiclase) sin requerir ajustes significativos.	Pueden no ser tan precisos como otros modelos en ciertos conjuntos de datos complejos.
No requiere supuestos de distribución	Los árboles de clasificación no asumen una distribución específica de los datos, lo que los hace flexibles y aplicables a diversas situaciones.	La calidad de las predicciones puede ser inferior en comparación con métodos más avanzados, especialmente en problemas más complejos.

Cuadro 1: Ventajas y desventajas de los árboles de decisión.

datos no utilizados durante el entrenamiento. Presentamos las ventajas y desventajas de los arboles en la tabla 1

4. Ensamblado de modelos

El ensamblado de modelos es un enfoque poderoso en el aprendizaje automático que combina múltiples modelos simples para obtener un modelo único y potencialmente muy eficaz. Estos modelos de construcción simple se conocen a veces como “aprendices débiles”, ya que por sí solos pueden proporcionar predicciones mediocres.

Por ejemplo, consideremos un aprendiz débil que es un clasificador que solo puede predecir correctamente si la temperatura es alta o baja, pero falla en días con temperaturas moderadas. Este clasificador, por sí solo, no es muy efectivo. Sin embargo, al combinarlo con otros aprendices débiles que pueden manejar diferentes condiciones cli-

máticas, como lluvia o viento, el ensamblado de modelos puede generar un clasificador robusto capaz de manejar una variedad de situaciones climáticas.

A continuación, discutiremos una técnica de ensamblado: boosting que está basada en la construcción de modelos de regresión o clasificación.

4.1. Boosting con árboles

Boosting con árboles o árboles de impulso es una suma de árboles individuales, expresado como:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m),$$

en donde M denota el numero de modelos debiles a utilizar y $T(\cdot, \cdot)$ representa un árbol con parámetros Θ_m. Este modelo se construye de manera progresiva mediante el algoritmo de impulso secuencial hacia adelante (Algoritmo *). En cada paso de este procedimiento, se debe resolver el problema de optimización

$$\Theta_m^* = \underset{\Theta_m}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)\right),$$

donde L es la función de pérdida especificada por el usuario, $f_{m-1}(x)$ es el modelo actual y $T(x; \Theta_m)$ es el árbol inducido.

Algoritmo* - Impulso Secuencial Hacia Adelante

1. Inicializar el modelo con $f_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma)$.
2. Para $m = 1$ hasta M:
   a) Calcular los residuos generalizados $r_{im} = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$, donde $f_{m-1}(x)$ es el modelo actual.
   b) Ajustar un árbol de regresión a los residuos, generando regiones terminales R_{jm}.
   c) Calcular las constantes $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$.
   d) Actualizar el modelo $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Obtener el modelo final $\hat{f}(x) = f_M(x)$.

El procedimiento de boosting implica ajustar secuencialmente estos árboles, cada uno de los cuales se enfoca en corregir los errores residuales cometidos por los modelos anteriores. Los árboles individuales se ajustan utilizando versiones modificadas del conjunto de datos original, donde se da más peso a las instancias mal clasificadas.

La formulación de boosting destaca su enfoque iterativo para mejorar la precisión del modelo, aprendiendo de los errores de los modelos previos y combinándolos de manera ponderada para obtener una predicción final más robusta.

5. Gradient Boosting

El *Gradient Boosting* se destaca por su capacidad para construir modelos altamente precisos mediante la combinación secuencial y estratégica de modelos más simples, generalmente árboles de decisión poco profundos. Cada nuevo árbol se construye secuencialmente para corregir y mejorar los errores identificados por los árboles previos. La predicción para una nueva observación se obtiene al combinar las contribuciones de todos los árboles individuales que conforman el modelo. Este potente algoritmo de aprendizaje automático sobresale en problemas de regresión y clasificación. Su singularidad radica en la construcción secuencial de modelos débiles, generalmente representados por árboles de decisión poco profundos. El enfoque secuencial, donde cada nuevo modelo se centra en mejorar las deficiencias del anterior, es una característica distintiva del Gradient Boosting y contribuye significativamente a su capacidad para aprender patrones complejos y construir modelos altamente precisos.

La esencia del Gradient Boosting consiste en mejorar iterativamente la capacidad predictiva del modelo mediante la fusión de información de modelos más simples. En cada iteración, el algoritmo se centra en corregir los errores residuales del modelo actual, ajustando un nuevo modelo débil específicamente diseñado para abordar esas deficiencias. En el contexto del Gradient Boosting con árboles de decisión, cada árbol se entrena para predecir los residuos del modelo anterior. La combinación progresiva de estos árboles contribuye gradualmente a mejorar la precisión del modelo global. Este enfoque, basado en la optimización secuencial, destaca por su capacidad para construir modelos altamente precisos mediante la amalgama estratégica de modelos más simples, convirtiéndolo en una herramienta valiosa en el campo del aprendizaje automático.

Es decir, el Gradient Boosting es una técnica de ensamblado de modelos en aprendizaje automático que combina múltiples modelos simples, generalmente árboles de decisión débiles, para formar un modelo más robusto y preciso. La idea central detrás de Gradient Boosting es construir modelos de manera secuencial, enfocándose en mejorar la predicción en las áreas donde los modelos anteriores han tenido un rendimiento deficiente.

La idea del Gradient Boosting se originó en la observación de Leo Breiman de que el boosting puede interpretarse como un algoritmo de optimización en una función de costo adecuada. Posteriormente, se desarrollaron algoritmos explícitos de boosting para regresión, por Jerome H. Friedman, simultáneamente con la perspectiva más general de Gradient Boosting funcional de Llew Mason, Jonathan Baxter, Peter Bartlett y Marcus Frean. Ellos introdujeron la visión de los algoritmos de boosting como algoritmos iterativos de descenso de gradiente funcional. Es decir, algoritmos que optimizan una función de costo en el espacio de funciones al elegir iterativamente una función (hipótesis débil) que apunta en la dirección del gradiente negativo.

El proceso de Gradient Boosting se realiza en etapas sucesivas, y cada etapa agrega un nuevo modelo al conjunto existente. La construcción del nuevo modelo se guía mediante la minimización de los errores residuales del modelo anterior. En otras palabras, el nuevo modelo se ajusta a la diferencia entre las predicciones del modelo actual y las observaciones reales.

La fórmula general para el modelo de Gradient Boosting se expresa como:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

donde $f_M(x)$ es la predicción final, $T(x; \Theta_m)$ representa el modelo individual en la m-ésima iteración y Θ_m son los parámetros óptimos para ese modelo.

La optimización de los parámetros se realiza mediante el ajuste de cada nuevo modelo a los residuos del modelo anterior, utilizando una función de pérdida específica. A través de este enfoque iterativo, Gradient Boosting mejora gradualmente la capacidad predictiva del modelo final.

Consideremos el caso de regresion y la perdida cuadratica. En este caso queremos aprender un modelo f que prediga valores de la forma $\hat{y} = f(x)$ minimizando la RSS presentada anteriormente, i.e $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Ahora consideremos un algoritmo de boosting con M iteraciones. En cada etapa $m, 1 \leq m \leq M$, supongamos que tenemos un modelo imperfecto f_m (para m suficientemente bajo el modelo puede retornar $\hat{y}_i = \bar{y}$). Para poder mejorar f_m, nuestro algoritmo debe agregar un nuevo estimador $h_m(x_i)$. Por lo que

$$f_{m+1}(x_i) = f_m(x_i) + h_m(x_i) = y_i,$$

o de manera equivalente $h_m(x) = y_i - f_m(x_i)$- Por lo que gradient boosting ajusta h_m al residuo $y_i - f_m(x_i)$. Como en otros métodos de boosting, $f_{m+1}(x)$ intenta corregir los errores de su predecesor. Esta idea se puede generalizar a otras funciones de perdida si observamos que los residuos $h_m(x_i)$ son proporcionales al gradiente negativo de la perdida cuadrática L_2 con respecto a f, es decir

$$L_2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$
$$-\frac{\partial L_2}{\partial f(x_i)} = \frac{2}{n}(y_i - f(x_i)) = \frac{2}{n}h_m(x_i)$$

Por lo que el gradient boosting puede especializarse a algún algoritmo de descenso por gradiente. Generalizar esto conlleva insertar una función de perdida diferente y su gradiente. Ilustramos el procedimiento de boosting en la figura 3, en donde mostramos

iteraciones del método, comenzando con 1 solo arbol y aumentando hasta tener 1000 arboles. La función que estamos intentando aproximar es la función Doppler $f(x) = \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.05}\right)$, la cual es usada al poner a prueba métodos de aproximación de funciones y en estadística no paramétrica. Los puntos azules son generados de la función Doppler y la curva en rojo denota nuestra aproximación. A medida aumentamos el numero de arboles, la aproximación mejora.

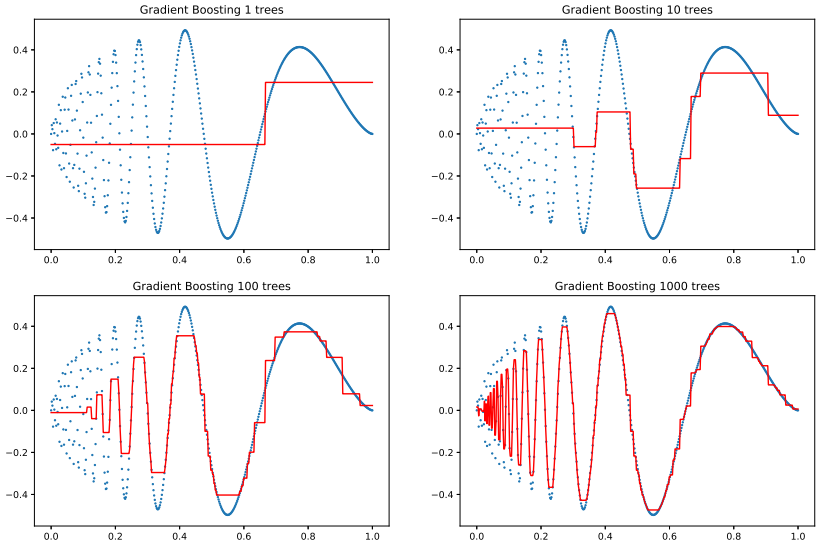

Figura 3: Ilustración del método de boosting.

En general para una funcion de perdida L y una clase de regresores o clasificadores debiles $\mathcal{H}$, tenemos

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma), f_m(x) = f_{m-1}(x) + \left(\arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + h_m(x_i)) \right] \right)(x),$$

para $m \geq 1$, en donde $h_m \in \mathcal{H}$ es una función regresor o clasificador débil. El problema anterior no es factible computacionalmente, por lo que se resuelve una versión simplificada del mismo. La idea es aplicar descenso por gradiente. Específicamente una versión especial: el descenso por gradiente funcional.

Implementaciones de Gradient Boosting

A continuación se muestra el algoritmo de aumento de árboles de gradiente, la cual es una técnica poderosa en el aprendizaje automático que combina la capacidad predictiva de múltiples árboles de decisión débiles para formar un modelo robusto y preciso. Este algoritmo se utiliza tanto en problemas de regresión como de clasificación y se basa en la construcción iterativa de árboles para corregir los errores del modelo anterior. d

1. Inicializar $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. Para $m = 1$ hasta M:
   a) Para $i = 1, 2, \ldots, n$ calcular los seudo residuales $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$.
   b) Ajustar un árbol de regresión a los objetivos r_{im} dando regiones terminales R_{jm}, $j = 1, 2, \ldots, J_m$.
   c) Para $j = 1, 2, \ldots, J_m$ calcular $\gamma_{jm} = \arg\min_\gamma \sum_{xi \in R_{jm}} L(y_i, f_{m-1}(xi) + \gamma)$.
   d) Actualizar $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Salida $\hat{f}(x) = f_M(x)$.

6. Conclusiones

En conclusión, el análisis detallado de la introducción al aprendizaje estadístico, con un enfoque central en el boosting con árboles, destaca la relevancia y el potencial de esta metodología en el campo del aprendizaje automático. El boosting con árboles se presenta como una herramienta poderosa para mejorar la capacidad predictiva al combinar múltiples clasificadores débiles en un modelo fuerte.

La capacidad de superar las limitaciones inherentes a los modelos individuales, capturando patrones más complejos, se muestra como uno de los principales puntos fuertes de esta técnica. Sin embargo, reconocemos la necesidad de abordar desafíos pendientes, como la interpretabilidad de los modelos y consideraciones relacionadas con la escalabilidad.

En resumen, resaltamos la importancia del boosting con árboles como un enfoque valioso en el aprendizaje estadístico, proporcionando una visión inicial de sus beneficios y desafíos. Este trabajo introductorio sienta las bases para investigaciones y aplicaciones más profundas en el campo, mostrando el camino hacia un entendimiento más completo y la optimización continua de estas técnicas en el aprendizaje automático.

Referencias

James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning: with Applications in Python*. Springer Texts in Statistics. Cham: Springer International Publishing.
URL https://link.springer.com/10.1007/978-3-031-38747-0

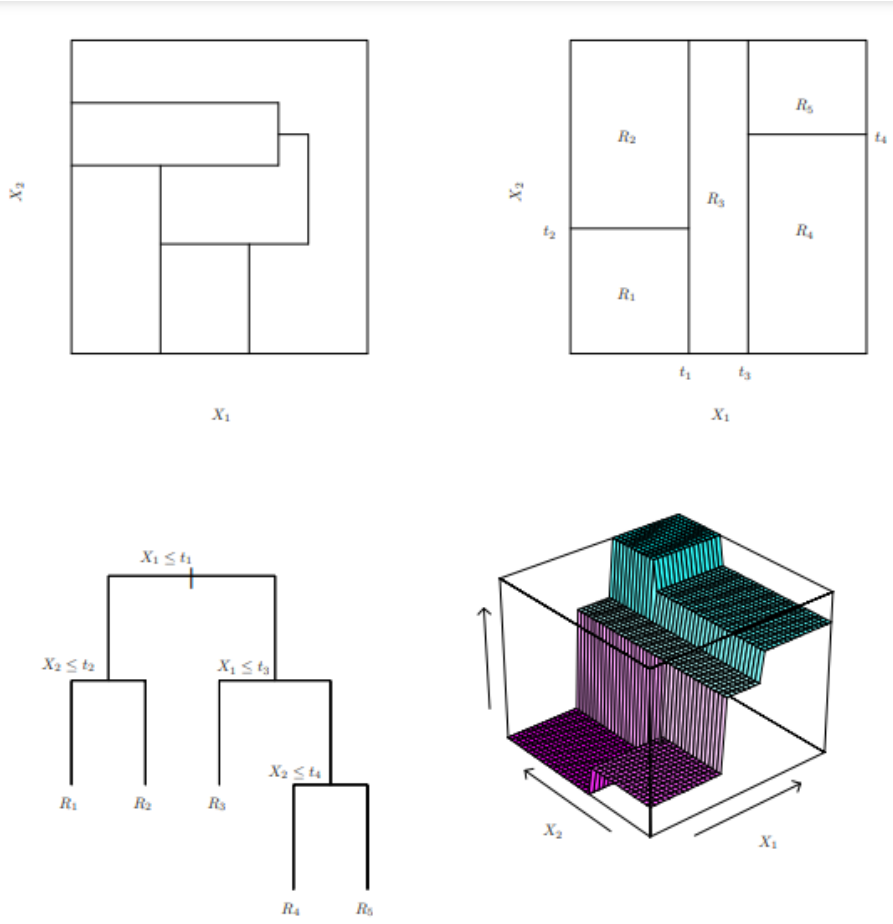

Figura 1: Arriba a la izquierda: Una partición del espacio de características bidimensional que no podría resultar de la división binaria recursiva. Arriba a la derecha: El resultado de la división binaria recursiva en un ejemplo bidimensional. Abajo a la izquierda: Un árbol correspondiente a la partición en el panel superior derecho. Abajo a la derecha: Una representación gráfica en perspectiva de la superficie de predicción correspondiente a ese árbol.

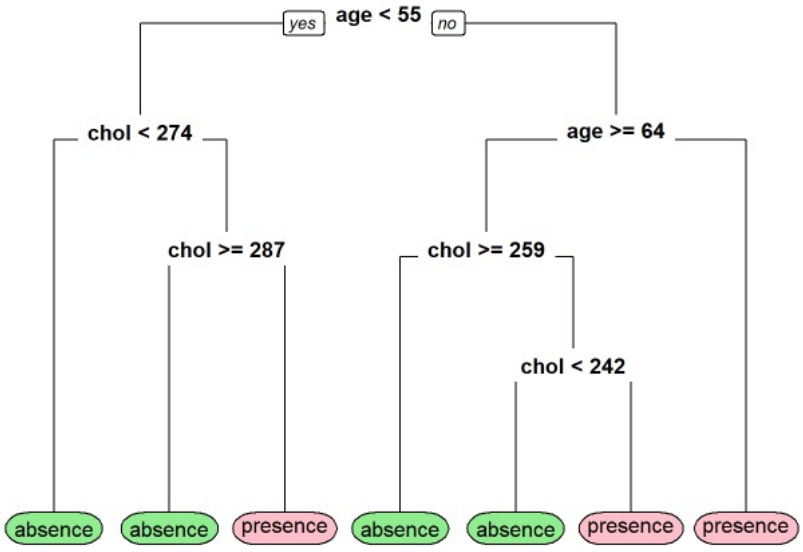

Figura 2: Ejemplo de árbol de clasificación.