Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática e Instituto de Cálculo

Trabajo Final Integrador

Tesis presentada para optar al título de Magister en
Estadística Matemática de la Universidad de Buenos Aires.

Autor: Martín Gabriel Cargnel
Directora: Dra. Daniela Rodriguez

Febrero, 2024

Trabajo final integrador

Resumen

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Índice general

1. Introduction

Dentro del modelado estadístico se destacan dos objetivos principales: Predecir e Explicar. Esta distinción puede parecer filosófica en un primer momento, pero es de interés estudiarla dado que genera diferencias metodológicas importantes que llevan a los investigadores y practicantes a optar por técnicas y modelos muy distintos para alcanzar sus objetivos.

A continuación, se estudian las dos principales corrientes descritas en [Shmueli, 2010].

1.1. Modelado Explicativo

Cuando el interés es interpretar o explicar, los estudios se centran en entender el efecto que tienen variables explicativas (X) en una variable de respuesta (Y). Este tipo de estudios tratan de emplear modelos más simples para que sea más sencillo entender cómo cada covariable se relaciona con la variable de respuesta. En estos casos, típicamente se utilizan regresiones lineales por su simplicidad.

Generalmente esta clase de modelado se utiliza mucho en las ciencias sociales, existe una teoría muy fuerte por detrás que sustenta los modelos. En definitiva, se puede considerar que los modelos se utilizan con el objetivo de testear las relaciones causales entre las variables.

Finalmente, pese a que la capacidad de predicción, medida utilizando alguna métrica como error cuadrático medio o error de clasificación, pasa a un segundo plano es de interés encontrar modelos que pasen un cierto umbral de precisión.

1.2. Modelado predictivo

Cuando el objetivo es predecir el foco cambia y dado que se busca predecir nuevos valores de la variable dependiente (Y), dadas ciertas variables explicativas ó predictoras (X). En estos casos se suelen utilizar modelos más complejos, como ensambles de árboles ó redes neuronales, normalmente asociados al aprendizaje automático.

El procesamiento del lenguaje natural es un caso claro donde el objetivo es predecir. Para esta disciplina se emplean redes neuronales con múltiples capas que hacen muy difícil su interpretación que, de todas formas, pasa a un segundo plano. Como puede esperarse cuando el objetivo es predecir la teoría tampoco juega un rol tan importante.

1.3. Trade off entre explicar y predecir

Según lo descrito anteriormente, parece haber un *trade-off* entre predecir y explicar. Es decir, que si el objetivo es predecir se deberían optar por modelos más complejos poco interpretables pero muy flexibles para minimizar lo más posible una métrica de error. Por otra parte si el objetivo es explicar, deberíamos aceptar modelos con baja capacidad predictiva, dado que para aumentar el poder predictivo necesariamente deberíamos complejizar demasiado los modelos, al punto de no poder interpretarlos.

Entre *trade-off* también puede entenderse (segundo libro Molnar) como que en aplicaciones donde se busca predecir el modelo se adapta a los datos, mientras que cuando se busca explicar los datos (o la realidad) se simplifica para ser explicada por el modelo.

Por lo que es de interés estudiar técnicas que permitan utilizar modelos complejos para explicar el efecto de X sobre Y.

1.4. Aprendizaje automático interpretable

Generalmente los modelos de aprendizaje automático destacan por su capacidad predictiva. Sin embargo, muchas veces se consideran cajas negras dado que, debido a su complejidad, es muy difícil responder preguntas como: ¿Qué variables son más importantes? ó ¿Cuál es el efecto de las variables explicativas en la variable de respuesta?

Sin embargo, existe una batería de técnicas que permiten responder estas preguntas e intentar mitigar el *trade-off* descrito anteriormente. Las técnicas que se estudiarán en este trabajo pueden ser consideradas agnósticas al modelo, es decir se asume que el modelo es una caja negra y se aplican luego técnicas para interpretarlo. Esto se utiliza dado que la cantidad de modelos que pueden interpretarse 'directamente' es limitada. Pero al usar estas técnicas se evita tener que restringir tanto la cantidad de modelos a utilizar, pudiendo emplear, por ejemplo, modelos más flexibles.

1.5. Ensambles de árboles como candidatos

En este trabajo se decidió estudiar los ensambles de árboles como modelo ya que, junto con técnicas de interpretabilidad, permiten mitigar el *trade-off* entre interpretabilidad y poder predictivo. Este modelo se eligió sobre otras alternativas debido a su alto poder predictivo, lo que lo hace muy popular para tareas de predicción. Su popularidad y buen rendimiento resultaron en la disponibilidad de numerosos paquetes en varios lenguajes de programación, lo que facilita su implementación de manera sencilla y eficiente.

Además, dado que los ensambles de árboles se basan en utilizar múltiples arboles de decisión para la predicción, mantienen algunas ventajas de árboles individuales, como la capacidad de modelar las interacciones entre las variables.

Cabe destacar que si bien estos modelos pueden utilizarse tanto para clasificación como para regresión, en este trabajo se solo se estudiará la metodología para

problemas de regresión. De todas formas, todos los resultados a los que se lleguen son fácilmente adaptables a problemas de generalización.

1.6. Estructura del trabajo

El trabajo se estructura de la siguiente manera: en el capítulo 2 se hace una revisión de la teoría detrás de los árboles de regresión y clasificación, mencionando el algoritmo que se usa para podarlos, cómo se comportan cuando las variables están correlacionadas y ante la presencia de *outliers*. Terminando con algunas de las limitaciones que llevaron al desarrollo de técnicas más avanzadas.

En el capitulo 3 se estudian los ensambles de árboles y cómo estos alivian algunas de las limitaciones de los arboles individuales. En particular se profundiza sobre las técnicas *Bagging* y *Boosting*.

Luego, en el capítulo 4, se revisan tres técnicas que permiten interpretar los ensambles de árboles descritos anteriormente.

Finalmente se presenta una aplicación de todo lo descrito anteriormente en la capítulo 5, seguido por las conclusiones en la capítulo 6.

1.7. Notación

Sea P_{XY} la distribución conjunta inducida por el proceso generador de datos, donde X es una variable aleatoria p-dimensional e Y es una variable aleatoria unidimensional. El conjunto de datos se denota como $D_n = (x(1), y(1)), ..., (x(n), y(n))$, donde $x(i) \in R_p$ e $y(i) \in R$ para $i \in 1, ..., n$. Para las secciones 2, 3 y 4 se utilizan datos simulados con distribución $\mathcal{U}(-1, 1)$, con $p = 10$ y $n = 1000$. El vector y fue creado de forma tal que no sea lineal y ocurran interacciones entre las variables, además el término de error ϵ se asumió $\mathcal{N}(0, 1)$. Mientras que en la sección 5 se emplean tres conjuntos de datos reales.

El mapeo verdadero entre características y variable objetivo se denota como $f(X) = E[Y|X = x]$. Para cualquier modelo, denotamos $f : X \to Y$ como el modelo teórico y $\hat{f} : X \to Y$ como el modelo ajustado sobre D_n, el cual se obtiene minimizando una función de pérdida $L : Y \times R_p \to R_+^0$. A lo largo del trabajo se estudiarán distintos tipos de modelos, denotados como f^{tree} para árboles de regresión, f^{rf} para Random Forest y f^{gb} para Gradient Boosting.

2. CART

En este capítulo se hará una revisión sobre árboles de regresión y clasificación (CART). Los CART entran dentro de la categoría de análisis supervisado no paramétrico, por lo que tienen una gran flexibilidad. Otra de las ventajas de estos algoritmos es su interpretabilidad, dado que es muy sencillo entender el efecto de cada variable explicativa en la variable dependiente simplemente siguiendo la estructura del árbol. Además, permiten entender naturalmente las interacciones entre dichas variables.

En la fig-arbol-decision se muestra un árbol de decisión cuyo objetivo es predecir el índice de progresión de la diabetes utilizando la edad y el índice de masa corporal (BMI). El árbol se interpreta de arriba hacia abajo, lo que indica que el índice de masa corporal es la variable más relevante, ya que aparece en la primera división. Se observa que niveles más altos de BMI están asociados con un aumento en el índice de progresión de la diabetes.

Por otro lado, la edad también influye en la variable dependiente, pero su efecto se observa después de las primeras condiciones basadas en el BMI. Por ejemplo, para pacientes con un BMI menor o igual a 24.35 y una edad menor o igual a 57.5, el índice de progresión de la diabetes es 99.69. Esto destaca (1) la facilidad con la que se pueden interpretar visualmente los árboles de decisión y (2) cómo estos modelos representan de manera natural las interacciones entre diferentes variables.

2.1. Elementos que componen un árbol

Antes de pasar a la construcción de los árboles, es de interés enfatizar que estos modelos se leen de arriba hacia abajo y definir ciertos elementos que los componen. En primer lugar definimos los nodos como los puntos en los cuales se dividen las observaciones. A su vez pueden clasificarse como:

- Raíz: el primer nodo que contiene todas las observaciones y sobre el cual se hace la primer partición.
- Internos: son los puntos en los cuales se dividieron las observaciones.
- Terminales u hojas: presentan las predicciones finales del modelo, son nodos que no tienen descendientes inmediatos.

Luego, como se ve en la fig-arbol-decision los nodos definidos anteriormente están unidos por flechas que llamaremos ramas y tienen el objetivo de conectan los distintos nodos del árbol.

Habiendo definido estos elementos podemos pasar a estudiar cómo construir los árboles.

2.2. Construyendo árboles de regresión

Para construir los árboles necesitamos particionar secuencialmente el conjunto de datos X que tenemos. El objetivo final es construir un modelo regresión que nos permita estimar el valor de una variable dependiente Y. Para ello necesitamos encontrar:

1. Encontrar un criterio para particionar los datos.
2. Una regla para determinar cuando un nodo es terminal.
3. Una forma de asignar un valor a las predicciones de los nodos terminales.

2.2.1. Criterio para particionar los datos

Dado que necesitamos crear $R_1, R_2, ..., R_J$ regiones distintas y disjuntas que, a priori, podrían tener cualquier forma. Sin embargo, por simplicidad tendrán forma rectangular. Además, dado que se trata de un algoritmo de regresión supervisado tiene el objetivo de predecir con el menor error posible, por lo que se suele utilizar el error cuadrático medio para medir el error:

$$MSE = \frac{1}{n}\sum_{n=1}^{N}(Y - \hat{Y})^2 \tag{2.1}$$

con n la cantidad de observaciones, Y los valores reales de la variable dependiente y $\hat{Y}$ las predicciones realizadas por nuestro modelo con la muestra X.

En definitiva, si lo que buscamos son regiones que minimicen el error cuadrático medio podemos expresarlo como

$$MSE(T) = \frac{1}{n}\sum_{j=1}^{J}\sum_{i:\in R_J}(Y(i) - \hat{Y}_{R_j})^2 \tag{2.2}$$

con el árbol T, n el número total de observaciones en la muestra, J el número de regiones R, $Y(i)$ el valor real de las observaciones i que cayeron se encuentran en la región R_j y $\hat{Y}_{R_j}$ la predicción de nuestro modelo.

Sin embargo, calcular esta expresión es computacionalmente imposible dado que implica buscar todas las posibles particiones en todas las posibles regiones que se pueden generar con la muestra. A modo de ejemplo esto significa que si tenemos p variables y cada variable tiene s puntos en los que podría ser particionada, el número total de posibles particionas de todas las variables en cada nodo sería $m * p$. Dado que cada partición es binaria, el número de nodos crece exponencialmente con la profundidad del árbol d. Es decir que para calcular la cantidad total de particiones posibles (TP) para todo el árbol tendríamos que calcular

$$TP = \sum_{d=0}^{D} m * p * 2^d \tag{2.3}$$

que crece exponencialmente por lo que se hace computacionalmente muy intensiva.

Es por ello que se utiliza el algoritmo conocido como división binaria recursiva, este algoritmo, que va de arriba hacia abajo en el árbol, se considera greedy dado en cada paso solo mira el valor óptimo en ese momento, por lo que hace menos cálculos aunque puede que no llegue a la solución óptima. En definitiva el algoritmo consiste en considerar todas las $X_1, X_2, ..., X_p$ y todos los posibles valores en los cuales se podría particionar (s) para quedarnos con el que minimice el error cuadrático medio en ese paso. En definitiva buscamos

$$MSE(T) = \frac{1}{n}\sum_{r=1}^{R}\sum_{i:x_i \in R_r(j,s)}(Y(i) - \hat{Y}_{R_r})^2 \tag{2.4}$$

con n el número de observaciones, R el número total de regiones después de aplicar todas las divisiones, $R_r(j,s)$La región r creada al dividir el predictor X_j en el punto de corte s, siendo X_j el predictor que minimiza el error, $Y(i)$ el valor real de las observaciones i dentro de la región R_r y $\hat{Y}_{R_r}$ los valores predichos para todas las observaciones i en la región R_r

Este algoritmo ya no crece de manera exponencial como el anterior, por lo que es más sencillo de calcular que el anterior.

2.2.2. Definir cuando un nodo es terminal

Definir cuando un nodo es terminal es equivalente a pensar cuando un nodo

2.2.3. Asignar valores a nodos terminales

Dado que la función a optimizar es el error cuadrático medio, se puede probar que la media muestral será la constante que minimice ese error.

Partimos de que queremos c que minimice

$$MSE(c) = \frac{1}{n}\sum_{i=1}^{n}(y_i - c)^2 \tag{2.5}$$

para ello primero derivamos con respecto de c

$$\begin{aligned}
\frac{d}{dc}MSE(c) &= \frac{d}{dc}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - c)^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{d}{dc}\left[(y_i - c)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}2(y_i - c)(-1) \\
&= -\frac{2}{n}\sum_{i=1}^{n}(y_i - c)
\end{aligned} \tag{2.6}$$

y luego igualamos a cero:

$$\begin{aligned} -\frac{2}{n}\sum_{i=1}^{n}(y_i - c) &= 0 \\ \sum_{i=1}^{n}(y_i - c) &= 0 \\ \sum_{i=1}^{n} y_i - nc &= 0 \\ \sum_{i=1}^{n} y_i &= nc \\ \frac{1}{n}\sum_{i=1}^{n} y_i &= c \end{aligned} \tag{2.7}$$

Por lo que los valores que tendrán los nodos terminales (predicciones) serán simplemente la media de las observaciones que están en ese nodo.

2.3. Poda de árboles

Construir un árbol con demasiadas particiones puede llevar a un sobre ajuste. Básicamente porque si pensamos en el árbol más grande que se puede construir con una muestra, el mismo tendría un nodo terminal por cada observación, lo cual llevaría a un error muy bajo en la muestra de entrenamiento, pero una capacidad de generalización muy mala.

Para evitar esto, buscaremos poner un límite a cuanto puede crecer el árbol, para ello existen múltiples alternativas como fijar cotas a la profundidad del árbol, al número de observaciones que pueden quedar en la misma hoja o bien hacer crecer un árbol muy grande y luego "podarlo"para obtener árboles más pequeños que podemos comparar con validación cruzada. Este último procedimiento será descrito en el trabajo.

Antes de describir el procedimiento es importante definir T_0 como un árbol maximal y T' sería un subárbol podado de T. Para que un árbol sea considerado sub-árbol de otro tiene que tener $T' = T_0 - T_t$, donde T_t es una rama (subárbol) que se elimina de T_0 para obtener T'.

El problema de querer encontrar sub-árboles es que pueden ser demasiados, haciendo que sea computacionalmente demasiado exigente. Es por ello que se utiliza cost complexity pruning para trabajar un pequeño número de subárboles. A continuación vemos la ecuación:

$$\sum_{m=1}^{|T|}\sum_{i:x_i \in R_r(j,s)} (Y(i) - \hat{Y}_{R_r})^2 + \alpha|T| \tag{2.8}$$

con α que sirve para regular la intensidad de la regularización y $|T|$ el número de nodos terminales del árbol T. Esta expresión permite obtener una secuencia ordenada de subárboles que podemos evaluar por medio de validación cruzada. Queda claro que aumentar α se generan árboles más pequeños.

Se puede ver en la @fig-arbol-alpha como a medida que aumentan los valores de α inicialmente el error cuadrático medio disminuye, pero luego empieza a subir debido al sobre ajuste.

2.4. Outliers y Limitaciones

2.4.1. Outliers

Es sabido que las regresiones lineales son muy sensibles a *outliers*. Sin embargo, como se ve en Brieman los valores atipicos terminan siendo aislados en pequeños nodos. Esta capacidad de aislar los *outliers* permite hacer el modelo menos sensible a datos atípicos.

2.4.2. Limitaciones

Dado que este trabajo se centra en la interpretación de modelos, la principal limitación que se destaca es la falta de estabilidad. Es decir que la varianza de los árboles genera que las particiones puedan ser muy distintas ante cambios en los datos. Esto lógicamente atenta contra la interpretación.

Otras limitaciones son la falta de suavidad y las dificultades que tienen para captar una estructura aditiva en los datos.

3. Ensambles de árboles

Como se mencionó anteriormente, dos de los problemas principales de lo árboles de decisión son la baja capacidad predictiva y la inestabilidad. Por lo que ese capítulo se estudian dos métodos para evitar estos problemas, en particular *boosting* y *bagging* con dos implementaciones clásicas como *Random Forest* y *Gradient Boosting Machines*.

Los ensambles mejoran la capacidad predictiva y reducen la variabilidad. La idea general es agrupar múltiples modelos con el objetivo de tener un único modelo con mejor capacidad predictiva.

3.1. *Bagging*

Uno de los grandes problemas de los árboles decisión es la varianza que tienen, por lo que parece intuitivo aplicar técnicas que permitan promediar métodos estadísticos para reducir la varianza manteniendo el sesgo bajo.

La idea de *Bagging* es calcular $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ usando B muestras boosrtapeadas y promediar las predicciones para reducir la varianza. Es decir

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x) \tag{3.1}$$

La ventaja de utilizar *Bagging* para predecir es que no hay que preocuparse por la profundidad de los árboles, dado que los árboles individuales tendrán mucha poco sesgo y la varianza será reducida al tomar promedios. Además, si bien *Bagging* puede utilizarse en múltiples métodos estadísticos, es particularmente útil para árboles de decisión dado que permite reducir la varianza sin aumentar el sesgo.

Bagging reduce la varianza sin aumentar el sesgo. Partiendo de las predicciones

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x) \tag{3.2}$$

calcular el error viene dado por

$$MSE = E[(y - \hat{f}_{bag}(x))^2] = \text{Bias}^2 + \text{Varianza} + \sigma^2 \tag{3.3}$$

ahora nos concentramos en cómo la varianza afecta al error

$$Var(\hat{f}_{bag}(x)) = Var(\frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x)) = \frac{1}{B^2} Var(\sum_{b=1}^{B} \hat{f}^b(x)) \tag{3.4}$$

asumiendo que los $\hat{f}^b(x)$ están identicamente distribuidos, se puede reescribir:

$$Var(\sum_{b=1}^{B} \hat{f}^b(x)) = \sum_{b=1}^{B} Var(\hat{f}^b(x)) + \sum_{i=1}^{B}\sum_{j\neq i}^{B} cov(\hat{f}^i(x), \hat{f}^j(x)) \tag{3.5}$$

ahora asumiendo varianzas iguales ($Var(\hat{f}^b(x)) = \sigma^2$) y correlaciones identicas (ρ) para cada par de árboles:

$$Var(\sum_{b=1}^{B} \hat{f}^b(x)) = B\sigma^2 + B(B-1)\rho\sigma^2 \tag{3.6}$$

volviendo a la varianza total:

$$\begin{aligned} Var(\hat{f}_{bag}(x)) &= \frac{1}{B^2}(B\sigma^2 + B(B-1)\rho\sigma^2) \\ &= \sigma^2(\frac{1}{B} + \frac{\rho B(B-1)}{B^2}) \end{aligned} \tag{3.7}$$

Por lo tanto:

- Cuando $B \to \infty$, $\frac{1}{B} \to 0$ y la varianza se aproxima a $\rho\sigma^2$
- La reducción de varianza depende tanto de B como de ρ
- Como $\rho < 1$, la varianza final ($\rho\sigma^2$) es menor que la varianza original (σ^2)

Bagging tiene una forma muy natural de calcular el error, básicamente se basa en el hecho de que cuando se hace un muestra *boostrap* solamente se toma en cuenta aproximadamente el 63 % de los datos.

Para probar que *Bootstrap* solo ve aproximadamente 2/3 de los datos, partimos de la probabilidad de No ser seleccionado en una muestra:

$$P(\text{No ser seleccionado en una muestra}) = 1 - \frac{1}{n} \tag{3.8}$$

Luego, como se toman n muestras con reemplazo, la probabilidad de no ser seleccionado ninguna muestra

$$P(\text{No ser seleccionado en ninguna muestra}) = (1 - \frac{1}{n})^n \tag{3.9}$$

cuando $n \to \infty$:

$$\lim_{n\to\infty}(1 - \frac{1}{n})^n = \frac{1}{e} \approx 0{,}368 \tag{3.10}$$

- La probabilidad de ser seleccionado al menos una vez es $1 - \frac{1}{e} \approx 0{,}632$
- Aproximadamente el 63.2 % de las observaciones serán incluidas en cada muestra bootstrap
- El restante 36.8 % de las observaciones quedan fuera de la muestra (out-of-bag)

Este resultado permite estimar el error usando las observaciones *out-of-bag* como muestra de testeo. Con un B suficientemente grande, este error es asintóticamente equivalente a *leave-one-out cross validation*, lo cual es particularmente útil cuando se cuenta con una muestra grande donde el *cross validation* tradicional sería computacionalmente costoso.

Una de las limitaciones de *Bagging* es que, al utilizar todas las variables independientes en cada árbol del modelo, puede no reducir significativamente la varianza cuando estas variables están altamente correlacionadas. Esto se debe a que los árboles generados tienden a ser similares, lo que limita el beneficio del promediado de las predicciones. Este punto puede verse en la demostración anterior, donde a menor ρ menor es el error.

3.1.1. Random Forest

Random Forest propone una mejora con respecto a *Bagging* al tener en cuenta solo un sub-conjunto de las covariables para cada *split*. Esto permite que los árboles sean distintos entre ellos, haciendo que los árboles estén menos correlacionados entre ellos y por lo tanto reduciendo más el error.

El parámetro que define la cantidad de variables que se tiene cuenta en cada *split* se denomina m. Cabe destacar que un *Random Forest* con $m = p$, siendo p la cantidad de covariables, es *bagging*. Por lo que *Random Forest* puede considerarse un caso particular de *bagging*. Mas formalmente se puede pensar el predictor *Random Forest* como

$$\hat{f}_{\text{rf}}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \Theta_b) \tag{3.11}$$

Donde $(T(x; \Theta_b))$ representa el valor predicho por el b-ésimo árbol de decisión para una observación x, construido a partir de una muestra *bootstrap* y parámetros aleatorios Θ_b.

Es interesante notar que, si tomamos si $B \to \infty$, es decir cuando se promedian muchos árboles, por Ley de los Grandes Números se puede probar que:

$$\lim_{B \to \infty} \hat{f}_{\text{rf}}(x) = E[T(x; \Theta_b)] \tag{3.12}$$

Esto significa que, con suficientes árboles, la predicción de *Random Forest* se aproxima al valor esperado de un único árbol. La demostración formal y los detalles pueden verse en CITAR RF, sin embargo de manera intuitiva puede verse en la figura @fig-rf-predictions como a medida que aumenta la cantidad de árboles la variabilidad de las predicciones disminuye.

3.2. Boosting

A diferencia de *Bagging*, *Boosting* no promedia predicciones, sino que construye un modelo iterativamente. Donde cada modelo se construye con el objetivo de corregir el error cometido por el modelo anterior. A continuación se presenta un ejemplo de cómo funciona el algoritmo de *Gradient Boosting Machines (GBM)* propuesto por [Friedman, 2001].

Como en cualquier problema de aprendizaje supervisado, el objetivo es estimar $\hat{f}(x)$ a partir de un conjunto de datos (X, Y). En este caso, el conjunto de datos es una muestra i.i.d. de (X, Y) de tamaño n:

$$(X, Y) = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{3.13}$$

Donde x_i es un vector de covariables y y_i es la variable de respuesta, en este caso, continua. Por lo que se trata de un problema de regresión.

El algoritmo de *Gradient Boosting Machines* propone un método para encontrar un aproximación de la función $f(x)$ que minimiza una función de pérdida $L(y, f(x))$:

$$\hat{f}(x) = \arg\min_f \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) \tag{3.14}$$

En definitiva, se ve que la forma de aproximar la función $f(x)$ es a través de una combinación lineal de modelos base $h(x, \theta)$:

$$\hat{f}(x) = \sum_{b=1}^{B} \gamma_b h_b(x, \theta_b) \tag{3.15}$$

3.2.1. Algoritmo

El algoritmo de *Gradient Boosting Machines* requiere un conjunto de datos (X, Y), una función de pérdida $L(y, f(x))$, un número de iteraciones B y un modelo base $h(x, \theta)$. Con esto, el algoritmo es el siguiente:

1. Inicializar $\hat{f}_0(x)$ con una constante
2. Para cada iteración $b = 1, 2, ..., B$:
   - Calcular el gradiente de la función de pérdida y utilizar su valor negativo.
   - Ajustar el modelo base $h_b(x, \theta_b)$ a los datos $(X, g_b(X))$
   - Encontrar el mejor gradiente descend step-size γ_b:

$$\gamma_b = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \hat{f}_{b-1}(x_i) + \gamma h_b(x_i, \theta_b)) \tag{3.16}$$

   - Actualizar la aproximación:

$$\hat{f}_b(x) = \hat{f}_{b-1}(x) + \gamma_b h_b(x, \theta_b) \tag{3.17}$$

3. Devolver el modelo final.

3.2.2. Funciones de pérdida

Dependiendo el problema y los datos con los que se esté trabajando, se pueden utilizar distintas funciones de pérdida. A continuación se presentan algunas de las más comunes para problemas de regresión. Es decir, cuando la variable de respuesta (y) es continua.

Error Cuadrático Medio (MSE)

Una de las funciones de pérdida más comunes es el error cuadrático medio (MSE), que se define como:

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2 \tag{3.18}$$

Una de las ventajas que tienen este tipo de funciones de pérdida es que son diferenciables, lo cual facilita el cálculo del gradiente. Que simplemente es:

$$\begin{aligned} \frac{\partial L(y, f(x))}{\partial f(x)} &= \frac{\partial}{\partial f(x)} \frac{1}{2}(y - f(x))^2 \\ &= \frac{1}{2} 2(y - f(x)) \frac{\partial (y - f(x))}{\partial f(x)} \\ &= -(y - f(x)) = F(x) - y \end{aligned} \tag{3.19}$$

Luego, como el gradiente indica la dirección de máximo crecimiento, el negativo del gradiente indica la dirección de máximo decrecimiento. Lo cual es lo que se busca en el algoritmo de *boosting*, es decir, minimizar la función de pérdida. Por lo que el gradiente de la función de pérdida es:

$$-g_b(x) = y - F(x) \tag{3.20}$$

Lo cual es simplemente el residuo.

Alternativas robustas

Como se mencionó anteriormente, una de las ventajas de utilizar la función de pérdida MSE es que es diferenciable. Sin embargo, el hecho de que penalice los errores grandes puede ser una desventaja cuando se trabaja con *outliers*. Por lo que también se puede utilizar la función de pérdida absoluta (MAE) que se define como:

$$L(y, f(x)) = |y - f(x)| \tag{3.21}$$

ó la función de pérdida Huber que se define como:

$$L(y, f(x)) = \begin{cases} (y - f(x))^2 & \text{si } |y - f(x)| \leq \delta \\ 2\delta|y - f(x)| - 2\delta^2 & \text{si } |y - f(x)| > \delta \end{cases} \tag{3.22}$$

A continuación, en la fig-loss-functions, se presentan las funciones de pérdida anteriores. Puede verse que MSE aumenta mucho cuando hay errores grandes, mientras que MAE y Huber son más robustas a los *outliers*. Además, Huber se comporta similar a MSE cuando el error es pequeño, pero al aumentar el error, comienza a penalizar menos.

3.2.3. Modelos base

Si bien el algoritmo de *Boosting* puede utilizar múltiples modelos base, en este trabajo se expondrán únicamente los modelos base de árboles de decisión.

Como se mencionó anteriormente, una de las ventajas de los árboles de decisión es poder capturar interacciones entre las covariables. Esta característica puede ser

regulada mediante un hiperparámetro que indica la profundidad máxima de los árboles. En particular, la literatura sugiere utilizar árboles de poca profundidad en este tipo de modelos. Siendo los *stumps* los árboles de profundidad 1 los más utilizados.

En la fig-gbm-depth se puede ver el efecto de la profundidad de los árboles en el error del modelo. Puede verse que a medida que aumenta la profundidad del árbol, el error disminuye hasta llegar a aproximadamente 10, luego de lo cual el error se estabiliza.

3.2.4. Regularización

Uno de los principales problemas de los modelos de aprendizaje automático es el sobreajuste. Es decir, que el modelo se ajuste demasiado a los datos de entrenamiento y por lo tanto no generalice bien a nuevos datos. Esto se da porque las funciones que se utilizan son muy flexibles y pueden llegar a ajustarse al conjunto de datos, por lo que existen determinadas técnicas para evitar esto. En este trabajo, se estudiarán tres técnicas: *subsampling*, *shrinkage* y *early stopping*.

Subsampling

La idea intuitiva detrás del *subsampling* es introducir variabilidad en el entrenamiento del modelo. Esto se logra al considerar un subconjunto de las observaciones para cada iteración del algoritmo. El sampleo puede ser con o sin reemplazo. Cabe aclarar que para definir la proporción de la muestra que se utiliza, se utiliza el hyperparámetro *bag fraction* que toma valores entre 0 y 1. Por ejemplo, si se tiene una muestra de 1000 observaciones y se utiliza un *bag fraction* de 0.1, se utilizarán 100 observaciones para cada iteración.

Pese a que esta técnica puede ayudar a en casos donde los *datasets* sean muy grandes y el costo computacional sea alto, al solo considerar una parte de las observaciones el entrenamiento puede ser menos preciso por lo que se debe balancear.

Shrinkage

Shrinkage se utiliza para reducir el efecto de los árboles individuales en el modelo. Intuitivamente, es un hiperparámetro que permite regular cuánto se toman en cuenta los errores de los árboles anteriores y se basa en que es preferible mejorar poco a poco las predicciones.

En definitiva, el shrinkage puede incluirse $\hat{f}_b(x)$ como:

$$\hat{f}_b(x) = \hat{f}_{b-1}(x) + \lambda\gamma_b h_b(x, \theta_b) \tag{3.23}$$

Donde λ es el *shrinkage* que toma valores entre 0 y 1. Lógicamente, a medida que λ se acerca a 0, los incrementos de las predicciones son cada vez más pequeños y por lo tanto el modelo es menos sensible a los errores de los árboles anteriores, alcanzando así una mejor generalización. Además, al elegir un *shrinkage* pequeño, el modelo converge más lento, por lo cual se necesita un mayor número de iteraciones Bpara que el modelo converja al mismo error.

En la fig-shrinkage se puede ver el efecto del shrinkage en el error del modelo tanto en la muestra de entrenamiento como en la muestra de test. Puede verse que en la muestra de entrenamiento el error disminuye más rápido a medida que aumenta

el shrinkage. Sin embargo, en la muestra de testeo puede verse como el *shrinkage* afecta tanto la velocidad de convergencia como la capacidad de generalización del modelo.

Early Stopping

Puede verse también en la fig-shrinkage que el modelo converge a un error mínimo. Sin embargo, puede ser que el modelo se sobreajuste a los datos de entrenamiento y por lo tanto no generalice bien a nuevos datos. Para evitar esto, se puede utilizar el *early stopping* que consiste en detener el entrenamiento cuando el error en la muestra de test deja de disminuir o lo hace de manera muy lenta.

De esta forma, se puede evitar el sobreajuste y reducir el tiempo de entrenamiento de estos modelos. Esto último es particularmente útil cuando se trabaja con muestras grandes.

4. Interpretabilidad

Tal como se mencionó anteriormente, uno de los *trade-off* de los ensambles de árboles es que al agrupar múltiples árboles el modelo en conjunto deja de ser interpretable. Esto puede no suponer un problema si el objetivo es predecir, pero si el objetivo es interpretar el modelo el hecho de trabajar con 'cajas negras' puede ser un problema.

Es por ello que se desarrollaron técnicas para extraer información de modelos de aprendizaje automático. Estas técnicas se aplican una vez entrenado el modelo y permiten entender cómo el mismo utiliza las variables. Pese a que existes distintas técnicas para interpretar modelos, algunas son específicas para ciertos tipos de modelos mientras que otras son específicas, en este trabajo se estudiará como acceder a la importancia de cada covariable y al efecto que tiene cada una de ellas sobre la variable dependiente.

4.1. Importancia de las variables

Con el acceso a información que se tiene actualmente, no es extraño querer incluir múltiples covariables en un mismo modelo. El problema, es que entender cuales de ellas realmente están aportando al mismo no es trivial. Por lo que se desarrollaron múltiples técnicas para evaluarlo, todas con sus argumentos a favor y en contra.

En este trabajo nos centramos en dos: *split gain feature importance* y *permutation feature importance*.

4.1.1. *Split Gain Feature Importance*

Esta técnica fue propuesta por [**?**] y asignar importancias en base a la cantidad de *splits* que cada variable tiene en el árbol. Esta metodología solamente se puede aplicar para árboles de decisión y más formalmente se ve para un solo árbol:

$$\hat{I}_j^2(T) = \sum_{t=1}^{j-1} \hat{i}_t^2 1(v_t = 1) \tag{4.1}$$

done I es la influencia para el nodo j del árbol T, siento v la *splitting variable* que se suma por todos los nodos no terminales.

La eq-split-importance es calculada para un solo árbol. Por lo que cuando se trabaja con ensambles de árboles se debe generalizar a

$$\hat{\mathbf{I}}_j^2 = \frac{1}{B} \sum_{m=1}^{B} \hat{I}_j^2(T_b) \tag{4.2}$$

Intuitivamente, se puede entender como el promedio de los *splits* de cada variable. Sin embargo, esta medida esta basada en heurísticas y no contiene fundamentos estadísticos sólidos. Tal es así que fue criticada por [Zhou and Hooker, 2020] y [Strobl et al., 2008] por dar demasiada influencia a variables correlacionadas y favorecer variables categóricas con muchas categorias.

El problema de las variables correlacionadas radica en que el modelo no puede distinguir entre aquellas variables que realmente aportan a la generación de los datos con las que simplmenente correlacionan con variables influyentes, por lo que termina asignando mayor influencia a variables que no deberían tenerla.

En el caso de las variables categóricas, este método prioriza aquellas variables que son propensas a tener muchos *splits*, sin que esto siginifique que las variables son importantes o estan reduciendo el error.

4.1.2. Permutation Feature Importance

Otra alternativa para calcular las importancias de las variables es uilizar *Permutation feature Importance* (PFI), técnica desarrollada por [Breiman, 2001a] exclusivamente para *Random Forest* que luego fue generalizada para cualquier modelo [Fisher et al., 2019]. Estas técnicas se basan, intuitivamente, en entender cómo cambia la función de pérdida cuando se permutan las variables. Las variables más importantes serán aquellas que más cambien el error del modelo al ser permutadas. Como se ve a continuación, si bien [Fisher et al., 2019] propone métodos para calcular la importancia en modelos de clasificación, se proesentará la adaptación a un modelo de regresión. Por lo que habiendo entrenado un modelo de *random forest* $\hat{f}$ para cada covariable $j \in 1, 2, ..., J$ se:

1. Construye un nuevo conjunto de datos X_j permutando las observaciones de la covariable j y dejando el resto de las observaciones fijas.

2. Entrena un nuevo modelo $\hat{f}_j$ utilizando el nuevo conjunto de datos X_j y calcula el out-of-bag error.

$$E_{OOB}(X_j) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}^{(b)}(X_{i,j}) \right)^2 \tag{4.3}$$

3. Luego se compara el out-of-bag error con el error del modelo original y se calcula la diferencia mediante:

$$PFI(j) = E_{OOB}(X_j) - E_{OOB}(X) \quad \text{ó} \quad PFI(j) = \frac{E_{OOB}(X_j)}{E_{OOB}(X)} \tag{4.4}$$

Sin embargo, como se mencionó anteriormente este método para calcular importancias no es generalizable dado que utiliza los errores out-of-bag, por lo que un modelo que no utilice esta metodología no podría usarla. Sin embargo, es fácilmente extendible a dichos casos. A continuación se presenta la generalización propuesta en [Fisher et al., 2019].

Los autores se basan en la propuesta de [Breiman, 2001a], pero deciden llamar a la medida de importancia *Model Reliance* (MR). Por lo que partiendo de un modelo entrenado $\hat{f}$, una variable de respuesta y, una función de pérdida L y una matriz de diseño X con covariables $j \in 1, 2, ..., J$, para cada covariable j se:

- permuta la covariable j
- Calculamos la pérdida

$$L_{perm}(f) = \mathcal{E}[L(f, (Y, X_1^{perm}, X_2))] \tag{4.5}$$

  donde se permutan los valores de X_1, dejando constantes las demás variables.

- se compara la perdida con el error original

$$MR^j(f) = \frac{L^j_{perm}(f)}{L(f)} \tag{4.6}$$

De esta forma se obtiene cuanto afecta cada covariable a la variable de respuesta. Para calcular L_{perm} los autores proponen dos alternativas,

1. Realizar una permutación por todas las observaciones

$$\hat{e}_{perm}(F)\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i} L\{f, y_j, X_{1[i,.]}, X_{2[i,.]}\} \tag{4.7}$$

   Sin embargo esto puede ser computacionalmente muy peasado, por lo que proponen como alternativa

2. Dividir la muestra a la mitad y reemplazar los valores de X_1 de la primera mitad con los de la segunda y vice versa.

$$\begin{aligned}\hat{e}_{\text{divide}}(f) = \frac{1}{2\lfloor n/2 \rfloor}\sum_{i=1}^{\lfloor n/2 \rfloor}\Bigg[L\Big(f, \Big(y_{[i]}, X_{1\left[i+\lfloor n/2 \rfloor\right]}, X_{2[i]}\Big)\Big) \\ +L\Big(f, \Big(y_{i+\lfloor n/2 \rfloor}, X_{1[i]}, X_{2\left[i+\lfloor n/2 \rfloor\right]}\Big)\Big)\Bigg].\end{aligned} \tag{4.8}$$

4.1.3. Model Class Reliance

Un problema común a la hora de calcular importancias es que el modelo que terminamos usando se decide únicamente en función de una métrica de error. Esto no supondría inconvenientes si las importancias de los modelos fueran parecidas. Sin embargo, suele pasar que para muchos modelos con un poder predictivo similar las importancias de las variables cambien. A continuación se presenta un ejemplo en el cual se entrenaron múltiples *Gradient Boosting Machines* con diferentes combinaciones de hiperparámetros, eligiendo las combinaciones que minimicen el error cuadrático medio. Para el elegir dicha combinación se utilizó validación cruzada con $k = 5$. El conjunto de datos para este ejemplo en particular fue *Abalone* [**?**] donde se busca predecir la edad del Abalone en función de distintas características.

Específicamente, para este ejemplo se entrenaron modelos con cantidad de árboles de 100 a 900 (con incrementos de 100), tasas de aprendizaje de 0.01, 0.05 y 0.1, profundidad máxima de 2, 3, 4 y 5, y funciones de pérdida de tipo pérdida cuadrática, error absoluto y Huber. Siendo un total de 324 modelos entrenados.

En la tbl-display-importances se pueden ver los 15 modelos que menor error tuvieron, junto con los hiperparámetros asociados. Puede verse claramente, como hay múltiples modelos que tienen una performance predictiva similar pero distintos hiperparámetros.

Además, y posiblemente más preocupante, cuando vemos la fig-plot-importances vemos como las importancias de las variables (calculadas mediante *Permutation Feature Importance*) varían entre modelos con un poder predictivo similar. Este mismo fenómeno puede verse también en la fig-plot-heatmap donde para los distintos modelos las variables tienen rankings de importancia muy diferentes. El heatmap muestra claramente cómo una misma variable puede ser considerada la más importante (ranking 1) en un modelo, mientras que en otro puede tener una importancia mucho menor (ranking 5 o 6). Esta inconsistencia en la importancia de las variables entre modelos con rendimiento similar sugiere inestabilidad en la interpretación de los resultados a los que se puede llegar.

El fenómeno observado, donde múltiples modelos con similar capacidad predictiva asignan importancias sustancialmente diferentes a las variables, se conoce como el *Rashomon Effect* en estadística [Breiman, 2001b]. Este término, inspirado en la película de Kurosawa donde un mismo evento es narrado desde perspectivas contradictorias es formalizado en [Fisher et al., 2019] como el çonjunto de Rashomon", definido como la colección de modelos con rendimiento predictivo similar pero con diferentes estructuras internas. Esta multiplicidad de explicaciones válidas representa un desafío fundamental para la interpretabilidad en *machine learning*, ya que sugiere que la importancia de una variable no es una propiedad intrínseca de los datos, sino que depende del modelo específico elegido.

Es por ello que los autores trabajaron sobre la *Model Class Reliance* (MCR), que se basa en para una clase de modelos $\mathcal{F}$, MCR es el rango de todos los posibles valores de MR sobre modelos que tengan un poder predictivo similar al óptimo. Para lo cual, primero se define el çonjunto de Rashomonçomo

$$R(\epsilon)\{f \in \{|e_{orig}(f) \neq e_{orig}(f_{ref}) + epsilon\} \tag{4.9}$$

Este set contiene todos los modes cuya pérdida sea como mucho ϵ peor que el modelo que minimiza el error f_{ref}

Luego se define el intervalo MCR como

$$[MCR^{-}(\epsilon), MCR^{+}(\epsilon)] = [\min_{f \in R(\epsilon)} MR(f), \max_{f \in R(\epsilon)} MR(f)] \tag{4.10}$$

Este intervalo MCR proporciona información valiosa sobre la importancia de una variable en toda la clase de modelos.

Interpretación

Si $MCR^{-}(\epsilon)$ es mayor que 1, significa que todos los modelos "buenos"(dentro del conjunto de Rashomon) dependen de la variable más que del modelo de referencia, lo que sugiere que la variable es consistentemente importante. Por otro lado, si $MCR^{+}(\epsilon)$ es cercano a 1, indica que ningún modelo "bueno"depende fuertemente de esa variable.

La interpretación del MCR nos permite evaluar si la importancia de una variable es consistente a través de diferentes modelos o si es específica de una arquitectura

particular. Un intervalo MCR estrecho sugiere que la importancia de la variable es estable en todos los modelos con buen rendimiento, mientras que un intervalo amplio indica que la importancia varía significativamente dependiendo del modelo elegido. Por ejemplo, pueden ocurrir casos donde para una variable J el $MCR^{-}(\epsilon)$ sea cercano a 1 pero $MCR^{+}(\epsilon)$ no. Esto indicaría existen modelos con buena capacidad predictiva que no dependen de la variable J por lo que dicha variable no sería tan importante.

Estimación

En la práctica para estimar MCR se utiliza un estimador *plug-in*, es decir se calcula $\hat{MR}(f)$ para todos los modelos que integren el conjunto Rashomon empírico $\hat{R}(\epsilon)$.

4.2. Efecto de las variables

Otra técnica útil para interpretar modelos de árboles es mediante *Partial Dependence Plots*. Los mismos también fueron introducidos por [Friedman, 2001], donde los plantea como una técnica para aislar el efecto de una covariable sobre la variable de respuesta, más formalmente lo que se busca estimar es:

$$\bar{f}(X_z) = \mathcal{E}_{x_l}[\hat{f}(x)] = \int \hat{F}(z, x_l)p(x_l)dx_l \tag{4.11}$$

donde $\hat{f}(x)$ es el modelo entrenado, z es el subset de variables a las que queremos estimar el efecto, x_l representa todas las otras predicciones (que se mantienen constantes), $p(x_l)$ es la distribución conjunta de todas las otras variables y $\mathcal{E}_{x_l}[.]$ es la esperanza sobre la distribución x_l

Sin embargo, en la práctica es raro conocer $p(x_l)$ por lo que se utiliza el siguiente procedimiento:

$$\bar{f}(z) \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}(z, x_l^i) \tag{4.12}$$

Cuyo algoritmo sería:

1. Para cada observación i en la muestra, se toman los valores originales de las variables no seleccionadas x_l^i
2. Se utilizan los valores modificados de las variables z en cada x_l^i dentro del modelo $\hat{F}$ para obtener las predicciones $\hat{F}(z, x_l^i)$
3. Se promedian las predicciones para $i = 1, 2, ..., n$

Cabe aclarar que al modificar solamente una variable y dejar el resto como estaban, se esta asumiendo que las variables z son independientes a todo el resto. Algo que puede o no ser cierto dependiendo de la naturaleza de los datos. Además, al cambiar los valores de las variables a evaluar se podrían dar combinaciones poco realistas en la practica.

Con el fin de ejemplificar los gráficos de dependencia parcial se entrenó un modelo de *Gradient Boosting Machine* con 100 árboles con el fin de predecir la edad del

Abalone en función de distintas medidas físicas. Una vez entrenado el modelo, se puede ver en la @fig-plot-impacts el efecto del peso total en la edad.

Sin embargo, como se mencionó anteriormente, los PDPs sufren problemas cuando las variables no son independientes. Para solventar esto, se puede utilizar el método de *Individual Conditional Expectation* (ICE) que consiste en calcular el PDP para cada observación en la muestra.

4.2.1. *Individual Conditional Expectation* (ICE)

Los ICE fueron introducidos por [**?**] como una técnica para estimar el efecto de una covariable sobre la variable de respuesta, más formalmente lo que se busca estimar es:

$$f_i(x_z) = E[Y|X_z = x_z, X_{-z} = x^i_{-z}] \tag{4.13}$$

Es decir, en lugar de calcular el efecto promedio de z sobre y calculando el promedio de $f(z, x_{-z})$ para todas las observaciones, se calcula el efecto de z para cada observación i manteniendo el resto de las variables constantes. De esta manera se puede obtener una estimación del efecto de z para cada observación y no solo un promedio, teniendo así una visión más detallada.

De esta forma, se podrían entender los PDPs como el promedio de los ICE. Dado que mientras el primero se calcula mediante la distribución marginal de las variables, los segundos lo hacen utilizando la distribución condicional.

Cabe destacar que PDP y ICE deberían ser similares cuando las variables son independientes, pero cuando las variables no son independientes, los ICE pueden variar mucho por lo que se recomienda utilizar las dos técnicas.

Como se puede ver en la fig-plot-ice, los ICE son mucho más variables que los PDP, lo que permite tener una visión más detallada del efecto de la variable en cuestión. Sin embargo, se puede ver que los ICE tienen distintas ordenadas al origen, lo que dificulta la comparación entre distintas variables y la identificación de efectos heterogéneos.

Para solventar este problema, se puede utilizar el método de *Centered ICE* que consiste en centrar los ICE en algún punto de la distribución de la variable (los autores de este método sugieren el valor mínimo o máximo de la variable para tener gráficos sencillos de interpretar). Más formalmente se busca centrar cada curva $\hat{f}_i$ en un valor de referencia (x^*, x_{ci}) tal que:

$$\hat{f}^{(i)}_{cent} = \hat{f}^{(i)} - 1\hat{f}(x^*, x_{ci}) \tag{4.14}$$

A modo de ejemplo, si x^* es el valor mínimo de x_s todas las curvas empiezan en cero, por otro lado si es el máximo se puede ver ver el efecto acumulado de x_s sobre $\hat{f}$ relativo al caso base.

A modo de ejemplo, en la fig-plot-centered-ice se puede ver el *Centered ICE* para la misma variable de antes. En este caso x^* es el valor mínimo por lo que todas las curvas empiezan en cero. También, se puede ver en rojo la curva de PDP centrada.

4.2.2. *Derivative ICE*

Con el fin de seguir identificando las interacciones entre las variables los autores proponen calcular los gráficos de la derivada parcial de $\hat{f}$ con respecto a x_s. A modo

de formalizarlo, se asume independencia entre x_s y el resto de las covariables, por lo que $\hat{f}$ puede ser escrita como:

$$\hat{f}(x) = \hat{f}(x_s, x_c) = g(x_s) + h(x_c) \Rightarrow \frac{\partial \hat{f}(x)}{\partial x_s} = g'(x_s) \tag{4.15}$$

lo cual significa que la relación entre x_s y $\hat{f}$ no depende de x_c. Cuando esto se cumple todas las líneas deberían ser equivalentes.

Causalidad

Pese a que [**?**] menciona por qué los modelos de aprendizaje automático no pueden ser utilizados para hacer inferencia causal, dado que todos estos modelos se basan en el nivel de asociación entre variables, sin llegar a la intervención o contrafactuales. Sin embargo, [**?**] plantea que los PDPs y ICEs bajo ciertas condiciones pueden tener una interpretación en causal.

Los autores se basan en que @eq-pdp es igual al *backdoor adjustment* definido por [**?**] como

$$P(Y|do(X_s = x_s)) = \int P(Y|X_s = x_s, X_c = x_c) dP(x_c) \tag{4.16}$$

donde $P(Y|do(X_s = x_s))$ se refiere a la distribución de Y luego de hacer un intervención en X_s. Esta formula permite identificar el efecto causal de X_s en Y siempre y cuando se cumpla cuando (1) ningún nodo X_c sea dependiente descendiente de X_s y (2) que X_c 'bloquee' cualquier camino "back-door.[en]tre X_s e Y que se puede pensar intuitivamente como una causa común entre X_s e Y que impide entender el efecto causal de X_s.

$$E[Y|do(X_s = x_s)] = \int E[Y|X_s = x_s, X_c = x_c] dP(x_c) \tag{4.17}$$

Ahora si partimos de $\hat{f}(x)$ la función aprendida por el modelo, que aproxima la expectativa condicional, es decir,

$$\hat{f}(x) \approx E[Y \mid X = x]. \tag{4.18}$$

asumiendo que el modelo está bien calibrado, es decir,

$$\hat{f}(x_S, x_C) = E[Y \mid X_S = x_S, X_C = x_C], \tag{4.19}$$

podemos vincular ambas expresiones de la siguiente forma:

$$\begin{aligned} \text{PDP}(x_S) &= \int \hat{f}(x_S, x_C)\, dP(x_C) \\ &= \int E[Y \mid X_S = x_S, X_C = x_C]\, dP(x_C) \\ &= E[Y \mid do(X_S = x_S)] \end{aligned} \tag{4.20}$$

por lo que puede verse que eq-pdp y eq-back-door-expectation son equivalentes si condicionando en el set C es complementario al set S.

4.3. Relación con el proceso generador de los datos

Es interesante considerar que, aunque los modelos de aprendizaje automático suelen superar en rendimiento predictivo a los métodos estadísticos convencionales, estos últimos permiten establecer un vínculo directo entre los parámetros del modelo y las propiedades del proceso generador de datos (DGP). Esta capacidad resulta fundamental para comprender en profundidad la naturaleza subyacente de la población.

En [**?**] se propone un marco estadístico que relaciona formalmente las técnicas de interpretación basadas en *Partial Dependence Plots* (PDP) y *Permutation Feature Importance* (PFI) con el DGP. En este contexto, se introducen los conceptos de DGP-PD y DGP-PFI, definidos respectivamente como

$$DGP\text{-}PD(x) = E_{X_C}\left[f(x, X_C)\right] \tag{4.21}$$

y

$$DGP\text{-}PFI = E_{\bar{X}_S, X_C, Y}\left[L\left(Y, f(\bar{X}_S, X_C)\right)\right] - E_{X,Y}\left[L\left(Y, f(X)\right)\right] \tag{4.22}$$

donde $f(x) = E[Y|X = x]$ es la función de esperanza condicional. Los autores realizan una descomposición del error de estos estimadores en términos de sesgo y varianza, identificando fuentes de error tales como la aproximación de Monte Carlo para el cálculo de las esperanzas, la posible especificación incorrecta del modelo y la variabilidad inherente al proceso de entrenamiento.

Para capturar la incertidumbre debida a la variabilidad en el entrenamiento, se introduce la metodología *learner-PD/PFI*, la cual promedia los resultados obtenidos a partir de múltiples modelos reajustados. Esto permite incorporar la variabilidad del aprendizaje en la estimación y, en consecuencia, obtener intervalos de confianza que reflejen de manera más realista la incertidumbre de las medidas de interpretación.

No obstante, esta metodología asume condiciones fuertes, como la insesgadez del modelo. Además, los intervalos de confianza pueden resultar demasiado estrechos debido al solapamiento en el remuestreo utilizado para reentrenar los modelos, y la dependencia entre variables puede afectar los estimadores.

5. Aplicaciones

En esta sección se utilizarán tres conjuntos de datos para probar algunas de las técnicas descriptas anteriormente. En particular, se utilizará *Airfoil Self-Noise*, *Concrete Comprehensive Strenght* y *Wine Quality*. Para todos los casos se realizará un breve análisis exploratorio, seguido por una búsqueda de hiperparámetros con el fin de encontrar la combinación de los mismos que minimice el error de validación cruzada en la muestra de entrenamiento. Dicho subconjunto de datos tendrá el 80 % de los datos. Una vez encontrada la combinación óptima de hiperparámetros se evaluarán los modelos en la muestra de testeo, para luego calcular PFI y PDP. Cabe aclarar que los modelos a entrenar y comparar serán una regresión lineal, un árbol de decisión, *Random Forest* y *Gradient Boosting Machines*. Las métricas que se usarán para evaluar los modelos son: Error Cuadrático Medio (RMSE), Error Absoluto Medio (MAE) y Coeficiente de Determinación (R^2). El RMSE mide la raíz cuadrada del promedio de los errores al cuadrado: $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$, el MAE mide el promedio de los errores absolutos: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, y el R^2 indica la proporción de la varianza en la variable dependiente que es predecible a partir de las variables independientes: $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$.

En todos los casos para los árboles de decisión, se testearan los hiperparámetros: profundidad máxima (10, 20, 30), mínimo numero de observaciones para hacer una división (2, 5, 10) y mínimo número de observaciones por hoja (1, 2, 4). Para *Random Forest*, se testeará el hiperparámetro número de árboles (100 a 2000 en incrementos de 100). Para *Gradient Boosting Machines*, se testearan los hiperparámetros número de árboles (100 a 2000 en incrementos de 100) y learning rate (0.01, 0.1, 0.2).

Cabe aclarar que el objetivo de las aplicaciones no es llegar a modelos óptimos, ni entender realmente cómo las variables interaccionan. Sino simplemente demostrar algunas de las metodologías descritas en los capítulos anteriores.

5.1. *Airfoil Self-Noise*

Este conjunto de datos pertenece a la NASA y fue obtenido a partir de una serie de pruebas aerodinámicas y acústicas de secciones de palas de perfiles aerodinámicos bidimensionales y tridimensionales realizadas en un túnel de viento anecoico [**?**] . Los datos contienen perfiles aerodinámicos NACA 0012 de diferentes tamaños a diversas velocidades de túnel de viento y ángulos de ataque. La envergadura del perfil aerodinámico y la posición del observador fueron las mismas en todos los experimentos. A continuación vemos una tabla que resume las variables incluidas:

— Variable — Descripción — ————-————-— — frequency — Frecuencia en hertzios — — attack-angle — Ángulo de ataque en grados — — chord-length

— Longitud de cuerda en metros — — free-stream-velocity — Velocidad de flujo libre en metros por segundo — — suction-side-displacement-thickness — Espesor de desplazamiento del lado de succión en metros — — scaled-sound-pressure — Presión sonora escalada en decibelios (variable dependiente) —

En particular, se buscará predecir la variable *scaled-sound-pressure* utilizando las demás. Los datos cuentan con 1503 observaciones, sin datos faltantes.

En la fig-air_corr puede verse la correlación entre las variables que incluye el conjunto de datos. Queda claro que hay una alta correlación entre '*suction-side-displacement-thickness*' y '*attack-angle*'. Por lo que se decidió eliminar el angulo de ataque por tener una correlación relativamente alta con '*chord-lenght*' y menor que '*scaled-sound-pressure*' con la variable de respuesta.

Habiendo entrenado los modelos con los hiperparámetros definidos en la introducción. Se puede ver en la tbl-air_predictions los resultados en la muestra de testeo. En particular se destaca que *Gradient Boosting* tiene un menor error en todas las métricas. Por otra parte, la regresión lineal tiene el peor ajuste, esto último se repite con los tres conjuntos de datos que se probaron y va en línea con que los modelos más complejos tienden a tener una mejor performance predictiva.

A continuación, se procede a utilizar *Permutation Feature Importance* y *Partial Dependence Plots* para entender cómo el modelo de *Gradient Boosting* utilizó las variables independientes. Empezando con la fig-air_permutation_importance se puede ver que '*frequency*' y '*suction-side-displacement-thickness*' son las variables más importantes, seguidas por '*chord-length*' y '*free-stream-velocity*'.

Habiendo identificado las variables más importantes, el próximo paso sería entender cómo estas influyen en la variable de respuesta. Es decir como distintos cambios en la '*frequency*' (Hz) y '*suction-side-displacement-thickness*' (m). Para ello, se puede ver en la fig-air_partial_dependence los ICE (lineas celeste claro)y PDP (linea punteada) de dichas variables, junto con la cantidad de observaciones (líneas verticales en el eje X) La frecuencia exhibe una relación descendente con el nivel de presión sonora. En el rango de frecuencias estudiado (0-20000 Hz), se observan las siguientes características: La región de mayor influencia se encuentra en las frecuencias bajas (0-2500 Hz), donde el nivel de presión sonora alcanza sus valores máximos, aproximadamente 130-140 dB. A medida que la frecuencia aumenta, se evidencia una disminución gradual hasta alcanzar niveles de 100-110 dB a 20000 Hz. El análisis centrado revela que las frecuencias bajas contribuyen positivamente con una desviación de hasta +20 dB respecto a la media, mientras que las frecuencias altas tienen un efecto negativo de magnitud similar (-20 dB). Sin embargo, cabe destacar que prácticamente no hay observaciones por encima de 7500 de frecuencia. Por lo que las observaciones en estos niveles deberían tomarse con cautela.

Por otro lado, la relación entre el espesor de desplazamiento y el nivel de presión sonora presenta un comportamiento más complejo, los niveles de presión sonora se mantienen predominantemente en el rango de 120-130 dB. La respuesta muestra una naturaleza no lineal con múltiples discontinuidades. El análisis centrado indica fluctuaciones de ± 10 dB respecto al valor medio. Se observa una tendencia general levemente descendente, aunque menos pronunciada que en el caso de la frecuencia. Sin embargo, casi todas las observaciones se encuentran por debajo de 0.015.

5.2. *Concrete Compressive Strength*

A continuación se realiza el mismo análisis para el conjunto de datos *Concrete Compressive Strength* [**?**] que cuenta con 1030 observaciones distribuidas entre 9 variables. A continuación pueden verse todas las variables incluidas:

— Variable — Descripción — ————-————-— — Cement — Cantidad de cemento en la mezcla (kg/m^3) — — Blast Furnace Slag — Cantidad de escoria de alto horno (kg/m^3) — — Fly Ash — Cantidad de cenizas volantes (kg/m^3) — — Water — Cantidad de agua en la mezcla (kg/m^3) — — Superplasticizer — Cantidad de superplastificante (kg/m^3) — — Coarse Aggregate — Cantidad de agregado grueso (kg/m^3) — — Fine Aggregate — Cantidad de agregado fino (kg/m^3) — — Age — Edad del concreto en días — — Concrete compressive strength — Resistencia a la compresión del concreto (variable dependiente) —

La variable dependiente será la fuerza del concreto y se utilizarán todas las demás para predecirla.

En la @fig-concrete_corr se puede ver la correlación entre las variables, no se ven correlaciones por encima de 0.8 en valor absoluto por lo que se definió mantener todas las variables en el modelo.

En la tbl-concrete_predictions se pueden ver las predicciones de los modelos. Nuevamente, se ve que el modelos con mejor ajuste en la muestra de testeo es *Gradient Boosting*. A su vez puede verse que la regresión lineal es el que presenta las peores métricas.

En la @fig-concrete_permutation_importance puede verse la importancia de las variables del modelo *Gradient Boosting*, en particular se destaca la edad y la cantidad de cemento en la muestra como variables más importantes. Por lo que a continuación en la @fig-concrete_partial_dependence se pueden ver los PDPs y ICE de la edad y la cantidad de cemento en la muestra. Para la edad se observa una tendencia creciente pronunciada en los primeros 50 días, donde el efecto sobre la variable dependiente aumenta rápidamente. Después de los 50 días, la curva se estabiliza, mostrando un crecimiento más gradual y menos pronunciado. La relación con el cemento muestra un patrón escalonado, con varios puntos de inflexión claros. Se observan incrementos significativos en ciertos rangos específicos (alrededor de 250, 300 y 350 unidades). Es importante destacar que, para ambas variables, las líneas se ven mayormente paralelas. Esto último indica que dichas variables podrían ser bastante independientes de las demás.

5.3. *Wine Quality*

Estos datos son los resultados de un análisis químico de vinos cultivados en la misma región de Italia pero derivados de tres variedades diferentes. El análisis determinó las cantidades de 13 componentes encontrados en cada uno de los tres tipos de vinos. El conjunto de datos se obtuvo de [**?**] y contiene 4898 observaciones.A continuación se presenta una tabla con las variables del conjunto de datos y su significado:

— Variable — Significado — ————-————-— — fixed_acidity — Acidez fija en el vino (principalmente ácido tartárico) — — volatile_acidity — Acidez volátil en el vino (principalmente ácido acético) — — citric_acid — Ácido cítrico presente en

el vino — — residual_sugar — Cantidad de azúcar residual en el vino — — chlorides — Contenido de cloruros (sal) en el vino — — free_sulfur_dioxide — Dióxido de azufre libre en el vino (conservante) — — total_sulfur_dioxide — Dióxido de azufre total en el vino (libre + combinado) — — density — Densidad del vino — — pH — Nivel de acidez/alcalinidad del vino (escala pH) — — sulphates — Contenido de sulfatos en el vino — — alcohol — Contenido de alcohol en el vino (% por volumen) — — quality — Calidad del vino (puntuación entre 0 y 10) —

En la fig-wine_corr puede verse la correlación entre las variables del conjunto de datos. En particular, se destaca la alta correlación entre '*total-sulfur-dioxide*' y '*free-sulfur-dioxide*', por lo que se decidió eliminar la segunda.

En la tbl-wine_predictions pueden verse los resultados de los modelos en la muestra de testeo. En particular, se destaca que *Random Forest* tuvo la mejor performance predictiva, inclusive superando a *Gradient Boosting* que había sido el que tuvo mejores resultados en los otros dos conjuntos de datos.

En la fig-wine_permutation_importance puede verse la importancia de todas las variables independientes incluidas en el modelo de *Random Forest*. En particular, se destacan el alcohol y la acidez volátil del vino como las variables más importantes. En la fig-wine_partial_dependence pueden verse los PDP y ICE de las dos variables más importantes. Para el alcohol se observa una relación positiva general entre el contenido de alcohol y la calidad del vino, la tendencia es creciente pero no lineal, con varios puntos de inflexión notables: Un primer incremento suave entre 9 % y 11 % de alcohol, un salto más pronunciado alrededor del 12 % de alcohol y una estabilización después del 13 % de alcohol. Por lo que el efecto es más pronunciado en el rango medio (1113 %). Las líneas individuales muestran un patrón bastante paralelo, sugiriendo pocas interacciones con otras variables.

Para la acidez volátil: Muestra una relación negativa con la variable objetivo siendo la tendencia es ligeramente decreciente y relativamente lineal. Por otra parte, el efecto es más suave y menos pronunciado que el del alcohol observándose: Una pendiente negativa consistente a lo largo del rango, mayor variabilidad en las predicciones individuales (líneas azules claras más dispersas) y que el efecto negativo se hace más pronunciado en valores más altos de acidez volátil

6. Conclusion

dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt

A. Appendix Title

Appendix goes here...

Bibliografía

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

[Breiman, 2001b] Breiman, L. (2001b). Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231.

[Fisher et al., 2019] Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232.

[Shmueli, 2010] Shmueli, G. (2010). To Explain or to Predict? *Statistical Science*, 25(3):289 – 310.

[Strobl et al., 2008] Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, 9:307.

[Zhou and Hooker, 2020] Zhou, Z. and Hooker, G. (2020). Unbiased measurement of feature importance in tree-based methods.