Doctoral Program in Statistics, Optimization and Applied Mathematics

Efficiency Analysis Trees

Miriam Esteve Campello

Supervisor:

Dr. Juan Aparicio Baeza

Co-supervisor:

Dr. Alejandro Rabasa Dolado

MIGUEL HERNÁNDEZ UNIVERSITY OF ELCHE

2022

Programa de Doctorado en Estadística, Optimización y Matemática Aplicada

Efficiency Analysis Trees

Miriam Esteve Campello

Director de la tesis:

Dr. Juan Aparicio Baeza

Codirector de la tesis:

Dr. Alejandro Rabasa Dolado

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

2022

Tesis presentada para optar al grado de Doctor con Mención Internacional por la Universidad Miguel Hernández de Elche, realizada bajo la dirección de los doctores, Juan Aparicio Baeza y Alejandro Rabasa Dolado

INDICIOS DE CALIDAD DE LA TESIS DOCTORAL

El trabajo realizado por Dña. Miriam Esteve Campello, dirigido por Juan Aparicio Baeza y codirigido por Alejandro Rabasa Dolado, titulado **Efficiency Analysis Trees**, dentro del Programa de Doctorado en Estadística, Optimización y Matemática Aplicada se presenta bajo la modalidad de **tesis convencional con los siguientes artículos y capítulo de libro como indicios de calidad:**

- Esteve, M., Aparicio, J., Rabasa, A., & Rodriguez-Sala, J. J. (2020). Efficiency Analysis Trees: A New Methodology for Estimating Production Frontiers through Decision Trees. Expert Systems with Applications, 113783. https://doi.org/10.1016/j.eswa.2020.113783

- Esteve, M., Rodriguez-Sala, J. J., Lopez-Espin, J. J., & Aparicio, J. (2021). Heuristic and Backtracking algorithms for improving the performance of Efficiency Analysis Trees. IEEE Access, 1-8. https://doi.org/10.1109/ACCESS.2021.3054006

- Esteve, M., Aparicio, J., Rodriguez-Sala, J. J. & Zhu, J. (2022). Random Forests and the measurement of super-efficiency in the context of Free Disposal Hull. European Journal of Operational Research. https://doi.org/10.1016/j.ejor.2022.04.024

- Aparicio, J., Esteve, M., Rodríguez-Sala, J. J., & Zofío, J. L. (2021). The estimation of productive efficiency through machine learning techniques: Efficiency Analysis Trees. En *Data-Enabled Analytics: DEA for Big Data*. Editorial Springer. Ed. Joe Zhu and Vincent Charles. https://doi.org/10.1007/978-3-030-75162-3_3

INFORME DEL DIRECTOR Y CODIRECTOR DE LA TESIS

JUAN APARICIO BAEZA, Catedrático de Universidad del Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche y director del Centro de Investigación Operativa.

ALEJANDRO RABASA DOLADO, Profesor Contratado Doctor del Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche y miembro del CIO.

INFORMAN QUE

Dña. Miriam Esteve Campello ha realizado bajo nuestra supervisión el trabajo titulado **Efficiency Analysis Trees** para optar al grado de Doctor con Mención Internacional, en el Programa de Doctorado de Estadística, Optimización y Matemática Aplicada conforme a los términos y condiciones definidos en su Plan de Investigación y de acuerdo al Código de Buenas Prácticas de la Universidad Miguel Hernández de Elche, cumpliendo los objetivos previstos de forma satisfactoria para su defensa pública como tesis doctoral.

Y para que conste, en cumplimiento de la legislación vigente y a los efectos oportunos, firmamos el presente certificado.

Fdo.: Juan Aparicio Baeza

Fdo.: Alejandro Rabasa Dolado

INFORME DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO

Dr. Domingo Morales González, Coordinador del Programa de Doctorado en Estadística, Optimización y Matemática Aplicada de la Universidad Miguel Hernández de Elche

INFORMA QUE

El trabajo realizado por Dña. Miriam Esteve Campello, dirigido por el Dr. Juan Aparicio Baeza y codirigido por Alejandro Rabasa Dolado, titulado **Efficiency Analysis Trees**, ha sido autorizado por la Comisión Académica del Programa de Doctorado en Estadística, Optimización y Matemática Aplicada para su presentación y defensa en el correspondiente tribunal en el Universidad Miguel Hernández de Elche.

Y para que conste, en cumplimiento de la legislación vigente y a los efectos oportunos, firmo el presente certificado.

Fdo.: Domingo Morales González

Coordinador del Programa de Doctorado en Estadística, Optimización y Matemática Aplicada

FINANCIACIÓN

Esta tesis ha sido posible gracias a la ayuda FPU17/05365 financiada por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033 y por el "Fondo Social Europeo Invierte en tu futuro".

Esta publicación es parte del proyecto de I+D+i PID2019-105952GB-I00, financiado por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033/.

Además, esta publicación también es parte del proyecto de I+D+i MTM2016-79765-P, financiado por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033/ y por "FEDER Una manera de hacer Europa".

ÍNDICE

LISTADO DE FIGURAS Y TABLAS

FIGURAS

TABLAS

ABSTRACT

The definition of technical efficiency through the prior estimation of a production frontier has been a relevant topic in the literature related to production theory and engineering. In the last forty years, many parametric and non-parametric approaches have been introduced to estimate production frontiers for a given set of data. However, few of these methodologies are based on machine learning techniques, despite being a growing field of research. In this thesis, a new methodology based on regression trees is introduced to estimate the production frontiers satisfying the fundamental postulates of microeconomics, such as the property of free disposal. This new approach, known as Efficiency Analysis Trees (EAT), shares some similarities with the Free Disposal Hull (FDH) technique. However, unlike FDH, EAT overcomes the overfitting problem by using cross-validation to prune the deep tree obtained in a first stage. Through Monte Carlo simulations, the performance of EAT is measured, showing that the new approach reduces the mean square error associated with the estimation of the real frontier between 13% and 70% compared to standard FDH.

However, these individual decision trees have some drawbacks: (1) Individual trees do not usually have a high level of prediction accuracy, and (2) trees can be very poorly robust, that is, a small change in the data can cause a big change in the final structure of the fitted tree. That is why an aggregation learning method that works by building a multitude of decision trees at the time of training and aggregating the information from the individual trees into a final prediction value, a technique known as Random Forest,

shows that it is capable of overcoming these limitations (James et al., 2013). In this sense, in this thesis, the Random Forest technique is adapted (Breiman, 2001) (RF+EAT) to estimate production frontiers and technical efficiency. To do this, decision tree models are applied to estimate non-overfitted production possibility sets that satisfy the property of free disposability in the context of FDH. There are three main implications of the development of the new approach in this thesis. First, the estimates derived from technical efficiency are robust to resampling of the data and input variables. Secondly, a method is suggested to determine the importance of the input variables in the model, which allows a classification of the inputs to be established. Third, if the relationship between the sample size and the number of variables (inputs and outputs) is low or moderately low, the standard efficiency models in the literature may result in a considerable number of units being evaluated as technically efficient; especially in the case of FDH. This lack of discrimination is often referred to in the literature as the "curse of dimensionality." In this thesis, it is shown that the Random Forest technique can also be considered a remedy for this type of problem.

In another sense, from the computational point of view, the algorithm used by EAT is based on a heuristic technique to select the next node to be divided during the growth process of the corresponding decision tree. However, as shown in this thesis, this heuristic does not always produce the minimum mean square error among all the possible trees that could be developed. Therefore, one of the main objectives is to improve the accuracy of the production function estimator generated from EAT by resorting to backtracking techniques (Baase, 2009 and Horowitz and Sahni, 1978). In particular, we combine the idea behind the heuristic approach with the potentiality of backtracking ((Pearl, 1984 and Tarjan, 1972) to improve the quality of the EAT-based production function estimator. In addition, through this new approach, it is possible to reduce the computational load of the standard backtracking techniques applied to the EAT methodology, as shown in the simulated experiences carried out.

On the other hand, also from a computational approach, this thesis develops a new package in R, named **eat**, which includes the functions to estimate the production frontiers and the technical efficiency measures of EAT and RF+EAT. The package includes the functions to estimate the input and output oriented radial measures, the input and output oriented Russell measures, the directional distance function and the weighted additive model. Furthermore, from the perspective of visualizing the models, the package includes graphical representations of the production frontier through tree structures and obtaining rankings of input variable importance in the analysis. In this thesis, the operation of the package is described through the use of a real database.

RESUMEN

La definición de la eficiencia técnica mediante la estimación previa de una frontera de producción ha sido un tema relevante en la literatura relacionada con la teoría de la producción y la ingeniería. En los últimos cuarenta años se han introducido muchos enfoques paramétricos y no paramétricos para estimar las fronteras de producción dada una muestra de datos. Sin embargo, pocas de estas metodologías se basan en técnicas de aprendizaje automático, a pesar de ser un campo de investigación creciente. En esta tesis, se introduce una nueva metodología basada en árboles de regresión para estimar las fronteras de producción satisfaciendo los postulados fundamentales de la microeconomía, como la propiedad de libre disposición. Este nuevo enfoque, bautizado en inglés como Efficiency Analysis Trees (EAT), comparte algunas similitudes con la técnica de Free Disposal Hull (FDH). Sin embargo, y a diferencia de FDH, EAT supera el problema del sobreajuste utilizando validación cruzada para podar el árbol profundo obtenido en una primera etapa. Mediante simulaciones de Monte Carlo se mide el rendimiento de EAT mostrando que el nuevo enfoque reduce el error cuadrático medio asociado a la estimación de la frontera real entre un 13% y un 70% en comparación con el FDH estándar.

Sin embargo, estos árboles de decisión individuales presentan algunos inconvenientes: (1) Los árboles individuales no suelen presentar un alto nivel de exactitud de predicción y (2) los árboles pueden ser muy poco robustos, es decir, un pequeño cambio en los datos puede causar un gran cambio en la estructura final del árbol ajustado. Es por ello que un

método de aprendizaje por agregación que funciona construyendo una multitud de árboles de decisión en el momento de la formación y agregando la información de los árboles individuales en un valor de predicción final, técnica conocida como Random Forest, demuestra que es capaz de superar estas limitaciones (James et al., 2013). En este sentido, en esta tesis, se adapta la técnica Random Forest (Breiman, 2001) (RF+EAT) para estimar las fronteras de producción y la eficiencia técnica. Para ello, se aplica modelos de árboles de decisión para estimar conjuntos de posibilidades de producción no sobreajustados que satisfagan la propiedad de libre disponibilidad en el contexto de FDH. Las principales implicaciones del desarrollo del nuevo enfoque en esta tesis son tres. En primer lugar, las estimaciones derivadas de la eficiencia técnica son robustas al remuestreo de los datos y al remuestreo de las variables de entrada. En segundo lugar, se sugiere un método para determinar la importancia de las variables de entrada (inputs) en el modelo, que permite establecer una clasificación de los inputs. En tercer lugar, si la relación entre el tamaño de la muestra y el número de variables (inputs y outputs) es baja o moderadamente baja, los modelos de eficiencia estándar de la literatura pueden dar lugar a que un número considerable de unidades sean evaluadas como técnicamente eficientes; especialmente en el caso de FDH. Esta falta de discriminación suele denominarse en la literatura "maldición de la dimensionalidad". En esta tesis, se muestra que la técnica Random Forest también puede considerarse un remedio para este tipo de problemas.

En otro sentido, desde el punto de vista computacional, el algoritmo utilizado por EAT se basa en una técnica heurística para seleccionar el siguiente nodo que se va a dividir durante el proceso de crecimiento del árbol de decisión correspondiente. Sin embargo, como se muestra en esta tesis, esta heurística no siempre produce el mínimo error cuadrático medio entre todos los árboles posibles que podrían desarrollarse. Por consiguiente, uno de los principales objetivos es mejorar la precisión del estimador de la función de producción generada a partir de EAT recurriendo a técnicas de *backtracking* (Baase, 2009 y Horowitz y Sahni, 1978). En particular, combinamos la idea que subyace en el enfoque heurístico con la potencialidad del *backtracking* (Pearl, 1984 y Tarjan, 1972) para mejorar la calidad del estimador de la función de producción basada en EAT. Además, a través de este nuevo enfoque, se consigue disminuir la carga computacional

de las técnicas estándar de *backtracking* aplicadas a la metodología de EAT, como se muestra en las experiencias simuladas realizadas.

Por otro lado, también desde un enfoque computacional, en esta tesis se desarrolla un nuevo paquete en R, bautizado como **eat**, que incluye las funciones para estimar las fronteras de producción y las medidas de eficiencia técnica de EAT y RF+EAT. El paquete incluye las funciones para estimar las medidas radiales orientadas a los inputs y los outputs, las medidas de Russell orientadas a los inputs y outputs, la función de distancia direccional y el modelo aditivo ponderado. Además, desde la perspectiva de la visualización de los modelos, el paquete incluye representaciones gráficas de la frontera de producción mediante estructuras de árbol y la obtención de los rankings de importancia de las variables de entrada en el análisis. En esta tesis, se describe el funcionamiento del paquete mediante la utilización de una base de datos real.

CAPÍTULO 1: INTRODUCCIÓN

La medición de la eficiencia técnica con respecto al desempeño de cualquier tipo de empresa privada u organización pública ha sido una cuestión relevante para los gerentes de empresas, los ingenieros de producción y los encargados de formular políticas públicas en la Administración desde principios del siglo XX (véase, por ejemplo, Farrell, 1957; Fried et al, 2008; Arnaboldi et al, 2018 y O'Donnell, 2018). En este contexto, el objetivo es evaluar el rendimiento de un conjunto de observaciones, denominadas en general DMUs (Unidades Tomadoras de Decisiones o *Decision Making Units*, en inglés), que utilizan varios inputs para producir varios outputs en un análisis de "caja negra", ya que la forma en que se convierten los outputs a partir de los inputs, es decir, la tecnología subyacente, suele ser desconocida para cualquier observador externo (Cook y Zhu, 2014, y Lozano y Khezri, 2019). Un ejemplo de DMU es una granja que utiliza la tierra, la mano de obra y el capital (inputs/entradas) para producir, por ejemplo, fruta (output/producción). La medición de la eficiencia es un área de interés para una gran variedad de sectores, desde las empresas privadas que producen bienes hasta las industrias de servicios, como los hospitales. De igual modo, desde un punto de vista práctico, la estimación de la eficiencia es interesante para los ingenieros industriales dedicados a mejorar y supervisar el rendimiento de los procesos de fabricación dentro de las empresas. En este sentido, algunos trabajos recientes en esta línea son Yang y Chen (2020), donde la medición de la eficiencia se aplica para evaluar el rendimiento de la operación industrial de TFT-LCD de Taiwán, Olfati et al (2020), donde el análisis de la eficiencia se aplica bajo incertidumbre, y Shah et al (2019), donde se determina la eficiencia de los bancos comerciales.

Las técnicas modernas para medir la eficiencia técnica se basan en la evaluación de cierto tipo de distancia desde cada DMU hasta el límite (superior) de la tecnología, donde la tecnología representa el conjunto de puntos "producibles" en el espacio de inputs y outputs. Los puntos que pertenecen a esta frontera, también llamada frontera de producción, están asociados a las mejores prácticas (reales o virtuales) en materia de rendimiento y caracterizan la tecnología. De este modo, la estimación de la frontera de producción suele ser clave en la práctica para abordar el problema de la determinación de la eficiencia técnica a partir de una muestra de datos de las DMUs. Adaptando la nomenclatura de Färe y Primont (1995), donde se usa la notación x e y para denotar a los inputs $\boldsymbol{x} = (x_1, x_2, ..., x_n) \in R_+^m$ y a los outputs $\boldsymbol{y} = (y_1, y_2, ..., y_s) \in R_+^s$, respectivamente, en términos generales puede definirse al conjunto de posibilidades de producción o tecnología, es decir al conjunto de combinaciones técnicamente factibles de $(\boldsymbol{x}, \boldsymbol{y})$, como:

$$\Psi = \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in R_+^{m+s} : \boldsymbol{x} \text{ puede producir } \boldsymbol{y} \right\} \tag{1}$$

En este conjunto se hacen ciertas suposiciones, como la monotonía (*free disposability*, en inglés) de los inputs y outputs, lo que de manera más formal significa que si $(\boldsymbol{x}, \boldsymbol{y}) \in \Psi$, entonces $(\boldsymbol{x}', \boldsymbol{y}') \in \Psi$ si $\boldsymbol{x}' \geq \boldsymbol{x}$ y $\boldsymbol{y}' \leq \boldsymbol{y}$. Normalmente, se asume la convexidad de Ψ (ver, por ejemplo, Färe y Primont, 1995). En lo que respecta a la medición de la eficiencia técnica, una cierta parte del borde del conjunto de Ψ es de interés. Específicamente, la frontera eficiente de Ψ se puede definir como $\partial(\Psi) := \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in \Psi : \hat{\boldsymbol{x}} < \boldsymbol{x}, \hat{\boldsymbol{y}} > \boldsymbol{y} \Rightarrow (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \notin \Psi \right\}$. La eficiencia técnica se define como la distancia desde un punto que pertenece a Ψ hasta la frontera de producción $\partial(\Psi)$. Para un punto situado dentro de Ψ, es evidente que hay muchos caminos posibles para llegar a la frontera, cada uno asociado con una medida de eficiencia técnica diferente. Una de las medidas de eficiencia técnica más utilizas en la literatura es la conocida como modelo radial orientado al output (véase Charnes et al, 1978 y Banker et al., 1984), que es el inverso de la función clásica de la distancia de salida de Shepard (1953). Esta medida es

especialmente útil en un contexto en el que los inputs están predeterminadas (como, por ejemplo, la tierra y la mano de obra en una explotación agrícola) y la generación del máximo rendimiento es el objetivo apropiado desde una perspectiva técnica. En particular, el modelo radial orientado al output determina la puntuación de la eficiencia para un punto evaluado aumentando equiproporcionalmente todos sus outputs mientras se mantienen los inputs constantes:

$$\phi(\boldsymbol{x}_k, \boldsymbol{y}_k) = \max\{\phi_k \in R : (\boldsymbol{x}_k, \phi_k \boldsymbol{y}_k) \in \Psi\} \tag{2}$$

Las propiedades mencionadas anteriormente no son exhaustivas ni se mantienen universalmente. Por ejemplo, el supuesto de monotonía se relaja en los casos de congestión de los inputs (por ejemplo, cuando se contrata mano de obra hasta el punto de que tener más mozos de almacén interfieren en el trabajo de los demás debido a la congestión de personas en un espacio físico reducido).

Para ilustrar alguna de estas ideas, la Figura 1 representa una función de producción definida sobre un solo input x que produce un solo output y.

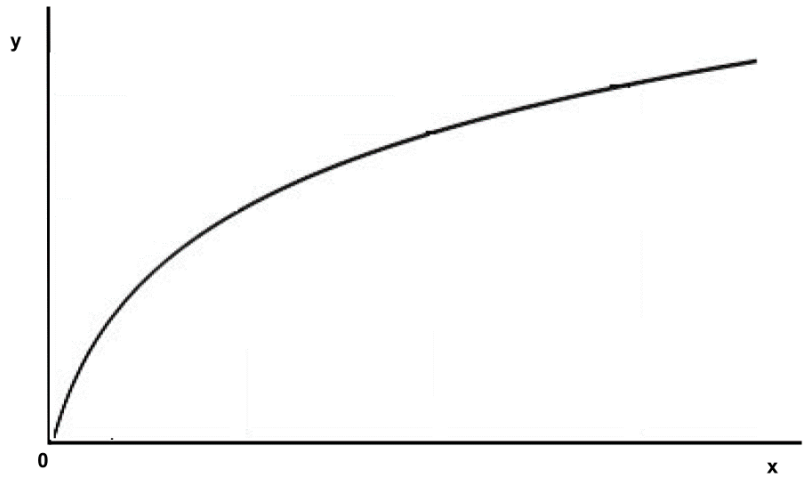


Figura 1. Frontera de producción para un solo output. Elaboración propia.

La estimación, en la práctica, de estas funciones de producción como superficies envolventes de las observaciones que satisfaces las propiedades anteriormente descritas (como la monotonía y la convexidad) comenzó con los trabajos de Farrell et al (1957), Aigner y Chu (1968) y Afriat (1972). Hoy en día, la estimación de las fronteras de producción se basa en metodologías muy diferentes, organizadas principalmente en técnicas paramétricas y no paramétricas. En el primer enfoque se necesita especificar paramétricamente la expresión matemática de la función de producción en función de una serie de parámetros que deben ser estimados y adoptar un conjunto de supuestos sobre los términos de error y eficiencia. En el mundo paramétrico, la función de producción debe ser descrita paramétricamente y debe añadir una componente estocástica que describa los choques aleatorios que afecten al proceso de producción. En este modelo, el Análisis de Fronteras Estocásticas (*Stochastic Frontier Analysis*, en inglés) (SFA) destaca como la metodología más relevante para estimar funciones de producción (Aigner et al, 1977; Meeusen y Broeck, 1977). En el segundo enfoque, el no paramétrico, la función de producción solo se basa en un conjunto de pocos postulados porque se basa más en los datos. Los métodos no paramétricos son capaces de considerar todos los outputs al mismo tiempo en el modelo de forma más natural que en el enfoque paramétrico, ya que éstas necesitan recurrir al concepto de función de distancia y asumir una serie de supuestos (Orea y Zofío, 2019). En la presente tesis, se centra toda la atención en los métodos no paramétricos.

En la actualidad, algunos de los enfoques no paramétricos más famosos para estimar las fronteras de producción son el Análisis Envolvente de Datos (o *Data Envelopment Analysis*, en inglés) (DEA) de Charnes et al. (1978) y Banker et al. (1984) y el Free Disposal Hull (FDH) de Deprins et al. (1984). En el caso de DEA, el estimador de la frontera de producción es una función lineal a trozos mientras que en el caso de FDH, el estimador es una función escalonada. DEA se basa en la construcción de una tecnología a partir de la satisfacción de unos pocos axiomas, la convexidad es uno de ellos. Otro postulado importante es el de libre disponibilidad (*free-disposability*, en inglés), que establece que, si cierto conjunto de inputs y outputs es producible, entonces cualquier conjunto de inputs y outputs que presente un valor mayor para los inputs y menor para

los outputs también es producible. En cierto sentido, significa que hacerlo peor siempre es factible. Además, se supone que DEA es determinista. Esto se refiere a que los valores de entrada y salida de DEA que pertenecen a la muestra de datos se han observado sin ningún ruido. Por consiguiente, la frontera de producción asociada con el conjunto de posibilidades de producción de DEA siempre envuelve por encima todas las observaciones de la muestra de datos. Sin embargo, muchos estimadores cumplen todos estos axiomas y, por lo tanto, se necesitan requisitos adicionales para determinar uno adecuado. En particular, la estimación más conservadora de la frontera de producción sería la asociada a una superficie que envolviera los datos y, al mismo tiempo, fuera lo más cercana posible a ellos. Este es el principio de conservación, también conocido como de mínima extrapolación o navaja de Ockham. Por supuesto, este último postulado es asimismo responsable del problema de sobreentrenamiento o sobreajuste (*overfitting*, en inglés) asociado a DEA. La estimación de DEA se ajusta como "un guante" a los datos con el que se ha construido el modelo. Por otro lado, Deprins et al. (1984) introdujeron el enfoque alternativo conocido como Free Disposal Hull (FDH), que es una técnica que se basa únicamente en la libre disponibilidad y en la mínima extrapolación y que, a diferencia de DEA, no asume la convexidad. Estas características conducen a un estimador de la frontera de producción basado en FDH, que tiene una forma escalonada, y que comparte con DEA el problema estadístico del sobreentrenamiento.

El problema del sobreentrenamiento que presentan DEA y FDH está relacionado con la falta de potencia de estas técnicas cuando el objetivo es de naturaleza inferencial. Se puede asumir que un Proceso de Generación de Datos (o *Data Generating Process*, en inglés) (DGP) produjo las observaciones, es decir, las DMUs. Para muchos científicos la capacidad de predecir con precisión es una medida crucial del valor del modelo. Si el enfoque considerado no puede predecir bien significa que el modelo no puede captar los patrones esenciales del DGP original. De ello se deduce que el modelo puede tener poco valor, excepto si su objetivo es meramente descriptivo para las observaciones. De hecho, un modelo que se ajusta bien a la muestra de los datos, como DEA o FDH, no necesariamente predecirá bien. El llamado, en el aprendizaje estadístico, rendimiento

fuera de la muestra (*out-of-sample*, en inglés) es tan importante como el rendimiento dentro de la muestra del modelo considerado. Si la capacidad de predicción es un criterio muy importante para evaluar un modelo, entonces la contribución de un predictor a esa capacidad es seguramente una medida razonable de la importancia de su predicción (Berk, 2016). La forma de evaluar el error de predicción de un modelo, también conocido como error de generalización, no consiste en utilizar los mismos datos que se emplean en el proceso de ajuste del modelo, sino en recurrir a una muestra de datos de prueba diferente (por ejemplo, aplicando validación cruzada, véase Browne, 2000).

La incorporación de ciertas características a DEA y a FDH vinculadas a las técnicas de predicción e inferencia ha atraído la atención de algunos investigadores en las últimas décadas. Cronológicamente hablando, Banker y Maindiratta (1992) y Banker (1993) demostraron que DEA puede ser interpretado como un estimador de máxima verosimilitud. Más tarde, Simar y Wilson (1998) y Simar y Wilson (2000a, 2000b) introdujeron la forma de determinar los intervalos de confianza para la puntuación (*score*, en inglés) de la eficiencia técnica, adaptando la metodología de bootstrapping (Efron, 1979) al contexto de DEA y FDH. Kuosmanen y Johnson (2010, 2017) demostraron que DEA puede interpretarse como una regresión de mínimos cuadrados no paramétricos sujeta a restricciones de forma en la frontera de producción y limitaciones de signo en los residuos. Además, estos autores presentaron los Mínimos Cuadrados Cóncavos No Paramétricos Corregidos (*Corrected Concave Non-Parametric Least Squares*, en inglés) y demostraron que, si el proceso de generación de datos es determinístico y los términos de ineficiencia están distribuidos de forma idéntica e independiente, su estimador es asintóticamente consistente. Otras técnicas alternativas no paramétricas recientes para estimar las fronteras de producción son las que aplican enfoques basadas en funciones de kernel y técnicas de regresión local (véase Du et al. 2013 y Tortosa-Ausina et al., 2012), donde los autores proponen un método de suavizado de kernel que considera restricciones de forma (por ejemplo, la monotonía) para funciones multivariadas, generalizando Hall y Huang (2001). Otra contribución interesante es la de Parmeter et al. (2014), que mostraron cómo se puede aplicar el bootstrapping ponderado por restricciones para imponer condiciones de suavidad a las estimaciones lineales. En particular, estos autores estimaron

una función de distancia de entrada tanto de forma paramétrica como no paramétrica, recurriendo en este último caso a la regresión lineal local generalizada. Véase también Henderson y Parmeter (2009). Sin embargo, ninguno de los enfoques mencionados aborda el problema mediante técnicas de aprendizaje automático, a pesar de que es uno de los marcos teóricos más naturales que se deben considerar dada la naturaleza basada puramente en datos de DEA y FDH.

El aprendizaje automático, la minería de datos, los grandes volúmenes de datos o el análisis de datos son sólo algunas de las formas de referirse al conjunto de técnicas modernas, basadas en algoritmos informáticos, que mejoran automáticamente a partir de la información proporcionada por un conjunto de datos de "entrenamiento", con el fin de hacer predicciones o tomar decisiones sin ser supervisados explícitamente para ello. En los últimos tiempos, estas técnicas suelen estar vinculadas a grandes bases de datos. En este aspecto, la comunidad científica, en general, se está orientando hacia este campo del análisis de datos, donde las técnicas de aprendizaje automático de las máquinas son usuales. Uno de los problemas que resuelve este tipo de técnicas es el sobreentrenamiento o sobreajuste (*overfitting*, en inglés). Por ejemplo, los árboles de clasificación y regresión (o *Classification and Regression Trees*, en inglés) (CART) de Breiman et al. (1984) construyen árboles de decisión binarios dividiendo repetidamente los nodos en dos nodos hijos, comenzando por el nodo raíz que contiene toda la muestra de aprendizaje. Sin embargo, un árbol demasiado profundo produce estimaciones demasiados optimistas, es decir, demasiado ajustadas a los datos y un árbol demasiado corto presenta una baja precisión sobre el conjunto de aprendizaje. En este sentido, Breiman et al. (1984) introdujeron una medida de complejidad de costes y un procedimiento de poda, basado en la validación cruzada, para superar dicho problema y evitar el sobreajuste en árboles demasiado profundos.

La presente tesis está organizada de la siguiente manera. En el Capítulo 2 se presentan los objetivos principales de la tesis. En el Capítulo 3 se introducen las técnicas FDH,

CART y Random Forest. En el Capítulo 4 Sección 4.1. se muestra la nueva metodología, bautizada con el nombre de Árboles de Análisis de Eficiencia (*Efficiency Analysis Trees*, en inglés) (de ahora en adelante nos referiremos a esta técnica como EAT), basada en las dos primeras técnicas descritas en el Capítulo 3. Además, se añaden ejemplos ilustrativos de la nueva técnica. En la Capítulo 4 Sección 4.2. se presenta la evolución de EAT al Random Forest para dotar a la nueva técnica de robustez. En la Capítulo 4 Sección 4.3. de la presente tesis se enfoca en los problemas computacionales derivados de esta nueva técnica y en sus soluciones a nivel de implementación y eficiencia computacional. En la Capítulo 5 Sección 4.4. se presenta el paquete de R (CRAN) que incorpora las principales funcionalidades de EAT, Random Forest para EAT, DEA y FDH. Por último, en el Capítulo 5 y Capítulo 7 se recogen las discusiones y conclusiones de todo el trabajo desarrollado en esta tesis.

CAPÍTULO 2: OBJETIVOS

FDH es una de las técnicas no paramétricas para la estimación de fronteras de producción que asume que, si cierto conjunto de inputs y outputs es producible, entonces cualquier otro conjunto de inputs y outputs que presente un valor mayor para los inputs y un valor menor para los outputs también es producible (libre disponibilidad). Esta propiedad está relacionada con la estimación de las funciones de producción no decrecientes. Adicionalmente, FDH es determinístico, lo que significa que siempre contiene a todas las observaciones que pertenecen a la muestra de datos (es decir, a la muestra de aprendizaje siguiendo la terminología del aprendizaje automático) y, gráficamente, la frontera de producción (superior) envuelve toda la nube de datos. Finalmente, el tercer postulado que cumple FDH se conoce como mínima extrapolación y se asocia al típico principio de solución de problemas de la navaja de Ockham. Se necesita añadir requisitos para seleccionar el estimador debido a que hay muchos posibles estimadores que pueden cumplir con la libre disponibilidad y ser deterministas. En este sentido, si uno se atiene a la navaja de Ockham, la estimación más conservadora de la frontera de producción sería la relativa a una superficie que envolviera los datos desde arriba y, al mismo tiempo, lo más cerca posible de la nube de datos. Por todo ello, desde una perspectiva no paramétrica, FDH es una técnica muy atractiva, ya que no se basa en hipótesis limitadoras del Proceso de Generación de Datos (DGP). Ésta es una característica compartida con las técnicas habituales de aprendizaje automático, que son claramente enfoques basados en datos. Sin embargo, FDH estándar no puede considerarse una técnica de aprendizaje adecuada. El objetivo principal de FDH parece ser la descripción del comportamiento extremo de un conjunto de observaciones, satisfaciendo los axiomas microeconómicos

antes mencionados. En cambio, la mayoría de las técnicas del aprendizaje automático (como, por ejemplo, árboles de clasificación y regresión, bosque aleatorio, máquinas de vectores soporte, por nombrar algunas) presentan además otros objetivos más allá de la estadística descriptiva. Se ocupan de la noción de generalización o error de predicción, que es una medida de la precisión con que un método puede predecir los valores de los resultados de datos no vistos anteriormente. En particular, el principio de mínima extrapolación no permite que FDH abandone el campo de la estadística descriptiva para entrar en el más interesante y moderno campo del aprendizaje estadístico. Una consecuencia de esta característica de FDH es su conocido problema de sobreajuste. El principio de mínima extrapolación es responsable de este inconveniente. La superficie envolvente superior estimada por FDH encaja como un "guante" en la muestra de datos. En la práctica, el problema del sobreajuste se observa en el hecho de observar que la mayoría de DMUs evaluadas con una puntuación de eficiencia por el FDH es de uno (es decir, muchas unidades son identificadas como eficientes). Además, la potencia de predicción o generalización de FDH es baja, por lo que reporta malos resultados cuando el objetivo es predecir, por ejemplo, el valor de salida de una nueva unidad no observada. En el marco tradicional, la noción de generalización y error de predicción está vinculada a la inferencia estadística. Por supuesto, si la descripción de la muestra es el objetivo principal, entonces FDH es una técnica adecuada. Sin embargo, los investigadores y profesionales deben ser conscientes de las limitaciones de las técnicas como las de FDH para determinar una estimación precisa del DGP subyacente. Si aplicamos FDH estándar, algunas unidades evaluadas en la muestra pueden parecer técnicamente eficientes, pero sólo lo son dentro de esta muestra. En cierto sentido, la definición habitual de eficiencia técnica en el marco de FDH es una noción específica de la muestra o una estimación dentro de la muestra de la puntuación de eficiencia. Por el contrario, las técnicas habituales del aprendizaje automático aplican la noción de estimaciones fuera de la muestra, centrándose en el DGP real. Las estimaciones de la función de producción subyacente generadas por la técnica de FDH son funciones escalonadas (véase Figura 3). Otra técnica no paramétrica muy conocida para determinar los conjuntos de posibilidades de producción es el análisis envolvente de datos (DEA) de Charnes et al. (1978) y Banker et al. (1984). En este caso, las estimaciones son funciones lineales a trozos. A diferencia de FDH, DEA también asume la convexidad. Esto significa que, si se supone que dos

perfiles de entrada y salida son producibles, entonces cualquier combinación convexa de ellos también es factible. Gráficamente, FDH podría considerarse el "esqueleto" de DEA, ya que la envoltura convexa de la frontera estimada por FDH coincide con la frontera de DEA (Daraio y Simar, 2005) (véase Figura 4).

La idea detrás del concepto de generalización en la teoría del aprendizaje estadístico (Vapnik, 2000) consiste en lograr un equilibrio entre la precisión alcanzada en algunos de los datos (conjunto de entrenamiento) y la capacidad del método para "aprender" cualquier conjunto de entrenamiento sin errores. Como se ha comentado, muchas de las técnicas del aprendizaje automático siguen ciertos objetivos predictivos que, además, permiten paliar el problema del sobreentrenamiento o sobreajuste. En concreto, los Árboles de Clasificación y Regresión (CART) de Breiman et al. (1984) introducen una medida de complejidad de costes y un procedimiento de poda basado en la validación cruzada que permite superar este problema. CART para la regresión permite predecir el valor de una variable respuesta numérica a partir de un conjunto de predictores. El éxito de CART en la literatura especializada se debe, en parte, a la simplicidad del principio que está detrás del método: se selecciona un criterio (normalmente minimizar el error cuadrático medio) para generar recursivamente particiones binarias de los datos. Gráficamente hablando, la salida final de este procedimiento es un árbol que comienza en el nodo raíz, que contiene todos los datos observados, se construye a través de nodos intermedios y termina en los nodos terminales, también llamados hojas, que contienen subconjuntos disjuntos de los datos observados (véase Figura 5).

La naturaleza binaria del proceso se refleja en cada uno de los nodos padre, excepto en las hojas, que dan lugar a dos nodos hijos. La división de los datos en cualquier nodo está causada por una variable predictiva y un umbral para esta variable. Dadas todas las posibles combinaciones de cada predictor y cada umbral, CART construye árboles de regresión eligiendo la división que minimiza la suma del error cuadrático medio (MSE)

de los dos nodos hijos. CART produce una predicción sobre la media de la variable de respuesta condicionada a los valores de los predictores.

Alcanzado este punto, si se comparan las Figura 2 y Figura 5, es decir las funciones de predicción de FDH y de CART, se observa claramente un gran parecido en la forma "escalonada" de sus estimadores. En este sentido, la hipótesis principal que se plantea en esta tesis es que, al existir un enorme parecido en la estructura "escalonada" de sus funciones, parece posible relacionar estas dos técnicas, CART y FDH.

El objetivo principal de la tesis consiste en adaptar la técnica de los árboles de regresión, basada en CART, para que sean capaces de estimar las fronteras de producción que satisfagan las propiedades de libre disponibilidad y que envuelvan por encima los datos. A través del nuevo enfoque, se observará que el espacio de inputs es dividido por una secuencia de particiones binarias en nodos terminales (o nodos hoja). En cada nodo terminal, el valor de la respuesta prevista (el output) será constante. Así que, gráficamente, la función del predictor parecerá "escalonada" y presentará similitudes y diferencias con respecto a FDH cuando se construya su árbol profundo. Por otra parte, con el fin de superar el problema habitual del sobreentrenamiento o sobreajuste, se aplicará un procedimiento de poda basado en la validación cruzada. Por esta razón, la nueva metodología bautizada como *Efficiency Analysis Trees* (EAT) también podrá ser vista como una versión fuera de la muestra de FDH. Véase también Misiunas et al. (2016), Zhu et al. (2018), Badiezadeh et al. (2018), Zhu (2019), Lee y Cai (2020) y Olesen y Ruggiero (2022) para leer sobre el reciente interés de la comunidad DEA en salvar la brecha entre la estimación de las fronteras de producción y la ciencia de datos, el aprendizaje automático y los grandes volúmenes de datos. Entre otras ventajas, se permitirá determinar la evaluación de la eficiencia fuera de la muestra para las DMUs evaluadas dentro del nuevo modelo. Además, y aunque este tipo de técnica no paramétrica no puede aportar información relativa a la importancia o significación estadística de cada predictor (variables de entrada), la nueva metodología propuesta será capaz de determinar un listado ordenado de la importancia de cada uno de ellos. Finalmente, y a partir del

punto de vista de la visualización de los datos, el nuevo enfoque representará gráficamente a través de simples árboles situaciones multivariantes que de otra manera serían difíciles de dibujar.

Otro de los objetivos que se abordan en la tesis, es el de mezclar otra de las técnicas del aprendizaje automático basado en árboles y la estimación de fronteras de producción, adaptando el Random Forest de Breiman (2001) para estimar la eficiencia técnica. Random Forest suele proporcionar un mejor rendimiento predictivo que el que se podría obtener de cualquiera de los árboles de decisión individuales que lo componen. En este nuevo enfoque, los árboles de decisión se construirán para estimar las superficies envolventes superiores de todas las observaciones en el correspondiente espacio de input-output, capturando el comportamiento extremo de los datos, y satisfaciendo las propiedades del FDH, como libre disponibilidad. Esta adaptación, bautizada como RF+EAT, dará como resultado una nueva metodología más robusta en sus predicciones debido al remuestreo de los datos y las variables de entrada. También, esta nueva adaptación, sugerirá otro método para determinar la importancia de las variables de entrada en el modelo, basándose en la formulación que se suele utilizar con frecuencia en Random Forest estándar. Otra aportación a la literatura de RF+EAT será la solución del problema de la "maldición de la dimensionalidad", que sufren técnicas como FDH y DEA.

Desde un enfoque computacional, el algoritmo implementado para EAT se basa en una técnica heurística para seleccionar el siguiente nodo que se va a dividir durante el proceso de crecimiento del árbol de decisión correspondiente. En la presente tesis, se demostrará que esta heurística no siempre producirá el mínimo error cuadrático medio de todos los posibles árboles que se podrían generar si planteásemos todas sus posibles combinaciones. En este aspecto, otro de los objetivos es el de mejorar la precisión de la función de predicción generada a partir de EAT recurriendo a técnicas de retroceso (o *backtracking*, en inglés) (Baase, 2009 y Horowitz y Sahni, 1978). En concreto, se

combinará la idea que subyace al enfoque heurístico con la potencialidad del *backtracking* (Pearl, 1984 y Tarjan, 1972) para mejorar la calidad del estimador de la función de producción de EAT. Además, gracias a este nuevo enfoque, se podrá disminuir la carga computacional de las técnicas estándar de *backtracking* aplicada a las metodologías de árbol de decisión como EAT, como se demostrará en las experiencias simuladas.

Por último, se introducirá la librería de EAT implementada en R que se encuentra disponible como software libre bajo la licencia GNU General Public License version 3 en el repositorio GitHub (https://github.com/MiriamEsteve/EAT) y que está disponible en el Comprehensive R Archive Network (CRAN) en https://cran.r-project.org/web/packages/eat.

CAPÍTULO 3: MATERIALES Y MÉTODOS

En este tercer capítulo se describen brevemente las principales nociones relacionadas con *Free Disposal Hull* (FDH), *Classification and Regression Trees* (CART) y Random Forest. Adicionalmente, se añadirán otras nociones necesarias para la introducción de la nueva metodología propuesta en el Capítulo 4.

3.1. FREE DISPOSAL HULL (FDH)

Hoy en día, hay diferentes métodos no paramétricos en la literatura para estimar la frontera eficiente de Ψ. Entre ellas, cabe mencionar las siguientes. El estimador FDH fue introducido por Deprins et al. (1984) y se basa sólo en el supuesto de libre disponibilidad. En cambio, el estimador DEA requiere supuestos más fuertes, como la convexidad del conjunto de Ψ. El supuesto de la convexidad se utiliza ampliamente en la economía, pero no siempre es válido. El conjunto de posibilidades de producción podría admitir rendimientos crecientes a escala (es decir, la producción aumenta más rápidamente que los inputs, lo que gráficamente no puede ser modelado por la convexidad), o podrían no existir valores fraccionarios de inputs o outputs. Por lo tanto, FDH puede dar un estimador más general y flexible que DEA (véase Aragon et al., 2005).

En lo que respecta a la técnica de FDH, este enfoque ha vuelto a atraer el interés de los investigadores en los últimos años. Por ejemplo, Tavakoli y Mostafaee (2019) adaptaron

los modelos de análisis envolvente de datos en red a modelos bajo libre disponibilidad, proporcionando los outputs intermedios objetivo para las observaciones ineficientes. Barbosa et al. (2019) desarrollaron un Algoritmo Genético Híbrido Evolutivo para trabajar sobre ciertos problemas de asignación en el que se explotan DEA y FDH para comparar el rendimiento de especificaciones alternativas en los parámetros del algoritmo. Kerstens et al. (2019) reconsideraron la forma en que se obtienen las metafronteras a partir de estimaciones no paramétricas de las fronteras subyacentes específicas de los grupos, y llegaron a la conclusión de que la estrategia habitual de convexificación consistente en suponer un metaconjunto convexo suele dar resultados erróneos. O, desde un punto de vista teórico, los trabajos de Cazals et al. (2002) y Daraio y Simar (2005), donde le introduce un estimador basado en FDH, robusto a los valores extremos y el concepto de medida de eficiencia condicional de orden-m, respectivamente.

Los modelos no paramétricos, como FDH, son especialmente atractivos, ya que no se basan en hipótesis restrictivas sobre el proceso de generación de datos, una característica compartida con las técnicas habituales de aprendizaje automático, que son claramente enfoques basados en datos. En particular, Deprins et al. (1984) propusieron la forma de estimar el conjunto de posibilidades de producción a través de *Free Disposal Hull* como sigue:

$$\hat{\Psi}_{FDH} = \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in R_{+}^{m+s} : \exists i = 1, ..., n \text{ tal que } \boldsymbol{y} \leq \boldsymbol{y}_i, \boldsymbol{x} \geq \boldsymbol{x}_i \right\} \tag{3}$$

En el caso de la producción univariante, la función de producción se estimaría mediante $\hat{f}_{FDH}(\boldsymbol{x}) = \max_{i:\boldsymbol{x} \geq \boldsymbol{x}_i} \{y_i\}$. A continuación, en la Figura 2, se presenta un ejemplo gráfico del estimador de FDH, mostrando su típica forma escalonada no decreciente.

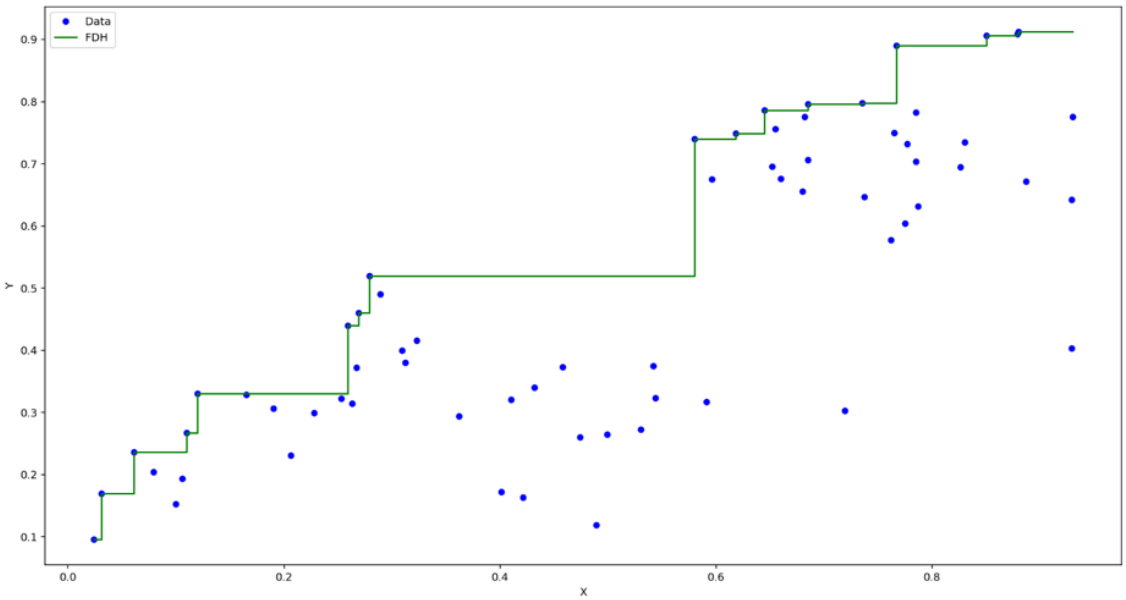


Figura 2. Frontera de producción de FDH.

Además, la puntuación de la eficiencia (su *score*) $\phi(\boldsymbol{x}_k, \boldsymbol{y}_k)$, para una medida radial orientada al output, puede ser estimada "enchufando" $\hat{\Psi}_{FDH}$ dentro de $\max\{\phi_k \in R : (\boldsymbol{x}_k, \phi_k \boldsymbol{y}_k) \in \Psi\}$ en lugar de Ψ. De esta forma, el problema de optimización puede ser reescrito como un programa de optimización lineal entero mixto:

$$
\begin{aligned}
\phi^{FDH}(\boldsymbol{x}_k, \boldsymbol{y}_k) = \quad & \max \quad \phi \\
& s.t. \\
& \sum_{i=1}^{n} \lambda_i x_{ji} \le x_{jk}, \qquad j = 1,\ldots,m \\
& \sum_{i=1}^{n} \lambda_i y_{ri} \ge \phi\, y_{rk}, \quad r = 1,\ldots,s \\
& \sum_{i=1}^{n} \lambda_i = 1, \\
& \lambda_i \in \{0,1\}, \qquad i = 1,\ldots,n
\end{aligned}
\tag{4}
$$

La técnica FDH es muy atractiva porque se basa en muy pocas suposiciones, pero, por construcción, sufre de sobreentrenamiento o sobreajuste (*overfitting*, en inglés). Nótese,

en la Figura 2, que la frontera estimada por FDH se ajusta como un "guante" a la muestra de datos, satisfaciendo además la libre disponibilidad (monotonía o *free disposability*). Este problema es compartido por otros enfoques bien conocidos basados en datos. En particular, las técnicas que pertenecen al campo del aprendizaje automático como, por ejemplo, los Árboles de Clasificación y Regresión (CART), presentan problemas de sobreajuste cuando se genera un árbol profundo. Este problema puede resolverse podando el árbol profundo mediante, por ejemplo, un proceso inteligente de validación cruzada.

El mismo tipo de medida técnica puede estimarse mediante el Análisis Envolvente de Datos por convexificación de la frontera de producción generada por FDH. Al fin y al cabo, FDH podría interpretarse como el "esqueleto" de DEA. A continuación, se muestra el modelo de Programación Lineal que se debe resolver en ese caso:

$$
\begin{aligned}
\phi^{DEA}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right) = \quad & \max \quad \phi \\
& s.t. \\
& \sum_{i=1}^{n} \lambda_i x_{ji} \leq x_{jk}, \qquad j = 1,\ldots,m \\
& \sum_{i=1}^{n} \lambda_i y_{ri} \geq \phi\, y_{rk}, \quad r = 1,\ldots,s \\
& \sum_{i=1}^{n} \lambda_i = 1, \\
& \lambda_i \geq 0, \qquad\qquad i = 1,\ldots,n
\end{aligned}
\tag{5}
$$

La diferencia entre los modelos (4) y (5) es la última restricción. DEA sólo requiere la no negatividad de las variables *lambda* (variables de intensidad) de valor real, mientras que FDH recurre a variables binarias. En la Figura 3 se ve claramente que DEA es el convexo de FDH.

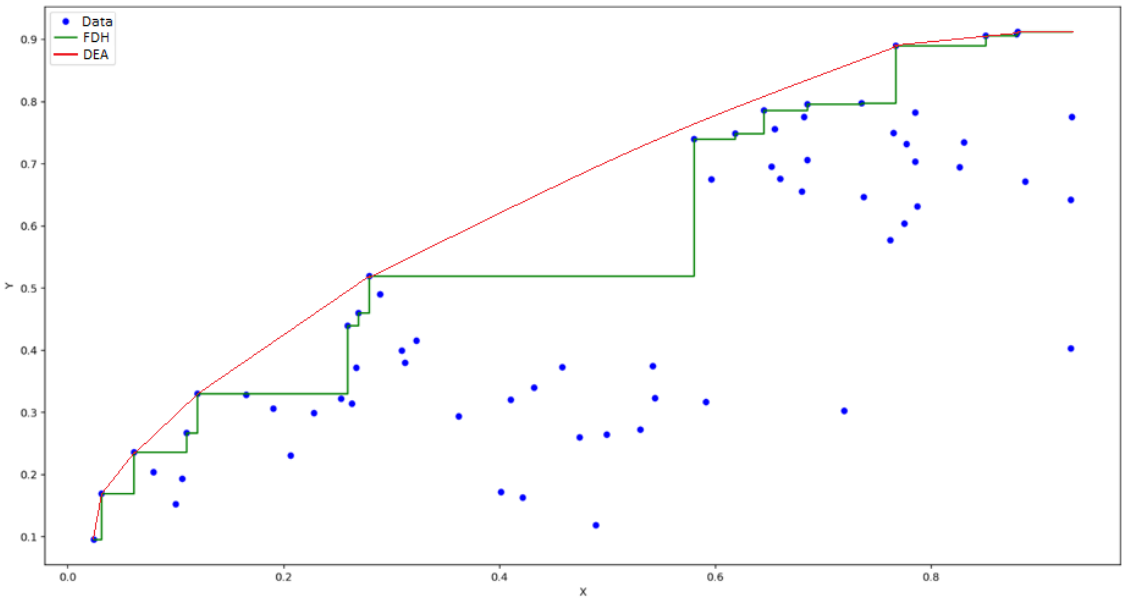


Figura 3. Frontera de producción de FDH y de DEA.

3.2. CLASSIFICATION AND REGRESSION TREES (CART)

CART (Breiman et al., 1984) es una técnica de aprendizaje automático con dos objetivos, dependiendo de la naturaleza de la variable respuesta. CART funciona como un modelo de clasificación no paramétrico cuando la variable respuesta es categórica y como un modelo de regresión cuando es numérica. Esta tesis se centra en este último enfoque, debido a la naturaleza de la variable respuesta (output en el contexto de producción). El principio que rige el funcionamiento de CART es relativamente simple: se elige un criterio determinado para generar recursivamente particiones binarias de los datos (la muestra de entrenamiento) hasta que ya no sea posible una división significativa o se mantenga una regla de parada. El resultado gráfico de este proceso es un árbol que comienza en el nodo raíz, se desarrolla a través de los nodos intermedios y termina en los nodos terminales (hojas). La naturaleza binaria de CART se refleja en cada nodo (padre), excepto en las hojas, dando lugar a dos nodos hijos. La división en cualquier nodo no terminal es causada por una variable predictiva y un umbral para esta variable. Dadas todas las formas posibles de dividir los datos en un nodo (padre) (es decir, las

combinaciones de cada predictor y cada umbral), CART construye árboles de regresión eligiendo la división que minimiza la suma del error cuadrático medio (MSE) de los dos nodos hijos. Específicamente, dada una muestra de entrenamiento $\aleph$ que consiste en $(x_1, y_1),...,(x_n, y_n)$, con $x_i \in R^m$ e $y_i \in R$, $i = 1,...,n$, CART tiene como objetivo predecir la variable respuesta y a través de los predictores $x_1,...,x_m$. Para hacer esto, la primera división de CART selecciona una variable predictiva j, $j = 1,...,m$, y un umbral $s_j \in S_j$, donde S_j es el conjunto de umbrales para la variable j, de tal forma que la suma del MSE calculada para los datos pertenecientes al nodo hijo izquierdo, es decir, los datos que satisfacen la condición $x_j < s_j$, y el MSE determinado con los datos pertenecientes al nodo hoja derecho, es decir, los datos que cumplen $x_j \geq s_j$, es minimizado. Si t es la forma de denotar al nodo padre y , además, t_L y t_R son los nodos hoja izquierdo y derecho, respectivamente, entonces, CART selecciona la mejor combinación (x_j, s_j) que minimiza:

$$R(t_L) + R(t_R) = \frac{1}{n} \sum_{(x_i, y_i) \in t_L} \left(y_i - y(t_L)\right)^2 + \frac{1}{n} \sum_{(x_i, y_i) \in t_R} \left(y_i - y(t_R)\right)^2 \qquad (6)$$

donde n es el tamaño de la muestra e $y(t_L)$ e $y(t_R)$ son las estimaciones de la variable respuesta y para los datos en los nodos t_L y t_R, respectivamente. Normalmente, $y(t_L) = \frac{1}{n(t_L)} \sum_{(x_i, y_i) \in t_L} y_i$ e $y(t_R) = \frac{1}{n(t_R)} \sum_{(x_i, y_i) \in t_R} y_i$, donde $n(t)$ es el tamaño de la muestra que contiene t. En otras palabras, $y(t_L)$ e $y(t_R)$ son la media de los datos contenidos en cada nodo hoja. De esta manera, el árbol tendría un nodo raíz y dos nodos adicionales. Una vez que CART genera una división, el proceso completo se repite para los nodos hijos resultantes. El proceso continúa hasta que no se puedan hacer más divisiones significativas o se cumpla una regla de parada predefinida (la usual es $n(t) \leq 5 = n_{\min}$ para todos los nodos hoja).

Incluso en una situación multivariante como la tratada por CART, esta técnica admite una representación gráfica del predictor final por un simple árbol como el de la Figura 4.

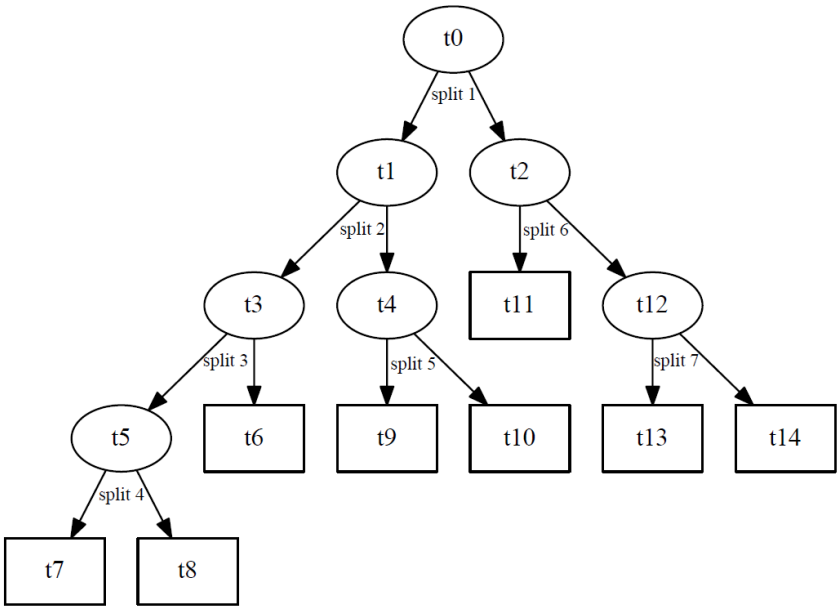


Figura 4. Ejemplo de una estructura de árbol CART.

Además, en dimensiones bajas, es posible dibujar el predictor a través de una función "escalonada" como la que se muestra en la Figura 5. El predictor es constante en cada nodo terminal. Obsérvese también que el predictor determinado por CART no envuelve los datos, sino que se ocupa del promedio de la variable de respuesta y, además, el predictor no satisface la propiedad de libre disponibilidad (o monotonidad). Estos hechos contrastan claramente con las características del predictor FDH (véase la Figura 2). Obviamente, CART no fue diseñado por Breiman et al. (1984) para tratar la estimación de fronteras de producción en microeconomía. Sin embargo, nótese que ambos enfoques, FDH y CART, generan funciones escalonadas como predictores.

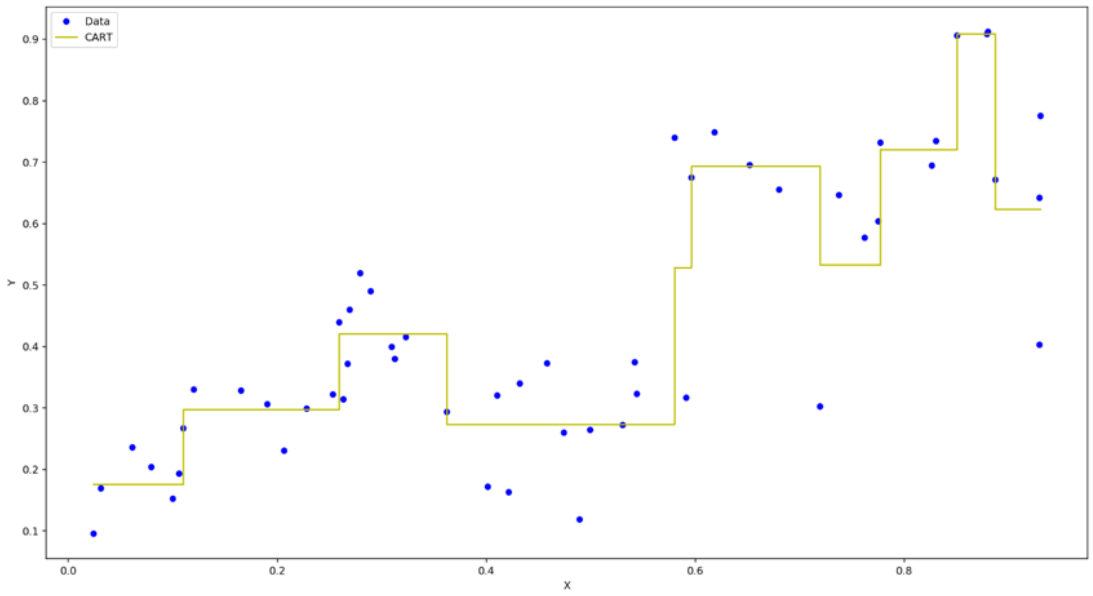


Figura 5. Función de predicción de CART.

Merece la pena mencionar que CART sufre de sobreentrenamiento o sobreajuste. El árbol que crece suele ser demasiado grande y las estimaciones que se hacen son demasiado optimistas. Incluso en el caso de que se utilice una restricción menos estricta como regla de parada, con un umbral para el tamaño de la muestra en cada nodo de la hoja mayor de cinco, se podría argumentar que este nivel prefijado es arbitrario. Además, en los árboles grandes ($n_{\min}$ pequeño) la precisión del enfoque es buena con respecto a la muestra de entrenamiento, ya que la función de paso asociada al predictor es muy cercana a los datos observados. Lamentablemente, el predictor depende demasiado de la muestra, lo que dificulta la generalización de los resultados y la obtención de una estimación adecuada. Esta fue una limitación claramente reconocida por Breiman et al. (1984) en su famoso libro. Para superar este problema sin asumir previamente ninguna distribución sobre el ruido estadístico, Breiman et al. (1984) propusieron podar el árbol de manera apropiada. Para ello, definieron la medida de complejidad del error $R_\alpha(T)$, que está basado en dos indicadores. La primera es una medida de la precisión del árbol, definida como la agregación del error cuadrático medio determinado en cada nodo hoja, mientras que la segunda es el número de nodos hoja como medida del tamaño o la complejidad del árbol.

Al mismo tiempo, un tercer elemento entra en juego: un parámetro α, que funciona como un peso para equilibrar los dos indicadores anteriores:

$$R_\alpha(T) = R(T) + \alpha\left|\tilde{T}\right| \tag{7}$$

donde $\tilde{T}$ es el conjunto de nodos hojas del árbol T, $R(T) = \frac{1}{n}\sum_{t\in\tilde{T}}\sum_{(x_i,y_i)\in t}\left(y_i - y(t)\right)^2$ y $|S|$ denotan la cardinalidad del conjunto S.

Ahora, la idea es podar minimizando la medida de la complejidad del error. El resultado es una secuencia de subárboles de tamaño decreciente $T_{\max} \succ T_1 \succ T_2 \succ ... \succ \{t_0\}$, donde $T_{\max}$ es el árbol obtenido mediante el uso de la regla de parada $n(t) \le n_{\min}$ y t_0 es el nodo raíz. Además, una secuencia creciente del valor de α está asociada a cada subárbol en la forma $0 = \alpha_1 < \alpha_2 < ...$ y satisfaciendo que α sea $\alpha_k \le \alpha < \alpha_{k+1}$. T_k denota el subárbol más pequeño de $T_{\max}$ que minimiza $R_\alpha(T)$.

Para puntuar cada subárbol en la secuencia y seleccionar el mejor, se suele recurrir a la validación cruzada. En la validación cruzada, la muestra de aprendizaje $\aleph$ se divide al azar en $\aleph_1,...,\aleph_V$, es decir dividiendo la muestra en submuestras de tamaño similar, siendo cada submuestra de aprendizaje $\aleph^{(v)} = \aleph - \aleph_v$. El proceso para obtener una puntuación para cada subárbol se basa en la repetición del procedimiento de crecimiento y poda de los árboles que se ha comentado anteriormente utilizando $\aleph^{(v)}$ en vez de $\aleph$. Para cada v, se producen los árboles $T^{(v)}(\alpha)$ que son aquellos que presentan el menor error de complejidad para el parámetro α, definido como $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$. Se denota como $d_k^{(v)}(x)$ al predictor que se corresponde con el árbol $T^{(v)}(\alpha'_k)$. Entonces, la puntuación

asociada al subárbol T_k en la secuencia original $T_{\max} \succ T_1 \succ T_2 \succ \ldots \succ \{t_0\}$ se denota de la siguiente forma:

$$R^{cv}(T_k) = \frac{1}{n}\sum_{v=1}^{V}\sum_{(x_i,y_i)\in \aleph_v}\left(y_i - d_k^{(v)}(x_i)\right)^2 \tag{8}$$

Finalmente, el subárbol seleccionado T_k es el árbol más pequeño que satisface la condición:

$$R^{cv}(T_k) \le R^{cv}(T_{k_0}) + SE \tag{9}$$

donde T_{k_0} es tal que $R^{cv}(T_{k_0}) = \min_k \{R^{cv}(T_k)\}$ y SE es la estimación del error estándar para $R^{cv}(T_{k_0})$.

En la Figura 6, se muestra cómo la forma del árbol final es determinada por CART después de que el proceso de poda haya sido implementado.

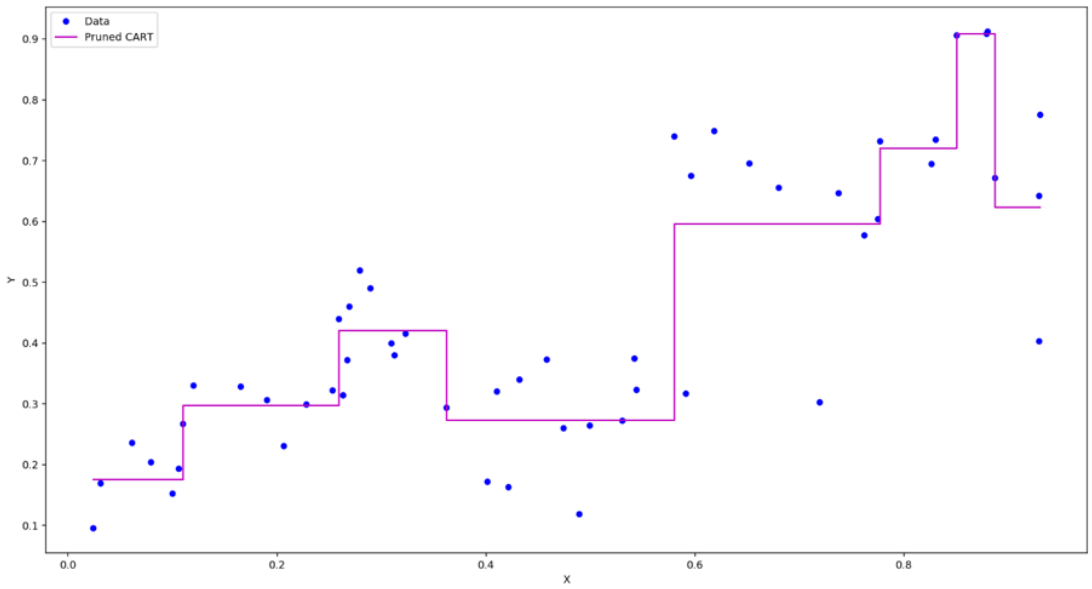


Figura 6. Función de predicción de CART podado.

A diferencia de los modelos tradicionales de regresión paramétrica, CART para la regresión no proporciona información sobre cuál es la relación entre cada variable x_j, $j = 1,...,m,$ con la variable respuesta. Mientras que los modelos paramétricos pueden proporcionar estimaciones puntuales y en intervalos de los coeficientes asociados a esa relación, CART no lo hace. No obstante, Breiman et al. (1984) también sugirieron cómo proporcionar al menos una clasificación de las variables x_j dependiendo de la importancia de cada variable. A priori, no es trivial cómo clasificar estas variables ya que al no dar la mejor división en un nodo, podría ser la segunda o tercera mejor opción. Por ejemplo, en el árbol final seleccionado, x_j podría no ocurrir en ninguna división. Sin embargo, si una cierta variable $x_{j'}$ se elimina del análisis y se construye otro árbol, entonces x_j podría aparecer de forma prominente en las divisiones, incluso con un árbol resultante tan preciso como el original. En tal contexto, se necesita que el método de clasificación detecte la importancia de x_j en el árbol original construido. Con tal fin, Breiman et al. (1984) propusieron un procedimiento para un ranking adecuado, jugando con la noción de las particiones alternativas.

Por otra parte, muchas son las propuestas en la literatura sobre cómo extender las técnicas de árboles de decisión con una sola variable respuesta al contexto de múltiples variables respuesta: De'ath, 2002; Appice y Dzeroski, 2007; Stojanova et al., 2012; y Levatic et al., 2014, por nombrar alguno. De estos, De'ath (2002), parece ser la extensión natural del método de partición recursivo univariado de CART (véase Borchani et al., 2015). Este enfoque funciona siguiendo los mismos pasos que CART, es decir, comenzando con todos los datos en el nodo raíz, y luego recursivamente encontrando la mejor división posible en los nodos padres y dividiendo los datos en consecuencia hasta que se satisfaga una cierta regla de parada. La principal diferencia con CART estándar es la redefinición de la medida utilizada para seleccionar la mejor combinación $\left(x_{j^*}^*, s_{j^*}^*\right)$ en cada división. En otras palabras, en el caso de la respuesta única el criterio utilizado coincide con el error cuadrático medio asociado a la variable de respuesta única, mientras que en el caso

de la respuesta múltiple los errores cuadráticos vinculados a todas las variables de respuesta se agregan en una medida final. Por último, otra diferencia es que cada hoja del árbol generada por el enfoque de De'ath (2002) se caracteriza por la media multivariante de los datos que le pertenecen, como una generalización directa del estándar univariante de CART. De'ath (2002) también indicó algunas características interesantes de la técnica. Los árboles multivariantes son fáciles de construir y visualizar. También son robustos a la adición de ruido aleatorio. Además, detectan automáticamente las interacciones entre las variables. Por último, manejan los valores perdidos en las variables predictoras (los inputs en nuestro contexto) con una mínima pérdida de información.

3.3. RANDOM FOREST

Random Forest es un método de aprendizaje por agregación que funciona construyendo una multitud de árboles de decisión en el momento de la formación y agregando la información de los árboles individuales en un valor de predicción final (la media en el problema de regresión) (Breiman, 2001). El algoritmo de entrenamiento para el Random Forest aplica una doble técnica de aleatorización: sobre los datos (a través del bootstrapping) y sobre la selección de los predictores. Dada una muestra de aprendizaje $\aleph$ de tamaño n, Random Forest repetidamente selecciona de forma aleatoria una muestra de tamaño n con reemplazamientos del conjunto $\aleph$. Entonces, el método ajusta los árboles CART a estas muestras pero, para ello, recurre a un algoritmo de aprendizaje de árbol modificado que elige, en cada división candidata del proceso de aprendizaje, un subconjunto aleatorio de los predictores. La razón de hacer esto es intentar descorrelacionar los árboles individuales que pueden ser estimados a partir de muestras ordinarias de bootstrap: si un determinado subconjunto de predictores es muy fuerte para predecir la variable de respuesta, estos mismos predictores serán seleccionados en muchos de los árboles individuales analizados, haciendo que se correlacionen. La agregación (no correlacionada) de modelos aleatorios reduce el error de predicción al disminuir el término de la varianza; la combinación de varios modelos aleatorios logra un mejor rendimiento que un solo modelo no aleatorio (Berk, 2016).

Los pasos típicos que se deben llevar a cabo en Random Forest se muestran en el Algoritmo 1 (véase Kuhn y Johnson, 2013). En primer lugar, el investigador elige el número de árboles que se van a ajustar (el parámetro p). En la práctica, suele bastar con considerar 500 ó 1000. En segundo lugar, p muestras de boostrap deben generarse, siendo cada una de ellas denotada como $\aleph_q$, $q = 1,...,p$. Para cada muestra boostrap $\aleph_q$, un árbol $T^{CART}\left(\aleph_q\right)$ es entrenado por el algoritmo CART. Sin embargo, en cada división, el algoritmo ya no considera todos los posibles predictores y sus umbrales. En su lugar, $mtry$ predictores se seleccionan cada vez al azar y pasan a ser las variables que se consideran para dividir el nodo padre correspondiente en dos nodos hijos. En el caso de Random Forest, no es necesario podar cada árbol. El proceso de poda por validación cruzada se sustituye por la doble técnica de aleatorización, sobre los datos y sobre la selección de los predictores, y la agregación final de las predicciones de todos los árboles. En general, los árboles más grandes pueden dar lugar a un menor sesgo en la predicción, mientras que la agregación de modelos de árboles de decisión aleatorios no correlacionados reduce la varianza del predictor.

Una vez ajustados todos los árboles a través de la aplicación del Algoritmo 1, la predicción del valor de la variable respuesta a partir de un vector de valores de los predictores, que no es necesariamente un vector de valores observado, se determina promediando la predicción individual correspondiente a cada árbol.

Una característica atractiva de los métodos de agregación en aprendizaje automático, como Random Forest, que construye modelos sobre muestras bootstrap, es el uso de las muestras complementarias $\aleph \setminus \aleph_q$ para formar estimaciones de ciertas medidas de interés, como el error de predicción, también conocido como error de generalización en la teoría del aprendizaje estadístico. En particular, Random Forest suele explotar la noción de *Out-Of-Bag* (OOB). El *Out-Of-Bag* estima el valor de la variable respuesta para la

observación $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ utilizando sólo los modelos individuales $T^{CART}(\aleph_q)$ que corresponden a las muestras bootstrap $\aleph_q$ tal que $(\boldsymbol{x}_i, \boldsymbol{y}_i) \notin \aleph_q$. A partir de esta definición, el error de predicción se define como el promedio de las estimaciones del *Out-Of-Bag* calculadas sobre todas las observaciones de la muestra de aprendizaje $\aleph$:

$$err^{RF+CART(\aleph)} = \frac{1}{n} \sum_{(x_i, y_i) \in \aleph} \sum_{r=1}^{s} \left(y_{ri} - y_r^{RF+CART(\aleph)}(\boldsymbol{x}_i) \right)^2 \tag{10}$$

donde $y_r^{RF+CART(\aleph)}(\boldsymbol{x}_i)$ es el componente r-ésimo del vector $\boldsymbol{y}^{RF+CART(\aleph)}(\boldsymbol{x}_i) = \frac{1}{|K_i(\aleph)|} \sum_{q \in K_i(\aleph)} \boldsymbol{d}_{T^{CART}(\aleph_q)}(\boldsymbol{x}_i), \qquad K_i(\aleph) = \{ q : q = 1,..., p, (\boldsymbol{x}_i, \boldsymbol{y}_i) \notin \aleph_q \}.$

$K_i(\aleph)$ representa el conjunto de índices entre p árboles entrenados de tal forma que sus correspondientes submuestras no incluyen $(\boldsymbol{x}_i, \boldsymbol{y}_i)$.

Input of the algorithm:

p, number of trees

$\aleph$, original data with m predictors and s outputs

mtry, number of predictors to select (using a specific rule)

n_{min}, max number of observations in a leaf node (5 by default)

Output of the algorithm:

$RF^{CART} := \{ T^{CART}(\aleph_q): q = 1, \ldots, p \}$

$\{ \aleph_q : q = 1, \ldots, p \}$

$RF^{CART} := \{\}$

FOR q: = 1 **TO** p

- $\aleph_q$:= Bootstrap sample of $\aleph$
- $T^{CART}(\aleph_q) := \{ t_0 : t_0 = \text{root node} \}$
- **FOR EACH** t in $T^{CART}(\aleph_q)$: t is a leaf node $\Leftrightarrow n(t) \leq n_{min}$
  - Randomly select *mtry* ($\leq m$) of the original predictors
  - Select the best predictor among the *mtry* predictors for t and split the data in t_L and t_R nodes
  - $T^{CART}(\aleph_q) := T^{CART}(\aleph_q) \cup \{ t_L , t_R \}$
- **END FOR**
- $RF^{CART} := RF^{CART} \cup \{ T^{CART}(\aleph_q) \}$

END FOR

Algoritmo 1. Random Forest.

El error de predicción es útil, por ejemplo, para determinar una medida de importancia de las variables predictoras, que puede utilizarse para crear una clasificación o ranking de predictores $x_1,...,x_m$. Para calcular la importancia del predictor x_j, se pueden seguir los siguientes pasos. Generar una nueva base de datos, $\aleph^j$, idéntica a la original $\aleph$, donde específicamente los valores de la variable x_j se han permutado aleatoriamente. Este proceso de "barajado" hace que ese predictor, en promedio, no esté relacionado con las variables de respuesta y todos los demás predictores. Aplicar la técnica de Random Forest sobre la nueva muestra de aprendizaje "virtual" $\aleph^j$. Determinar el valor del error de predicción vinculado a este último Random Forest: $err^{RF+CART(\aleph^j)}$. Finalmente, es posible calcular cuánto porcentaje aumenta el error de predicción del modelo cuando la variable x_j es barajada como $\%Inc^{RF+CART}\left(x_j\right)=100\cdot\left(\frac{err^{RF+CART(\aleph^j)}-err^{RF+CART(\aleph)}}{err^{RF+CART(\aleph)}}\right)$.

CAPÍTULO 4: RESULTADOS

4.1. ÁRBOLES DE ANÁLISIS DE EFICIENCIA

En este capítulo se introduce la nueva técnica basada en la adaptación de CART para estimar las fronteras de producción, que se ha bautizado como Árboles de Análisis de Eficiencia (*Efficiency Analysis Trees,* en inglés). De aquí en adelante nos referiremos a esta nueva técnica mediante el acrónimo EAT. Este nuevo enfoque permite determinar fronteras de producción satisfaciendo los axiomas habituales de la microeconomía mediante un modelo basado en datos que no asume ninguna distribución de probabilidad particular y, además, genera una función escalonada como predictor. Estas características son compartidas con la técnica de FDH. Sin embargo, mientras que FDH sufre de sobreajuste, el nuevo método supera este problema mediante la validación cruzada y la poda, propuesta en Breiman et al. (1984), que explota su estructura natural de árbol. Otra diferencia notable entre estas dos técnicas, EAT y FDH, es que el crecimiento del árbol, en el caso de EAT, se lleva a cabo de manera estadística, es decir, minimiza el error cuadrático medio y utiliza reglas de parada ligadas al tamaño de la muestra, y evita que se produzcan nodos terminales vacíos. Un nodo vacío implica una región en el espacio de inputs sin datos, pero con una estimación de la variable de respuesta (el output). En el caso de FDH, sí que puede producir este tipo de regiones, como se muestra más adelante,

un hecho algo extraño desde una perspectiva basada en los datos, ya que no es posible calcular una medida del error en ese nodo en particular.

A continuación, el nuevo enfoque EAT se divide en dos secciones, una para el caso de una única variable respuesta y otra para el caso de múltiples variables respuesta. Se añade otra sección a este capítulo para mostrar las bondades de la nueva metodología a partir de una experiencia computacional.

4.1.1. EL CASO DE UNA ÚNICA VARIABLE RESPUESTA

En el Aprendizaje Estadístico Supervisado convencional, todo comienza con una muestra de aprendizaje $\aleph$, que consta con ejemplos de $(x_1, y_1),...,(x_n, y_n)$, con $x_i \in R^m$ e $y_i \in R$, $i = 1,...,n$, como en CART. Ahora, comparando este nuevo enfoque con Breiman (1984), hay tres elementos que son clave para determinar el árbol predictor:

1. Una regla para asignar una estimación de la variable respuesta a cada nodo: $y(t)$.
2. Una forma de seleccionar una partición en cada nodo intermedio.
3. Una regla de parada para determinar cuándo un nodo es terminal.

Además, es necesario añadir dos claves más a nuestra nueva técnica de producción:

4. $y(t)$ debe estimar la frontera de producción en lugar de la media de la variable respuesta. Esta característica se vincula fácilmente con el punto 1 anterior.
5. La satisfacción de la libre disponibilidad. Este punto es probablemente el más difícil de abordar, por ello en esta sección se propone una forma de seleccionar el siguiente nodo que debe dividirse en el algoritmo de crecimiento del árbol y se sugiere una forma de modificar la estimación del output en cada nodo.

Se empieza con los puntos 1 y 4 proponiendo una estimación envolvente adecuada para la variable respuesta en el nodo t: $y(t)$. En este sentido, el valor $y(t)$ es aquél que minimiza $R(t)=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-y(t)\right)^2$ en el nodo t, es decir, la suma de los cuadrados del nodo interior, y, al mismo tiempo, satisface $y(t)\geq y_i,\ \ \forall(\boldsymbol{x}_i,y_i)\in t,$ que es el valor máximo de la variable respuesta observada en la muestra de datos perteneciente al nodo t. Este resultado se demuestra en la Proposición 4.1:

Proposición 4.1. El valor de $y(t)$ que minimiza $R(t)=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-y(t)\right)^2$ sujeto a $y(t)\geq y_i$, $\forall(\boldsymbol{x}_i,y_i)\in t$, es $y(t)=\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}$.

Demostración. Se asume que $y(t)=\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}+\delta,$ con $\delta>0.$ Entonces, $\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-y(t)\right)^2=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-\left(\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}+\delta\right)\right)^2=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(\left(y_i-\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}\right)-\delta\right)^2=$ $\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}\right)^2+\underbrace{\delta^2\frac{n(t)}{n}-2\delta\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}\right)}_{=A}.$ Notar que $A\geq 0$ como $y_i\leq\max_{(\boldsymbol{x}_k,y_k)\in t}\{y_k\},\quad\forall(\boldsymbol{x}_i,y_i)\in t.$ Por lo tanto, $\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-\left(\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}+\delta\right)\right)^2\geq\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t}\left(y_i-\max_{(\boldsymbol{x}_i,y_i)\in t}\{y_i\}\right)^2,\ \ \forall\delta>0,$ que prueba esta proposición. ■

Volviendo al punto 2, se asume que hay un nodo que debe ser dividido. Además, se recuerda que, en esta situación, CART estándar selecciona una variable predictora j, $j=1,...,m$, y un umbral $s_j\in S_j$, donde S_j es el conjunto de posibles umbrales para la variable j, tal que se minimiza la suma del error cuadrático medio (MSE) calculado para los datos pertenecientes al nodo hijo izquierdo, es decir, los datos que cumplen la

condición $x_j < s_j$, y el MSE determinado con los datos que pertenecen al nodo hijo derecho, es decir, que cumplen la condición $x_j \geq s_j$. Si t es la forma de denotar al nodo padre, entonces t_L y t_R son los nodos hijos izquierdo y derecho, respectivamente. Entonces, CART selecciona la mejor combinación $\left(x_j, s_j\right)$ que minimiza la expresión (6). EAT sigue el mismo criterio cuando divide un nodo intermedio.

Adicionalmente, un nodo terminal o final en EAT (punto 3) es aquél que satisface cierta regla de parada, como $n(t) \leq n_{\min}$. En este sentido, Breiman (1984) recomienda $n_{\min} = 5$. Otra regla de parada natural es la asociada con la situación en la que todas las observaciones de un nodo comparten los mismos valores para todas los inputs (comparten el mismo perfil de entrada). En este caso, no es posible dividir el nodo. En esta tesis se usa estas condiciones para las experiencias computacionales efectuadas. Sin embargo, con el propósito de buscar la simplicidad en el desarrollo de la parte metodológica, se asume en esta sección que no todas las observaciones en un nodo comparten el mismo perfil de entrada.

Como se ha mencionado anteriormente, probablemente la tarea más difícil es la relacionada con la definición de un algoritmo que divida iterativamente los nodos minimizando (6) y, al mismo tiempo, garantice la satisfacción de la propiedad de libre disponibilidad. En este nuevo enfoque, la clave está en el algoritmo de división de cada nodo intermedio. Antes de mostrar el algoritmo, es necesario introducir algunas definiciones y notaciones adicionales. T_k es el árbol creado después de la *k*-ésima división. $\tilde{T}_k$ es el conjunto de nodos terminales (u hojas) del árbol T_k. Por ejemplo, T_1 para el ejemplo gráfico de la Figura 4 es el (sub)árbol compuesto por $\{t_0, t_1, t_2\}$ con los nodos terminales t_1 y t_2. Para este mismo ejemplo, T_2 es el árbol $\{t_0, t_1, t_2, t_3, t_4\}$ con el conjunto de nodos terminales $\tilde{T}_2 = \{t_2, t_3, t_4\}$.

En esta estructura, cada nodo t se define por la satisfacción de una serie de condiciones en el espacio de input del tipo $\{x_j < s_j\}$ o $\{x_j \geq s_j\}$ (condiciones que se asocian con cada división). Por lo tanto, después de ejecutar un cierto número de divisiones, hay una región en el espacio de input en forma de un segmento si $m=1$, un rectángulo si $m=2$, un paralelepípedo si $m=3$, etc. Este es el soporte de un nodo t:

Definición 4.1. Siendo $k=1,...,K$. Teniendo $t \in \tilde{T}_k$. Entonces, el soporte de un nodo t se denota y define como $\text{supp}(t) = \{\boldsymbol{x} \in R_+^m : a_j^t \leq x_j < b_j^t, j=1,...,m\}$.

El parámetro a_j^t puede ser cero, mientras que el parámetro b_j^t puede ser $+\infty$. En la práctica, a_j^t está limitado por abajo por la observación más pequeña de la variable x_j en la muestra de aprendizaje, es decir, $a_j^t = \min_{1 \leq i \leq n} \{x_{ji}\}$. Notar también que si $(\boldsymbol{x}_i, y_i) \in t$, entonces $\boldsymbol{x}_i \in \text{supp}(t)$. Lo contrario también es cierto si $\boldsymbol{x}_i \in \text{supp}(t)$, entonces $(\boldsymbol{x}_i, y_i) \in t$ porque las divisiones producen subconjuntos disjuntos de la muestra de aprendizaje original.

Otra noción necesaria es la del conjunto de nodos Pareto dominantes dado un nodo $t \in \tilde{T}_k$. Esta definición está relacionada con la satisfacción de la propiedad de libre disponibilidad de la frontera de producción estimada a través de EAT.

Definición 4.2. Dado $k=1,...,K$. Teniendo $t \in \tilde{T}_k$. Entonces, el conjunto de nodos Pareto dominantes por un nodo t se denota y define como $I_{T_k}(t) = \{t' \in \tilde{T}_k - t : \exists \boldsymbol{x} \in \text{supp}(t), \exists \boldsymbol{x}' \in \text{supp}(t') \text{ tal que } \boldsymbol{x}' \leq \boldsymbol{x}\}$.

Un elemento en $I_{T_k}(t)$ es un nodo con al menos un vector de input en su correspondiente soporte, no necesariamente observado en la muestra de aprendizaje, de tal manera que

domina al menos un vector de input perteneciente al soporte del nodo t en el sentido de Pareto. Esta noción se explota para la satisfacción de la propiedad de libre disponibilidad.

Comprobar si un nodo $t' \in \tilde{T}_k$ pertenece a $I_{T_k}(t)$ es una tarea sencilla ya que basta con comparar las componentes de dos vectores, $\boldsymbol{a}^{t'}$ y $\boldsymbol{b}^t$, tal y como se muestra en la siguiente proposición.

Proposición 4.2. $t' \in I_{T_k}(t)$ si y solo si $\boldsymbol{a}^{t'} < \boldsymbol{b}^t$.

Demostración. Se asume que $t' \in I_{T_k}(t)$. Entonces, por la Definición 4.2, $\exists \boldsymbol{x} \in \text{supp}(t)$, $\exists \boldsymbol{x}' \in \text{supp}(t')$ tal que $\boldsymbol{x}' \leq \boldsymbol{x}$. Por la Definición 4.1, $\boldsymbol{a}^{t'} \leq \boldsymbol{x}'$ y $\boldsymbol{x} < \boldsymbol{b}^t$. Por lo tanto, $\boldsymbol{a}^{t'} < \boldsymbol{b}^t$. Ahora se asume que $\boldsymbol{a}^{t'} < \boldsymbol{b}^t$. Observar también que $\boldsymbol{a}^{t'} \in \text{supp}(t')$ por la Definición 4.1. Este no es el caso de $\boldsymbol{b}^t$ respecto a $\text{supp}(t)$. Se busca un punto $\tilde{\boldsymbol{x}}$ en $\text{supp}(t)$ tal que $\boldsymbol{a}^{t'} \leq \tilde{\boldsymbol{x}}$. Se puede aproximar a $\boldsymbol{b}^t$, desde el interior de $\text{supp}(t)$, siguiendo el camino asociado a la consideración de puntos definidos como la combinación convexa de $\boldsymbol{a}^{t'}$ y $\boldsymbol{b}^t$: $\lambda \boldsymbol{a}^{t'} + (1-\lambda)\boldsymbol{b}^t$, $\lambda \in (0,1)$. Para ello, basta con tomar valores de *lambda* cercanos a cero. Notar que $\boldsymbol{a}^{t'} \leq \lambda \boldsymbol{a}^{t'} + (1-\lambda)\boldsymbol{b}^t$. Por tanto, $\exists \tilde{\lambda} \in (0,1)$ tal que $\boldsymbol{a}^{t'} \leq \tilde{\boldsymbol{x}}$, con $\tilde{\boldsymbol{x}} = \lambda \boldsymbol{a}^{t'} + (1-\lambda)\boldsymbol{b}^t$ y $\tilde{\boldsymbol{x}} \in \text{supp}(t)$. Consecuentemente, $t' \in I_{T_k}(t)$. ■

En la Figura 7 se muestra una ilustración de la Proposición 4.2. Dos soportes asociados a los nodos t y t' aparecen en ella. Claramente, se sostiene la hipótesis de la proposición, es decir, $\boldsymbol{a}^{t'} < \boldsymbol{b}^t$. Consecuentemente, El nodo t está Pareto dominado por el nodo t'.

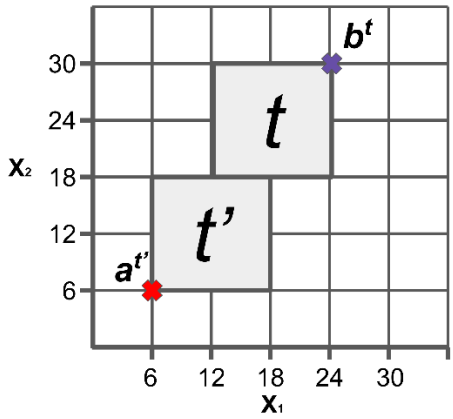


Figura 7. Ejemplo de Pareto dominancia de nodos. El nodo t' Pareto domina al nodo t.

Por otra parte, y volviendo a la notación, dado $t^* \in \tilde{T}_k$ tal que este nodo no satisface ninguna regla de parada y, además, es posible dividirlo en dos nodos hijos t_L y t_R, entonces el árbol asociado con esta división específica viene denotado como $T\left(k \mid t^* \to t_L, t_R\right)$. Notar que, en $T\left(k \mid t^* \to t_L, t_R\right)$, $t \notin \tilde{T}\left(k \mid t^* \to t_L, t_R\right)$. En efecto, $\tilde{T}\left(k \mid t^* \to t_L, t_R\right) = \left(\tilde{T}_k - t^*\right) \cup \{t_L, t_R\}$.

Seguidamente, se muestra cómo implementar bajo este nuevo enfoque el proceso de división dado un nodo $t^* \in \tilde{T}_k$ que no satisface ninguna regla de parada. Primero de todo, para todas las posibles combinaciones para (dividir) la variable x_j y los umbrales $s_j \in S_j$, es decir, $\left(x_j, s_j\right)$, el algoritmo debe determinar la suma de los errores cuadráticos medios (MSE) asociados a la división del nodo t^* en un nodo izquierdo y un nodo derecho con sus correspondientes estimaciones de la variable respuesta y. Después de eso, se obtiene la mejor combinación posible $\left(x_{j^*}^*, s_{j^*}^*\right)$: aquella que minimiza la suma de los errores. Para estimar la variable respuesta en cada nodo hijo, dada una combinación particular $\left(x_j, s_j\right)$, se sugiere combinar la Proposición 4.1 con el concepto de Pareto dominancia

de un nodo. Como se muestra más adelante, este punto será la clave para garantizar la libre disponibilidad. Se empieza con la descripción de este nuevo algoritmo con la evaluación del nodo hijo izquierdo, t_L, dada la combinación $\left(x_j, s_j\right)$. Primero, el algoritmo determina el conjunto de nodos que Pareto dominan a t_L, es decir, $I_{T\left(k|t\to t_L,t_R\right)}\left(t_L\right)$. Una vez que se conocen los elementos de este conjunto, se puede calcular la estimación de la variable respuesta y (el output) para el nodo t_L. En este nuevo enfoque, se fuerza a que la estimación nunca sea menor que las estimaciones de la variable respuesta en los nodos pertenecientes al conjunto de nodos que Pareto dominan a t_L. Así se consigue satisfacer la libre disponibilidad de una manera constructiva. Por lo tanto, se sugiere la siguiente estimación de la variable respuesta para el nodo hijo t_L:

$$y\left(t_L\right)=\begin{cases}\max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_L\right\}, & \text{si } \max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_L\right\}\geq y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right)\\ y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right), & \text{en otro caso}\end{cases} \quad (11)$$

donde $y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right)=\max\left\{y\left(t'\right):t'\in I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right\}$ e $y\left(t'\right)$ es la estimación de la variable respuesta de $t'\in\tilde{T}\left(k\,|\,t^*\to t_L,t_R\right)$. En otras palabras, $y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right)$ es la estimación más grande de la variable y de los nodos que Pareto dominan a t_L. La expresión (11) se puede reescribir de una manera más compacta como $y\left(t_L\right)=\max\left\{\max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_L\right\},y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)\right)\right\}$.

En cuanto al nodo hijo derecho, la expresión sería muy similar a la anterior.

$$y\left(t_R\right)=\begin{cases}\max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_R\right\}, & \text{si } \max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_R\right\}\geq y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_R\right)\right)\\ y\left(I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_R\right)\right), & \text{en otro caso}\end{cases} \quad (12)$$

Continuando con este análisis, se puede calcular el error cometido cuando y es la estimación $y(t_L)$ para el nodo t_L y cuando y es la estimación $y(t_R)$ para el nodo t_R, es decir, $R(t_L)=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t_L}\left(y_i-y(t_L)\right)^2$ y $R(t_R)=\frac{1}{n}\sum_{(\boldsymbol{x}_i,y_i)\in t_R}\left(y_i-y(t_R)\right)^2$, respectivamente. La suma de estos dos errores es la que hay que minimizar. Así pues, $\left(x_{j^*}^*, s_{j^*}^*\right)$ está definida como la combinación de las variables x_j y los umbrales $s_j \in S_j$, $j=1,...,m,$ tal que $R(t_L)+R(t_R)$ es minimizado. También, t_L^* y t_R^* son los nodos hijos asociados a la división $\left(x_{j^*}^*, s_{j^*}^*\right)$. De esta manera, se tiene que $T_{k+1}=T\left(k \mid t^* \to t_L^*, t_R^*\right)$..

Una vez que se ha obtenido el árbol $T\left(k \mid t^* \to t_L^*, t_R^*\right)$, la forma de actualizar los parámetros a y b correspondientes a los soportes de t_L^* y t_R^* es la siguiente:

$$
\begin{aligned}
a_j^{t_L} &= a_j^{t^*}, \quad \forall j=1,...,m \\
b_j^{t_L} &= b_j^{t^*}, \quad \forall j=1,...,m, j\neq j^* \\
b_{j^*}^{t_L} &= s_{j^*}^*, \\
a_j^{t_R} &= a_j^{t^*}, \quad \forall j=1,...,m, j\neq j^* \\
a_{j^*}^{t_R} &= s_{j^*}^*, \\
b_j^{t_R} &= b_j^{t^*}, \quad \forall j=1,...,m
\end{aligned}
\tag{13}
$$

La expresión (12) puede ser sustituida simplemente por $y\left(t_R\right)=y(t^*),$ donde $y(t^*)$ es la estimación del nodo padre t, como muestra la siguiente proposición, tras demostrar algún resultado necesario previo.

Lema 4.1. Si $t' \in I_{T\left(k|t^*\to t_L,t_R\right)}\left(t_L\right)$, entonces $t' \in I_{T_k}\left(t^*\right)$.

Demostración. Notar que, por definición, $\tilde{T}\left(k \mid t^* \to t_L, t_R\right) = \left(\tilde{T}_k - t^*\right) \cup \{t_L, t_R\}$. Además, $t' \in I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_L\right)$ si y solo si $\boldsymbol{a}^{t'} < \boldsymbol{b}^{t_L}$ (ver Proposición 4.2). Por (13), $\boldsymbol{b}^{t_L} \leq \boldsymbol{b}^{t^*}$. Esto implica que $\boldsymbol{a}^{t'} < \boldsymbol{b}^{t^*}$. Por tanto, $t' \in I_{T_k}\left(t^*\right)$. ■

Corolario 4.1. $y\left(t_L\right) \leq y\left(I_{T_k}\left(t^*\right)\right)$.

Demostración. Por el Lema 4.1, $I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_L\right) \subset I_{T_k}\left(t^*\right)$. Entonces, $y\left(I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_L\right)\right) \leq y\left(I_{T_k}\left(t^*\right)\right)$. Además, $\max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t^*\right\} \geq \max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t_L\right\}$. Por el algoritmo EAT, se sabe que $y\left(t^*\right) = \max\left\{\max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t^*\right\}, y\left(I_{T_k}\left(t^*\right)\right)\right\}$. Finalmente, por la forma más compacta de (11), se tiene que $y\left(t_L\right) \leq y\left(I_{T_k}\left(t^*\right)\right)$. ■

Corolario 4.2. $y\left(t_L\right) \leq y\left(t^*\right)$.

Demostración. Por el algoritmo EAT, $y\left(t^*\right) = \max\left\{\max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t^*\right\}, y\left(I_{T_k}\left(t^*\right)\right)\right\}$. Por el Corolario 4.1, $y\left(t_L\right) \leq y\left(I_{T_k}\left(t^*\right)\right)$. Por tanto, $y\left(t^*\right) = \max\left\{\max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t^*\right\}, y\left(I_{T_k}\left(t^*\right)\right)\right\} \geq y\left(t_L\right)$. ■

Proposición 4.3. $y\left(t_R\right) = y\left(t^*\right)$.

Demostración. $I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_R\right) = I_{T_k}\left(t^*\right) \cup \{t_L\}$ porque, por (13), los nodos t_R y t^* comparten el mismo parámetro $\boldsymbol{b}$ ($b_j^{t_R} = b_j^{t^*}$, $\forall j = 1, \ldots, m$) y $\boldsymbol{a}^{t_L} = \boldsymbol{a}^{t^*} < \boldsymbol{b}^{t^*}$. Entonces, por (11), $y\left(t_R\right) = \max\left\{\max\left\{y_i : \left(\boldsymbol{x}_i, y_i\right) \in t_R\right\}, y\left(I_{T_k}\left(t^*\right)\right), y\left(t_L\right)\right\}$. Adicionalmente, por el

algoritmo EAT, se sabe que $y(t^*) = \max\left\{\max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\}, y\left(I_{T_k}(t^*)\right)\right\}$. Se tiene dos escenarios. Escenario A: $y(t^*) = y\left(I_{T_k}(t^*)\right)$. De esta forma, se tiene que $y\left(I_{T_k}(t^*)\right) \geq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\}$. Si $y\left(I_{T_k}(t^*)\right) \geq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\}$, entonces $y\left(I_{T_k}(t^*)\right) \geq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\}$, porque $\max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\} \geq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\}$. Además, por el Corolario 4.1, $y\left(I_{T_k}(t^*)\right) \geq y(t_L)$. Por tanto, $y(t_R) = y\left(I_{T_k}(t^*)\right) = y(t^*)$, como se quería probar. Escenario B: $y(t^*) = \max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\}$. Por lo tanto, $\max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\} \geq y\left(I_{T_k}(t^*)\right)$. Se considera dos escenarios adicionales: B1 y B2. Escenario B1: $\max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\} = \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_L\}$. Por el Lema 4.1, $I_{T(k|t^* \to t_L, t_R)}(t_L) \subset I_{T_k}(t^*)$. Entonces, $y\left(I_{T(k|t^* \to t_L, t_R)}(t_L)\right) \leq y\left(I_{T_k}(t^*)\right)$. Como consecuencia, se tiene que $y\left(I_{T(k|t^* \to t_L, t_R)}(t_L)\right) \leq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\} = \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_L\}$. Por tanto, por (12), $y(t_L) = y(t^*)$. Ahora, se consigue $y(t_R) = y(t^*)$ desde $y(t_R) = \max\left\{\max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\}, y\left(I_{T_k}(t^*)\right), y(t_L)\right\}$ y $\max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\} \leq \max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\}$. Escenario B2: $\max\{y_i : (\boldsymbol{x}_i, y_i) \in t^*\} = \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\}$. Por el Corolario 4.2, $y(t_L) \leq y(t^*)$. Por tanto, $y(t_R) = \max\left\{\max\{y_i : (\boldsymbol{x}_i, y_i) \in t_R\}, y\left(I_{T_k}(t^*)\right), y(t_L)\right\} = y(t^*)$. ■

Después de realizar la división T_{k+1}, se tiene una estimación de la variable respuesta para cada nodo que pertenece a $\tilde{T}_{k+1}$. En particular, se tiene que $y_{T_{k+1}}(t) = y_{T_k}(t)$ si $t \neq t^*$ y, por (11) y (12), $y_{T_{k+1}}(t_L^*) = y(t_L^*)$ y $y_{T_{k+1}}(t_R^*) = y(t_R^*)$. Además, se define la estimación de

la variable respuesta correspondiente al nodo raíz t_0 como $y_{T(0)}(t_1) = \max\{y_i : (\boldsymbol{x}_i, y_i) \in t_1\} = \max\{y_i : i = 1,...,n\}$, es decir, esta estimación coincide con el máximo valor observado para la variable respuesta en la muestra de aprendizaje.

En este nuevo enfoque, la forma de seleccionar el nodo que va a ser dividido entre todos los nodos pertenecientes a $\tilde{T}_k$ consiste en escoger siempre el nodo hijo izquierdo que no cumpla ninguna regla de parada (esta estrategia será estudiada en más detalle en el Capítulo 4 Sección 4.3.). Por otra parte, hay muchas formas de definir el conjunto de umbrales S_j para cada variable x_j, $j = 1,...,m$. En este nuevo algoritmo, todos los elementos del conjunto S_j son los observados para las variables x_j en los datos pertenecientes al nodo t.

El proceso se repite hasta que se realizan todas las posibles divisiones, de tal manera que cada nodo terminal satisface la regla de parada. Sea K el número de divisiones ejecutadas en el proceso de crecimiento del árbol que corresponde a la última división realizada y sea T_K el árbol final. Así pues, $T_{\max} = T_K$.

Ahora, es el momento de mostrar el principal resultado de este trabajo, en el que se establece que el predictor asociado a la estructura del árbol $T_{\max}$ cumple con el axioma de libre disponibilidad (traducido a monotonía en el caso mono-output). En particular, el predictor $d(\boldsymbol{x})$ definido desde la información que contiene $T_{\max}$ tiene la siguiente forma: $d_{T_{\max}}(\boldsymbol{x}) = y_{T_{\max}}(t)$, para $t \in \tilde{T}_{\max}$ tal que $\boldsymbol{a}^t \leq \boldsymbol{x} < \boldsymbol{b}^t$, es decir, $\boldsymbol{x} \in \text{supp}(t)$.

Teorema 4.1. $d_{T_{\max}}(\boldsymbol{x}) : R_+^m \to R$ es una función monótona no decreciente.

Demostración. Se tiene que probar que si $\boldsymbol{x} \le \boldsymbol{x}'$, entonces $d_{T_{\max}}(\boldsymbol{x}) \le d_{T_{\max}}(\boldsymbol{x}')$. Primero, se prueba que si $d_{T_k}(\boldsymbol{x})$ es una función monótona no decreciente, entonces $d_{T_{k+1}}(\boldsymbol{x})$ es también una función monótona no decreciente. La diferencia entre los árboles T_k y T_{k+1} está en que un nodo de $\tilde{T}_k$, llamado t^*, ha sido dividido en dos nodos hijos, t_L^* y t_R^*. De esta manera, $T_{k+1} = T\left(k \mid t^* \to t_L^*, t_R^*\right)$. Además, $d_{T_{k+1}}(\boldsymbol{x}) = d_{T_k}(\boldsymbol{x})$ para todo $\boldsymbol{x} \in \left(R_+^m - \text{supp}\left(t^*\right)\right)$ y $d_{T_{k+1}}(\boldsymbol{x}) = y_{T_{k+1}}\left(t_L^*\right)$ si $\boldsymbol{x} \in \text{supp}\left(t_L^*\right)$ y $d_{T_{k+1}}(\boldsymbol{x}) = y_{T_{k+1}}\left(t_R^*\right)$ si $\boldsymbol{x} \in \text{supp}\left(t_R^*\right)$. Entonces, se tienen varios casos que estudiar. (i) Se asume que $\boldsymbol{x}, \boldsymbol{x}' \notin \text{supp}\left(t^*\right)$. Entonces, $d_{T_{k+1}}(\boldsymbol{x}) = d_{T_k}(\boldsymbol{x})$ y $d_{T(k+1)}(\boldsymbol{x}') = d_{T(k)}(\boldsymbol{x}')$ ya que sólo la estimación relacionada con t^* podría haber sido modificada después de la $k+1$ división. Como consecuencia, $d_{T_{k+1}}(\boldsymbol{x}) \le d_{T_{k+1}}(\boldsymbol{x}')$ bajo la suposición de que $d_{T_k}(\boldsymbol{x})$ es una función monótona no decreciente. (ii) Se asume que $\boldsymbol{x} \notin \text{supp}\left(t^*\right), \boldsymbol{x}' \in \text{supp}\left(t^*\right)$. Entonces, se puede asumir que $\boldsymbol{x} \in \text{supp}(t)$ para algunos $t \in \tilde{T}_{k+1}$. Si $\boldsymbol{x}' \in t_L^*$, por la Definición 2, $t \in I_{T_{k+1}}\left(t_L^*\right)$. Por (10), $y_{T_{k+1}}\left(t_L^*\right) = \max\left\{\max\left\{y_i : (\boldsymbol{x}_i, y_i) \in t_L^*\right\}, y\left(I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_L^*\right)\right)\right\} \ge y\left(I_{T\left(k \mid t^* \to t_L, t_R\right)}\left(t_L^*\right)\right) \ge y_{T_k}(t)..$ Ahora, $d_{T_{k+1}}(\boldsymbol{x}) \le d_{T_{k+1}}(\boldsymbol{x}')$ ya que $y_{T_k}(t) = d_{T_k}(\boldsymbol{x}) = d_{T_{k+1}}(\boldsymbol{x})$ y $d_{T_{k+1}}(\boldsymbol{x}') = y_{T_{k+1}}\left(t_L^*\right)$. Por el contrario, si $\boldsymbol{x}' \in t_R^*$, se pueden seguir pasos similares y lograr el mismo resultado. (iii) Se asume que $\boldsymbol{x} \in \text{supp}\left(t^*\right), \boldsymbol{x}' \notin \text{supp}\left(t^*\right)$. Por la hipótesis de monotonía de $d_{T_k}(\boldsymbol{x})$, se tiene $d_{T_k}(\boldsymbol{x}) \le d_{T_k}(\boldsymbol{x}')..$ Recordar que $d_{T_k}(\boldsymbol{x}') = d_{T_{k+1}}(\boldsymbol{x}')$ porque $\boldsymbol{x}' \in \left(R_+^m - \text{supp}\left(t^*\right)\right)$. Se va a probar que $d_{T_k}(\boldsymbol{x}) \le d_{T_k}(\boldsymbol{x}')$. Para hacer esto, primero se prueba que $y_{T_k}\left(t^*\right) \ge \max\left\{y_{T_{k+1}}\left(t_L^*\right), y_{T_{k+1}}\left(t_R^*\right)\right\}$. Por (11) y (12), se tiene $y_{T_{k+1}}\left(t_L^*\right) = \max\left\{\max\left\{y_i : (\boldsymbol{x}_i, y_i) \in t_L^*\right\}, y\left(I_{T\left(k \mid t^* \to t_L^*, t_R^*\right)}\left(t_L^*\right)\right)\right\}$, y $y_{T_{k+1}}\left(t_R^*\right) =$

$\max\left\{\max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_R^*\right\}, y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)\right)\right\}$. Además, $y_{T_k}\left(t^*\right)\geq \max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t^*\right\}$ por la forma en la que se define la estimación de la variable de respuesta de un nodo [expresiones (11) y (12)] y que $y_{T_0}\left(t_1\right)=\max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_1\right\}$. Además, $y_{T_k}\left(t^*\right)\geq \max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_L^*\right\}$ y $y_{T(k)}\left(t^*\right)\geq \max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_R^*\right\}$. Adicionalmente, si $t'\in I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_L^*\right)$, entonces, por la Definición 4.2, $\exists\hat{\boldsymbol{x}}\in\text{supp}\left(t_L^*\right)$ y $\exists\boldsymbol{x}''\in\text{supp}\left(t'\right)$ tal que $\boldsymbol{x}''\leq\hat{\boldsymbol{x}}$. Pero entonces, $t'\in I_{T_k}\left(t^*\right)$ ya que $\text{supp}\left(t_L^*\right)\subset\text{supp}\left(t^*\right)$. Por la hipótesis de monotonía de $d_{T(k)}\left(\boldsymbol{x}\right)$, se tiene que $d_{T_k}\left(\boldsymbol{x}''\right)\leq d_{T_k}\left(\hat{\boldsymbol{x}}\right)$. Sin embargo, notar que $d_{T_k}\left(\boldsymbol{x}''\right)=y_{T_k}\left(t'\right)$ y $d_{T_k}\left(\hat{\boldsymbol{x}}\right)=y_{T_k}\left(t^*\right)$. Así pues, $y_{T(k)}\left(t^*\right)\geq y_{T_k}\left(t'\right)$ para todo $t'\in I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_L^*\right)$. Esto implica que $y_{T_k}\left(t^*\right)\geq y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_L^*\right)\right)$. Y, por consecuencia, $y_{T_k}\left(t^*\right)\geq\max\left\{\max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_L^*\right\}, y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_L^*\right)\right)\right\}=y_{T_{k+1}}\left(t_L^*\right)$. Por analogía, se pueden seguir pasos similares para el nodo hijo derecho t_R^* y tener que $y_{T_k}\left(t^*\right)\geq\max\left\{\max\left\{y_i:(\boldsymbol{x}_i,y_i)\in t_R^*\right\}, y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)\right)\right\}=y_{\tilde{T}_{k+1}}\left(t_R^*\right)$. Así, $y_{T_k}\left(t^*\right)\geq\max\left\{y_{T_{k+1}}\left(t_L^*\right), y_{T_{k+1}}\left(t_R^*\right)\right\}$. De esta forma, se tiene que $d_{T_{k+1}}\left(\boldsymbol{x}\right)=y_{T_{k+1}}\left(t_L^*\right)$ si $\boldsymbol{x}\in t_L^*$ y $d_{T_{k+1}}\left(\boldsymbol{x}\right)=y_{T_{k+1}}\left(t_R^*\right)$ si $\boldsymbol{x}\in t_R^*$ e, incluso, $d_{T(k)}\left(\boldsymbol{x}\right)=y_{T(k)}\left(t^*\right)$. Por consiguiente, se consigue que $d_{T_{k+1}}\left(\boldsymbol{x}\right)\leq d_{T_k}\left(\boldsymbol{x}\right)$. Recordad que se tenía $d_{T_k}\left(\boldsymbol{x}\right)\leq d_{T_k}\left(\boldsymbol{x}'\right)=d_{T_{k+1}}\left(\boldsymbol{x}'\right)$. Así pues, se consigue lo que se quería probar: $d_{T_{k+1}}\left(\boldsymbol{x}\right)\leq d_{T_{k+1}}\left(\boldsymbol{x}'\right)$. (iv) Se asume que $\boldsymbol{x},\boldsymbol{x}'\in\text{supp}\left(t^*\right)$. Si $\boldsymbol{x},\boldsymbol{x}'\in\text{supp}\left(t_L^*\right)$ o $\boldsymbol{x},\boldsymbol{x}'\in\text{supp}\left(t_R^*\right)$, entonces $d_{T_{k+1}}\left(\boldsymbol{x}\right)=d_{T_{k+1}}\left(\boldsymbol{x}'\right)$ y se obtiene el resultado que se quería. Por ello, se supone que $\boldsymbol{x}\in\text{supp}\left(t_L^*\right)$ y $\boldsymbol{x}'\in\text{supp}\left(t_R^*\right)$. El otro caso no es posible a través de (12) $\boldsymbol{a}^{t_L^*}=\boldsymbol{a}^{t^*}$ y $\boldsymbol{b}^{t^*}=\boldsymbol{b}^{t_R^*}$, por la hipótesis de $\boldsymbol{a}^{t^*}<\boldsymbol{b}^{t^*}$ [por el contrario, $\text{supp}\left(t^*\right)=\varnothing$] y, por la aplicación de la Proposición 4.2, tenemos que $t_L^*\in I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)$ y $t_R^*\notin I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_L^*\right)$. Ahora, por (9),

$$y_{T_{k+1}}\left(t_R^*\right)=\max\left\{\max\left\{y_i:\left(\boldsymbol{x}_i,y_i\right)\in t_R^*\right\},y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)\right)\right\}\geq y\left(I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)\right)\geq y_{T_{k+1}}\left(t_L^*\right)$$

ya que $t_L^*\in I_{T\left(k|t^*\to t_L^*,t_R^*\right)}\left(t_R^*\right)$. Entonces, $d_{T_{k+1}}\left(\boldsymbol{x}\right)\leq d_{T_{k+1}}\left(\boldsymbol{x}'\right)$. Finalmente, por inducción, se obtiene el resultado deseado ya que $d_{T_0}\left(\boldsymbol{x}\right)$ es trivialmente una función monótona no decreciente. ■

En los procesos de producción de un solo output, la libre disponibilidad se traduce en la monotonía del predictor. De esta manera, el Teorema 4.1 indica que la tecnología inducida $\hat{\Psi}_{T_{\max}}:=\left\{\left(\boldsymbol{x},y\right)\in R_+^{m+1}:y\leq d_{T_{\max}}\left(\boldsymbol{x}\right)\right\}$, definida por el árbol profundo $T_{\max}$, satisface la conocida propiedad en la microeconomía de la libre disponibilidad. Siguiendo un razonamiento similar, y debido al proceso de actualización de la estimación de la variable de salida, cada subárbol T_k está también asociado con un predictor monótono decreciente $d_{T_k}\left(\boldsymbol{x}\right)$ y una tecnología $\hat{\Psi}_{T_k}:=\left\{\left(\boldsymbol{x},y\right)\in R_+^{m+1}:y\leq d_{T_k}\left(\boldsymbol{x}\right)\right\}$ que satisface la libre disponibilidad. Además, EAT genera predictores que son funciones escalonadas en R_+^m como la técnica conocida como FDH en el análisis de fronteras. Por consiguiente, parece apropiado comparar ambos enfoques.

Aunque EAT y FDH no producen los mismos predictores en general, la siguiente proposición muestra que ambas técnicas construyen la misma función escalonada como estimador del output cuando el número de variables x_j es uno, es decir, $m=1$, y la regla de para es $n_{\min}=1$.

Proposición 4.4. Dado $m=1$ y $n_{\min}=1$. Entonces, $d_{T_{\max}}\left(x\right)=\hat{f}_{FDH}\left(x\right)$ para todo $x\in R_+$.

Demostración. Se asume, sin pérdida de generalidad, que la muestra de aprendizaje puede ser clasificada en la forma $\left(x_1,y_1\right),\left(x_2,y_2\right),...,\left(x_n,y_n\right)$, con $x_1<x_2<...<x_n$. Después de

aplicar el algoritmo EAT, se obtendría que cada ejemplo de aprendizaje (x_i, y_i) pertenecería a un nodo terminal $t_i \in T_{\max}$ con un soporte de la forma $\text{supp}(t_i) = [x_i, x_{i+1})$, $i = 1,...,n$ $[x_{n+1} := +\infty]$. Dado $x_{i^*} = \min_{i: y_i = \max_{1 \le k \le n}\{y_k\}} \{x_i\}$. Entonces, $\hat{f}_{FDH}(x) = y_{i^*}$ para todo $x \ge x_{i^*}$. En cuanto a la técnica EAT, por construcción, $d_{T_{\max}}(x) \le \max_{1 \le k \le n}\{y_k\} = y_{i^*}$ para cualquier $x \in R_+$. Incluso, $d_{T_{\max}}(x) \ge y_{i^*}$ para todo $x \ge x_{i^*}$. Esta última desigualdad se debe al Teorema 1 y que $d_{T_{\max}}(x_{i^*}) \ge y_{i^*}$. Por lo tanto, $d_{T_{\max}}(x) = y_{i^*}$ para todo $x \ge x_{i^*}$ y $d_{T_{\max}}(x) = \hat{f}_{FDH}(x)$ para todo $x \ge x_{i^*}$. Obsérvese que tanto la estimación generada a partir de FDH como la estimación producida por EAT para un punto $x < x_{i^*}$ no dependen de la estimación de la variable respuesta para los valores $x \ge x_{i^*}$. En el caso de FDH, este hecho es evidente como consecuencia de su definición $\hat{f}_{FDH}(x) = \max_{i: x \ge x_i}\{y_i\}$. En el caso de EAT, durante la aplicación del algoritmo correspondiente, cuando la estimación de la variable respuesta para cualquier nodo t_i, $i = 1,...,i^* - 1$, debe determinarse, sólo los nodos con un soporte contenido en el segmento $[x_1, x_i)$ son los nodos Pareto dominantes de t_i y, por lo tanto, son los únicos nodos que pueden influir en la estimación final para t_i. Ahora, se puede repetir el proceso, pero centrando la atención en la submuestra de aprendizaje $(x_1, y_1), (x_2, y_2), ..., (x_{i^*-1}, y_{i^*-1})$. Si se repite tantas veces como sea necesario, se obtiene que $d_{T_{\max}}(x) = \hat{f}_{FDH}(x)$ para todo $x \in R_+^m$. ■

Por lo tanto, la Figura 2, que ilustra la forma típica del predictor de salida determinado por la aplicación de la técnica de FDH, también es válida para la técnica EAT suponiendo que $n_{\min} = 1$. Sin embargo, la Proposición 4.4 no se sostiene en general. De hecho, se muestra en la sección correspondiente a la experiencia computacional que las discrepancias entre FDH y EAT aumentan a medida que el tamaño de la muestra y el número de variables predictoras x_j aumentan.

A continuación, se muestra un ejemplo sencillo con $m = 2$ (es decir, dos inputs) con el fin de ilustrar algunas diferencias específicas entre las dos técnicas en cuanto a los resultados obtenidos. Se supone que la muestra de aprendizaje consiste en los siguientes ejemplos (x_{1i}, x_{2i}, y_i): A = (1, 4, 2), B = (2, 2, 1), C = (3, 1, 3), D = (2, 6, 5), E = (3, 4, 1), F = (4, 3, 2), G = (5, 1, 1), H = (4, 8, 10), I = (5, 5, 6), J = (6, 4, 4) y K = (7, 2, 7). Entonces, si se aplica la técnica de FDH, se obtiene el siguiente predictor de la función de producción (ver Figura 8). En esta Figura 8, se muestran las regiones determinadas por FDH en el espacio de input junto con los datos y el valor de la estimación del output.

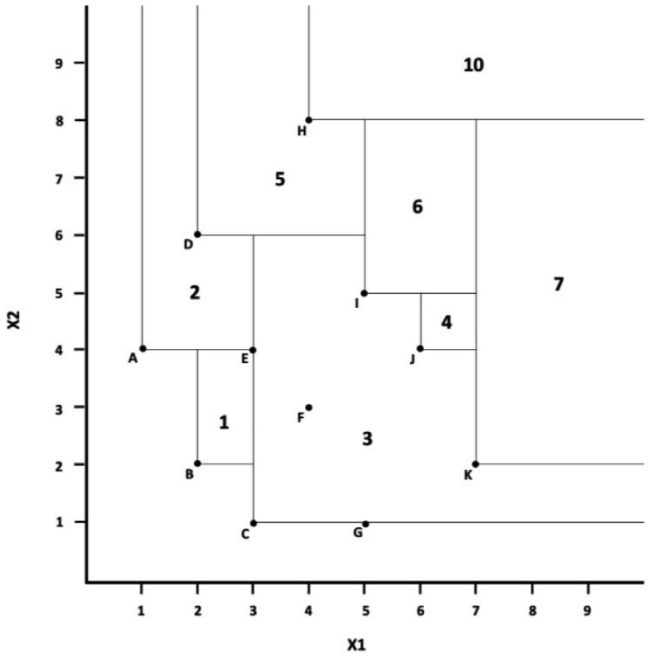


Figura 8. Ejemplo de FDH para dos inputs.

En cuanto a la técnica EAT, jugando con los mismos ejemplos, genera con $n_{\min} = 1$ el predictor que se muestra en la Figura 9. Obsérvese que cada nodo terminal del árbol está asociado a un soporte, es decir, un rectángulo. Obsérvese también que cada nodo tiene al menos una observación. Además, es fácil comprobar que ambas técnicas no devuelven la

misma estimación de la variable respuesta para todos los $(x_1, x_2) \in R_+^m$. Por ejemplo, los puntos de la esquina noroeste de la Figura 9 presentan una estimación de 2 en el caso de FDH y 10 en el caso de EAT.

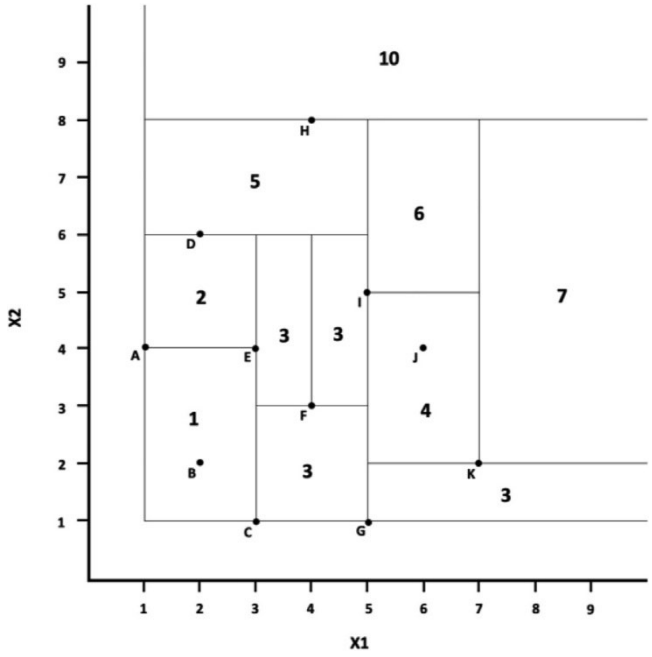


Figura 9. Ejemplo de EAT con dos inputs ($n_{\min} = 1$).

Además, aunque la técnica de FDH genera una función escalonada como predictor, como EAT, se sabe que no puede ser interpretada como una estructura basada en nodos y soportes en el espacio de inputs. Sin embargo, para reinterpretar los resultados asociados a la técnica FDH de esa manera, se podría obtener la estructura de soportes que se muestra en la Figura 10. En esta estructura, el nodo relacionado con el soporte que aparece en la esquina noroeste estaría vacío, es decir, ninguna observación pertenece a cada soporte. La técnica es capaz de determinar una estimación del output para el mismo, debido a los supuestos de la metodología FDH. Sin embargo, los enfoques habituales basados en datos necesitan datos para determinar una estimación en esa región del espacio de inputs. EAT, al igual que CART, no es capaz de determinar los nodos terminales sin datos, ya que para ejecutar una división ambos requieren calcular el error cuadrático medio en cada nodo

hijo, algo que sólo es posible si tenemos al menos un ejemplo en cada soporte correspondiente.

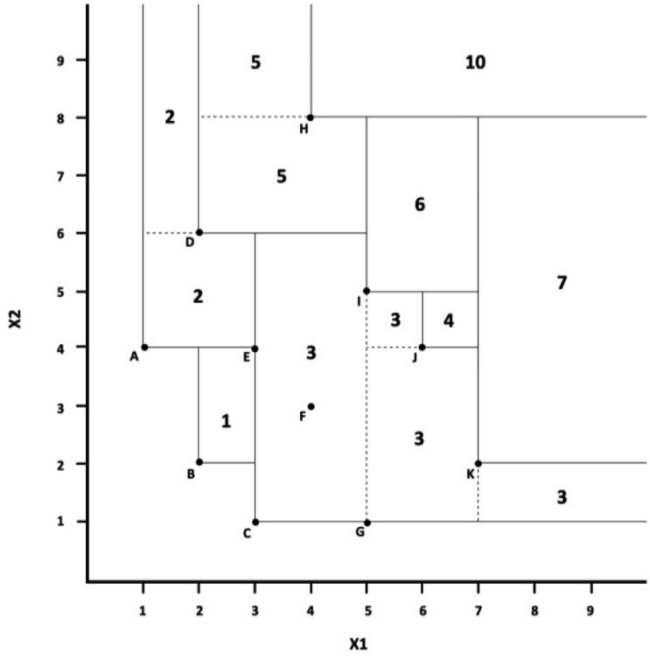


Figura 10. Ejemplo de FDH como un árbol de regresión.

Desafortunadamente, como se menciona anteriormente, FDH muestra un notable sobreajuste en los datos. De hecho, FDH produce estimaciones de eficiencia que son demasiado optimistas. Lo mismo ocurre con la técnica EAT cuando se permite que crezca un árbol profundo. Así que, en este punto, uno podría pensar que la introducción de este nuevo enfoque no tiene sentido, salvo por el hecho de que nos ha permitido crear un puente entre dos literaturas que han crecido en paralelo: el análisis de fronteras y el aprendizaje automático. Sin embargo, esto no es cierto. El nuevo enfoque, que se basa en CART, puede explotar ciertas técnicas normalmente vinculadas a CART, como la poda y la validación cruzada para superar el sobreajuste. En adelante, se denomina al árbol resultante del proceso de poda como T^*.

En general, se sugiere que siempre se aplique EAT junto con la poda y la validación cruzada para determinar una estimación adecuada de las funciones de producción. Para

ello, el proceso estándar llevado a cabo por CART para la poda del árbol profundo debería adaptarse ligeramente al contexto del análisis de fronteras. En el Capítulo 3 Sección 3.2., se menciona cómo podar un árbol profundo en CART a través de la validación cruzada. En el caso de EAT, el proceso es idéntico, excepto por la forma en que crecen los árboles, que se basa en el algoritmo introducido en esta sección, y por el orden de los subárboles seguido en el proceso de poda. En este nuevo contexto, la clave para la modificación del enfoque estándar será fijar la forma en la que se determina la secuencia de subárboles. Dado que el algoritmo EAT genera una secuencia de árboles de tamaño creciente en la forma $\{t_0\} \prec T_1 \prec T_2 \prec \ldots \prec T_K = T_{\max}$, y se garantiza que cada árbol T_k satisface la libre disponibilidad, se usa la secuencia inversa, es decir, $T_{\max} \succ T_{K-1} \succ T_{K-2} \succ \ldots \succ \{t_0\}$ como una secuencia de subárboles para el proceso de poda.

En la Figura 11, se muestra el predictor generado por EAT después de la poda, basado en el mismo ejemplo desarrollado en la Figura 2.

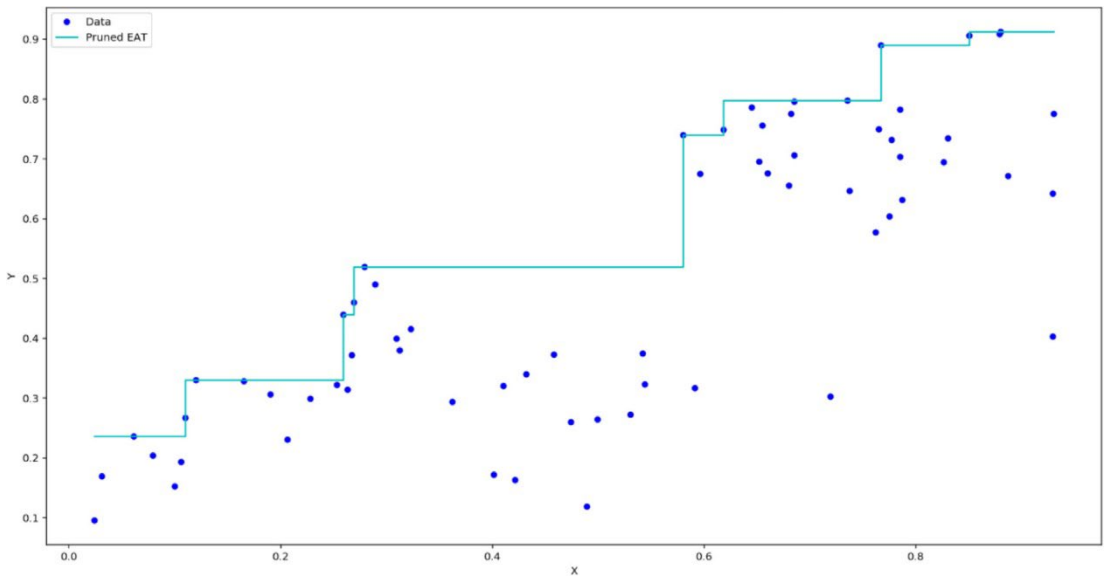


Figura 11. Ejemplo de las estimaciones EAT después del proceso de la poda.

Además, se muestra un ejemplo en tres dimensiones en las Figuras 12 y Figura 13. En la Figura 12 se muestra el predictor generado por EAT sin poda, mientras que en la Figura

13 se ilustra el predictor EAT de la función de producción después de realizar el proceso de poda.

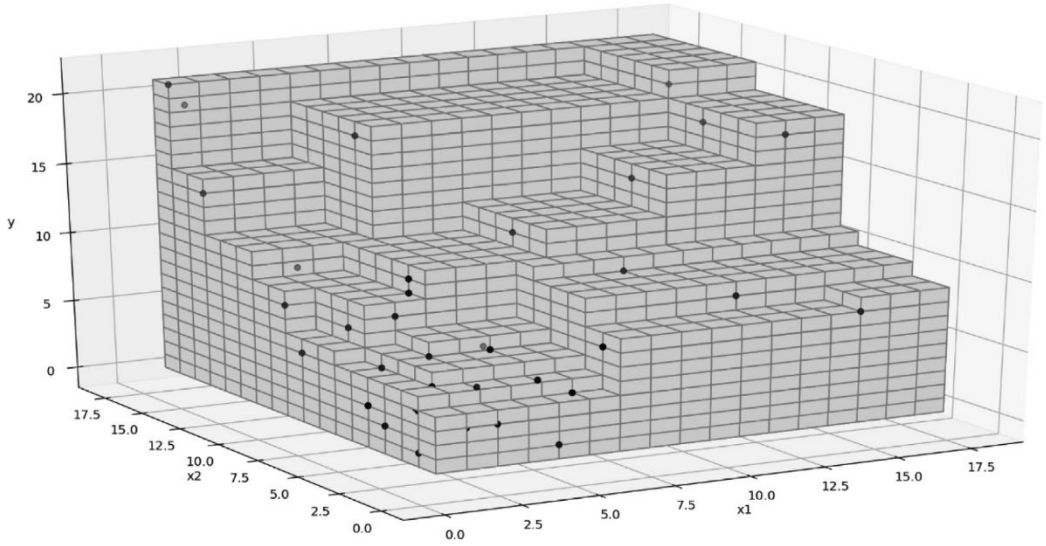


Figura 12. Ejemplo de las estimaciones EAT en tres dimensiones.

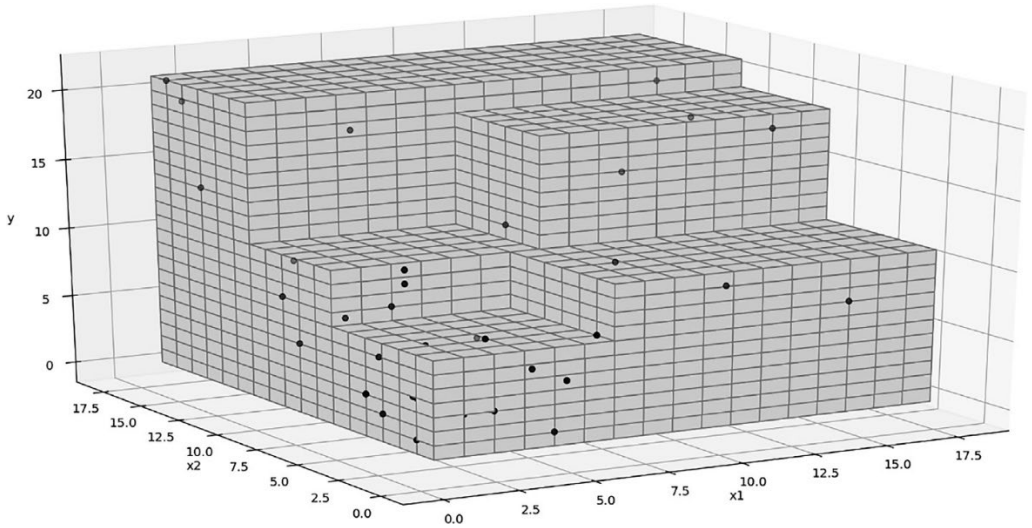


Figura 13. Ejemplo de las estimaciones EAT en tres dimensiones después del proceso de la poda.

La discusión anterior permite interpretar la técnica EAT como una forma de estimar las funciones de producción basada en los fundamentos del aprendizaje automático,

determinando una función escalonada monótona no decreciente como predictor. En cierto sentido, el nuevo enfoque es similar a la técnica estándar de FDH. Sin embargo, mientras que FDH sufre un problema de sobreajuste, la técnica EAT lo evita mediante la poda y la validación cruzada (Breiman et al., 1984), resolviendo los problemas iniciales.

Seguidamente, se establece la relación que existe entre la estimación de la producción de FDH, EAT sin poda y EAT podado. La función de producción estimada mediante el procedimiento de poda siempre es mayor o igual a la estimación obtenida al aplicar EAT sin poda y ésta es, a su vez, mayor o igual a la función de producción generada mediante la técnica FDH.

Proposición 4.5. $d_{T^*}(\boldsymbol{x}) \geq d_{T_{\max}}(\boldsymbol{x}) \geq \hat{f}_{FDH}(\boldsymbol{x})$.

Demostración. Este resultado es inmediato. ■

4.1.2. EL CASO DE MÚLTIPLES VARIABLES RESPUESTA

En esta sección se muestra cómo extender el algoritmo EAT univariado al contexto de múltiples outputs mediante la aplicación del enfoque de De'ath (2002). Los cambios son los mismos que en CART. En particular, e inspirados por el algoritmo de output único de EAT, se sugiere estimar el valor de la variable de respuesta y_r, $r = 1,...,s$, a través de la fórmula $y_r(t_L) = \max\left\{\max\left\{y_{ri} : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in t_L\right\}, y_r\left(I_{T(k|t^* \to t_L, t_R)}(t_L)\right)\right\}$ para el nodo hijo izquierdo; y la fórmula $y_r(t_R) = \max\left\{\max\left\{y_{ri} : (\boldsymbol{x}_i, \boldsymbol{y}_i) \in t_R\right\}, y_r\left(I_{T(k|t^* \to t_L, t_R)}(t_R)\right)\right\}$ para el nodo hijo derecho; donde $y_r\left(I_{T(k|t^* \to t_L, t_R)}(t_L)\right) = \max\left\{y_r(t') : t' \in I_{T(k|t^* \to t_L, t_R)}(t_L)\right\}$, $y_r\left(I_{T(k|t^* \to t_L, t_R)}(t_R)\right) = \max\left\{y_r(t') : t' \in I_{T(k|t^* \to t_L, t_R)}(t_R)\right\}$ y $y_r(t')$ es la estimación de la

variable respuesta y_r para el nodo $t' \in \tilde{T}\left(k \mid t^* \rightarrow t_L, t_R\right)$, $r = 1, ..., s$. Además, el algoritmo multivariante EAT elige la combinación $\left(x_j, s_j\right)$ tal que $R\left(t_L\right) + R\left(t_R\right) = \frac{1}{n} \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in t_L} \sum_{r=1}^{s} \left(y_{ri} - y_r\left(t_L\right)\right)^2 + \frac{1}{n} \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in t_R} \sum_{r=1}^{s} \left(y_{ri} - y_r\left(t_R\right)\right)^2$, es decir, el objetivo es minimizar la suma del error cuadrático medio (MSE) sobre todas las variables respuesta. Finalmente, como en el caso del output único, el proceso de división se repite hasta que se encuentra una división k', tal que para todo $t \in \tilde{T}\left(k'\right)$ la regla de parada $n\left(t\right) \leq 5$ es alcanzada. Entonces, k' será K y el árbol construido será el árbol profundo $T_{\max} = T\left(K\right)$. También, dado $\boldsymbol{d}_{T_{\max}}\left(\boldsymbol{x}\right) = \boldsymbol{y}_{T_{\max}}\left(t\right)$ para $t \in \tilde{T}_{\max}$ tal que $\boldsymbol{x} \in \text{supp}\left(t\right)$, es decir, $\boldsymbol{d}_{T_{\max}}\left(\boldsymbol{x}\right)$ es el predictor multidimensional definido a partir del árbol $T_{\max}$. A partir de este predictor, es posible definir la tecnología o el conjunto de posibilidades de producción inducido por $\boldsymbol{d}_{T_{\max}}\left(\boldsymbol{x}\right)$ como:

$$\hat{\Psi}_{T_{\max}} = \left\{\left(\boldsymbol{x}, \boldsymbol{y}\right) \in R_+^{m+s} : \boldsymbol{y} \leq \boldsymbol{d}_{T_{\max}}\left(\boldsymbol{x}\right)\right\} \tag{14}$$

Sin embargo, aunque la ampliación propuesta del algoritmo EAT de output único parece válida a priori porque corresponde a una aplicación directa del enfoque CART de respuesta múltiple de De'ath (2002), el método EAT de output múltiple no sería adecuado para abordar la estimación de las fronteras de producción a menos que las estimaciones de las tecnologías subyacentes que este nuevo algoritmo produce satisfagan el axioma de la libre disponibilidad. La proposición siguiente establece dicho resultado.

Proposición 4.6. El conjunto $\hat{\Psi}_{T_{\max}}$ cumple la propiedad de libre disponibilidad en inputs y outputs.

Demostración. Sea $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\Psi}_{T_{\max}}$ y $(\boldsymbol{x}',\boldsymbol{y}') \in R_{+}^{m+s}$ tal que $\boldsymbol{x}' \geq \boldsymbol{x}$ e $\boldsymbol{y}' \leq \boldsymbol{y}$. Se quiere probar que $(\boldsymbol{x}',\boldsymbol{y}') \in \hat{\Psi}_{T_{\max}}$. Por un lado, se tiene que $\boldsymbol{y} \leq \boldsymbol{d}_{T_{\max}}(\boldsymbol{x})$ porque $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\Psi}_{T_{\max}}$. Por otra parte, por la forma en que se define la actualización de la estimación de cada componente del vector de salida $\boldsymbol{y}(t)$ para cada nodo del algoritmo EAT de output múltiple, que sigue la definición del enfoque EAT de output único, se tiene que si $\boldsymbol{x}' \geq \boldsymbol{x}$, entonces $\boldsymbol{d}_{T_{\max}}(\boldsymbol{x}') \geq \boldsymbol{d}_{T_{\max}}(\boldsymbol{x})$ por el Teorema 4.1. Además, $\boldsymbol{y}' \leq \boldsymbol{y} \leq \boldsymbol{d}_{T_{\max}}(\boldsymbol{x}) \leq \boldsymbol{d}_{T_{\max}}(\boldsymbol{x}')$, lo que implica, por (14), que $(\boldsymbol{x}',\boldsymbol{y}') \in \hat{\Psi}_{T_{\max}}$. ■

Como se ha demostrado, el algoritmo EAT de output múltiple es capaz de producir estimaciones de una tecnología que satisface los axiomas clásicos de la microeconomía. En particular EAT, satisface la libre disponibilidad, como FDH. Además, el algoritmo EAT genera funciones de producción escalonadas; tal y como sucede con FDH. Sin embargo, FDH también se basa en el postulado de la mínima extrapolación, que establece que la estimación más conservadora de la frontera de producción sería la asociada a una superficie que envuelva los datos y esté lo más cerca posible de ellos. Se recuerda que la satisfacción de este enfoque está ligada al problema del sobredimensionamiento o sobreajuste mencionado anteriormente en la tesis. A priori, el algoritmo EAT no se basa en el axioma de la mínima extrapolación. No obstante, y dado que el árbol que se ha generado en este momento en esta sección por el enfoque EAT es el árbol más profundo, en cierto sentido, este árbol como predictor también sufre el problema del sobreajuste (véase Breiman et al., 1984). Es bien conocida la forma de corregir el problema del sobreajuste con los árboles de decisión. Una solución estándar consiste en podar el árbol profundo recurriendo a un proceso de validación cruzada. Esta es una de las ventajas de haber relacionado la estimación de las fronteras de producción con CART. En particular, para el caso de múltiples outputs, aplicamos los mismos pasos previamente definidos para el algoritmo EAT de output único.

Después del proceso de poda, obtenemos un subárbol final T^* que permite definir una estimación de la tecnología subyacente como $\hat{\Psi}_{T^*} := \left\{(\boldsymbol{x},\boldsymbol{y}) \in R_+^{m+s} : \boldsymbol{y} \le \boldsymbol{d}_{T^*}(\boldsymbol{x})\right\}$, donde $\boldsymbol{d}_{T^*}(\boldsymbol{x})$ es el predictor asociado con T^*. Como con $\hat{\Psi}_{T_{\max}}$, es posible probar que $\hat{\Psi}_{T^*}$ satisface el axioma de libre disponibilidad. Sin embargo, T^* no es tan profundo como $T_{\max}$. Las semejanzas entre FDH estándar y EAT multi-output pueden resumirse en el siguiente resultado, que establece que el FDH generado a partir de una cierta muestra "virtual" de aprendizaje coincide con la estimación de la tecnología subyacente producida por el algoritmo EAT multi-output. Los ejemplos que forman parte de esta muestra especial de aprendizaje son los siguientes puntos en el espacio de input-output: $\left(\boldsymbol{a}^t, \boldsymbol{d}_{T^*}(\boldsymbol{a}^t)\right)$, para todo $t \in \tilde{T}^*$. En otras palabras, son unidades "virtuales", es decir, no se observan necesariamente en la muestra original de aprendizaje, que consumen el vector de entrada $\boldsymbol{a}^t$ perteneciente al soporte del nodo t, y produce el output correspondiente a la estimación EAT del conjunto de outputs para el vector de entrada $\boldsymbol{a}^t$.

Proposición 4.7. La tecnología FDH construida a partir del conjunto de puntos $\left\{\left(\boldsymbol{a}^t, \boldsymbol{d}_{T^*}(\boldsymbol{a}^t)\right)\right\}_{t\in\tilde{T}^*}$ coincide con $\hat{\Psi}_{T^*}$:

$$\hat{\Psi}_{T^*} = \left\{(\boldsymbol{x},\boldsymbol{y}) \in R_+^{m+s} : \exists t \in \tilde{T}^* \text{ tal que } \boldsymbol{y} \le \boldsymbol{d}_{T^*}(\boldsymbol{a}^t), \boldsymbol{x} \ge \boldsymbol{a}^t\right\} \tag{15}$$

Demostración. Se denota el lado derecho de la expresión (15) como Ψ_{FDH}^{EAT}. Sea $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\Psi}_{T^*}$. Entonces, por definición, $\boldsymbol{y} \le \boldsymbol{d}_{T^*}(\boldsymbol{x})$. Sea $t_x \in \tilde{T}^*$ tal que $\boldsymbol{x} \in \text{supp}(t_x)$. Entonces, $\boldsymbol{d}_{T^*}(\boldsymbol{x}) = \boldsymbol{y}_{T^*}(t_x)$. Notar que $\boldsymbol{a}^{t_x} \in \text{supp}(t_x)$, lo que implica que $\boldsymbol{d}_{T^*}(\boldsymbol{a}^{t_x}) = \boldsymbol{y}_{T^*}(t_x)$. Además, se tiene que $\boldsymbol{a}^{t_x} \le \boldsymbol{x}$ porque $\boldsymbol{x} \in \text{supp}(t_x)$. En este sentido, se tiene que $\boldsymbol{d}_{T^*}(\boldsymbol{x}) = \boldsymbol{y}_{T^*}(t_x) = \boldsymbol{d}_{T^*}(\boldsymbol{a}^{t_x})$ y $\boldsymbol{x} \ge \boldsymbol{a}^{t_x}$. Como consecuencia, $\exists t \in \tilde{T}^*$ tal que $\boldsymbol{y} \le \boldsymbol{d}_{T^*}(\boldsymbol{a}^t)$ y $\boldsymbol{x} \ge \boldsymbol{a}^t$ y, también, $(\boldsymbol{x},\boldsymbol{y}) \in \Psi_{FDH}^{EAT}$. Sea ahora $(\boldsymbol{x},\boldsymbol{y}) \in \Psi_{FDH}^{EAT}$.

Entonces, $\exists t \in \tilde{T}^*$ tal que $\boldsymbol{y} \le \boldsymbol{d}_{T^*}\left(\boldsymbol{a}^t\right)$ y $\boldsymbol{x} \ge \boldsymbol{a}^t$. Debido a la forma de denotar la actualización del estimador de cada componente del vector de salida $\boldsymbol{y}(t)$ para cada nodo bajo el algoritmo EAT de output múltiple, se tiene que si $\boldsymbol{x}' \ge \boldsymbol{x}$, entonces $\boldsymbol{d}_{T^*}\left(\boldsymbol{x}'\right) \ge \boldsymbol{d}_{T^*}\left(\boldsymbol{x}\right)$. Por tanto, $\boldsymbol{d}_{T^*}\left(\boldsymbol{x}\right) \ge \boldsymbol{d}_{T^*}\left(\boldsymbol{a}^t\right)$. Entonces, tenemos $\boldsymbol{y} \le \boldsymbol{d}_{T^*}\left(\boldsymbol{a}^t\right) \le \boldsymbol{d}_{T^*}\left(\boldsymbol{x}\right)$, que implica, por la definición de $\hat{\Psi}_{T^*}$, que $\left(\boldsymbol{x}, \boldsymbol{y}\right) \in \hat{\Psi}_{T^*}$. ■

La Proposición 4.7 tiene implicaciones muy importantes. Este resultado muestra el modo de calcular cualquier medida de eficiencia técnica utilizando la estimación $\hat{\Psi}_{T^*}$ como caso base. En el caso de la medida radial orientada a la producción, la puntuación de eficiencia $\phi\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right)$ puede ser estimada "enchufando" $\hat{\Psi}_{T^*}$ en (2) en lugar de Ψ. En nuestro caso, y gracias a la relación entre FDH y la técnica de EAT multi-output establecida en la Proposición 4.7, por analogía con (4), el programa lineal de números enteros mixtos que se debe resolver es el siguiente:

$$
\begin{aligned}
\phi^{EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right) = \quad & \max \quad \phi_k \\
& s.t. \\
& \sum_{t \in \tilde{T}^*} \lambda_{ki} \boldsymbol{a}^t_j \le x_{kj}, && j = 1, \ldots, m \\
& \sum_{t \in \tilde{T}^*} \lambda_{ki} \boldsymbol{d}_{T^*_r}\left(\boldsymbol{a}^t\right) \ge \phi_k y_{kr}, && r = 1, \ldots, s \\
& \sum_{t \in \tilde{T}^*} \lambda_{ki} = 1, \\
& \lambda_{ki} \in \{0,1\}, && i = 1, \ldots, n
\end{aligned}
\tag{16}
$$

En la siguiente sección, mostramos cómo funciona el nuevo enfoque en comparación con FDH en varias experiencias computacionales.

4.1.3. EXPERIENCIA COMPUTACIONAL

En esta sección se describen los resultados de la simulación que sirven para comparar los métodos: FDH vs. EAT (podado). Para ello, se presenta una comparación sistemática de estos dos métodos de estimación de frontera en cuatro escenarios simulados alternativos para el caso de un solo output, y en otro escenario para el caso de múltiples outputs. La descripción de los cinco escenarios aparece en la Tabla 1.

Tabla 1. Escenarios a simular.

Escena -rios	#Inputs| #Outputs	Forma funcional
(1)	1|1	$y = \ln(x) + 3$
(2)	1|2	$y = 0.1x_1 + 0.1x_2 + 0.3(x_1x_2)^{1/2}$
(3)	1|3	$y = 0.1x_1 + 0.1x_2 + 0.1x_3 + 0.3(x_1x_2x_3)^{1/3}$
(4)	1|9	$y = \prod_{j=1}^{9} x_j^{\frac{1}{10}}$
(5)	2|2	$-\ln y_1 = -1 + 0.5 \cdot \left(\frac{\ln y_2}{\ln y_1}\right) + 0.25 \cdot \left(\frac{\ln y_2}{\ln y_1}\right)^2 - 1.5 \cdot (\ln x_1) - 0.6 \cdot (\ln x_2) + 0.2 \cdot (\ln x_1)^2 + 0.05 \cdot (\ln x_2)^2 - 0.1 \cdot (\ln x_1) \cdot (\ln x_2) + 0.05 \cdot (\ln x_1) \cdot \left(\frac{\ln y_2}{\ln y_1}\right) - 0.05 \cdot (\ln x_2) \cdot \left(\frac{\ln y_2}{\ln y_1}\right)$

El escenario 1 representa un caso de un único input con una función de producción muy conocida que utiliza la función logarítmica. El escenario 2 implica dos inputs, y el escenario 3 es la experiencia computacional que utiliza tres inputs. Para todos los escenarios, se prueban tres tamaños de conjuntos de datos de 50, 100 y 150 observaciones. Los datos de entrada fueron muestreados al azar con una $Uni[1,10]$ independiente de cada input y observación. Luego, se calculó el nivel de rendimiento eficiente y un término de ineficiencia aleatorio $u \sim |N(0,0.4)|$ que se restó para obtener los datos utilizados para el análisis. Se hicieron 100 experimentos $l = 1,...,100$ para cada combinación de escenario y tamaño del conjunto de datos para investigar el rendimiento relativo de los métodos. Estos tres escenarios fueron tomados de Kuosmanen y Johnson (2010). En cuanto al cuarto escenario, se tomó de Lee y Cai (2020), y se consideraron un output y nueve inputs.

Además, se simuló una situación de producción multi-input multi-output en el escenario 5. Se siguieron las ideas propuestas por Perelman y Santin (2009). La situación de múltiples outputs es más difícil de simular. En este caso, la dificultad está asociada con la forma de generar datos adecuados para los procesos de producción multi-output que satisfagan las típicas condiciones de regularidad en teoría de la producción. Al igual que en Perelman y Santin (2009), se consideró una distribución seminormal (*halfnormal*, en inglés) para generar el término de ineficiencia y también se incorporaron dos perturbaciones estadísticas aleatorias independientes (ruido aleatorio), lo que permite que las perturbaciones aleatorias afecten a los outputs en diferente cantidad y dirección. Además, se permitió que el 0%, 10% y 25% de las DMUs simuladas estuvieran en la frontera teórica. Se lanzaron 100 experimentos $l = 1,...,100$ para cada combinación de tamaño de muestra y porcentaje de unidades en la frontera. El rendimiento de cada método se evalúa mediante dos criterios estándar: el error cuadrático medio (MSE) y el sesgo (bias, en inglés). El MSE se define como $\sum_{l=1}^{100}\sum_{i=1}^{n}\left(d_{T_l^*}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right)^2 \Big/ 100n$ para el método EAT y como $\sum_{l=1}^{100}\sum_{i=1}^{n}\left(f_{FDH_l}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right)^2 \Big/ 100n$ para el método FDH, donde l está asociada a los datos simulados y los predictores de EAT y FDH en el experimento l, $l = 1,...,100$. Como de costumbre, el MSE mide la precisión de las estimaciones en términos cuadráticos. El sesgo, sin embargo, se calcula de dos maneras diferentes. La primera como $\sum_{l=1}^{100}\sum_{i=1}^{n}\left(d_{T_l^*}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right) \Big/ 100n$ para el enfoque de EAT y como $\sum_{l=1}^{100}\sum_{i=1}^{n}\left(\hat{f}_{FDH_l}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right) \Big/ 100n$ para el de FDH. En este caso, el signo del sesgo indica si la frontera estimada subestima sistemáticamente (signo negativo) o sobreestima (signo positivo) la frontera teórica $f(\boldsymbol{x})$. Sin embargo, como las desviaciones positivas y negativas se anulan cuando promediamos todas las observaciones y experimentos, también calculamos un segundo sesgo basado en el valor absoluto de las desviaciones: $\sum_{l=1}^{100}\sum_{i=1}^{n}\left|d_{T_l^*}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right| \Big/ 100n$ para EAT y $\sum_{l=1}^{100}\sum_{i=1}^{n}\left|\hat{f}_{FDH_l}\left(\boldsymbol{x}_i^l\right)-f\left(\boldsymbol{x}_i^l\right)\right| \Big/ 100n$ para FDH (ver Kuosmanen and Johnson, 2010).

En la Tabla 2 se presentan las estadísticas de MSE y de sesgo de los dos enfoques evaluados en los escenarios de un solo output y la medida de la discrepancia entre EAT (profundo) y FDH. Las dos primeras columnas indican el escenario y el tamaño de la muestra. Las dos columnas siguientes indican el MSE de las técnicas EAT y FDH. Las dos columnas siguientes muestran el sesgo, mientras que las dos siguientes están relacionadas con el sesgo basado en el valor absoluto. Por último, la última columna de la Tabla 2 indica la media del porcentaje de observaciones sobre todos los experimentos en los que la medida de la discrepancia entre EAT profundo y FDH es mayor o igual al 10%. Además, también se ha calculado (y mostrado entre paréntesis) la diferencia relativa entre EAT podado y FDH con respecto al MSE y el sesgo para facilitar la comparación de los resultados. Estos valores podrían verse como los porcentajes de reducción del MSE y el sesgo cuando se aplica EAT podado en lugar de FDH.

Tabla 2. Resultados de los escenarios (1)-(4) para FDH vs EAT.

Escenario	Núm. Obs.	Error cuadrático medio		Sesgo		Sesgo absoluto		Discrepancias entre FDH y EAT profundo
		FDH	EAT podado	FDH	EAT podado	FDH	EAT podado	
(1)	50	0.032	0.028 (13%)	-0.145	-0.115 (21%)	0.145	0.130 (10%)	0%
(1)	100	0.019	0.015 (18%)	-0.109	-0.079 (28%)	0.109	0.093 (15%)	0%
(1)	150	0.013	0.010 (19%)	-0.089	-0.057 (36%)	0.089	0.076 (15%)	0%
(2)	50	0.121	0.102 (16%)	-0.281	-0.241 (14%)	0.281	0.253 (10%)	2.23%
(2)	100	0.104	0.086 (18%)	-0.268	-0.235 (12%)	0.268	0.240 (10%)	4.27%
(2)	150	0.091	0.073 (19%)	-0.251	-0.218 (13%)	0.251	0.222 (12%)	5.93%
(3)	50	0.146	0.109 (25%)	-0.297	-0.192 (35%)	0.297	0.250 (16%)	5.46%
(3)	100	0.136	0.097 (29%)	-0.287	-0.188 (35%)	0.287	0.236 (18%)	12.72%
(3)	150	0.130	0.086 (34%)	-0.278	-0.188 (33%)	0.278	0.220 (21%)	20.59%
(4)	50	0.037	0.013 (65%)	-0.161	-0.048 (70%)	0.161	0.089 (45%)	46.00%
(4)	100	0.038	0.011 (70%)	-0.164	-0.028 (83%)	0.164	0.084 (49%)	62.00%
(4)	150	0.037	0.011 (70%)	-0.161	-0.011 (93%)	0.161	0.082 (49%)	65.33%

En cuanto a los resultados, todos los métodos se vieron afectados por el aumento de la dimensionalidad de un solo input a múltiples inputs, ya que el MSE aumenta a medida que aumenta el número de inputs. Además, el MSE del método EAT podado fue menor que el MSE de la técnica FDH para todos los escenarios y tamaños de muestra. Las mejoras oscilaron entre el 13% y el 70%. Se observa una cierta tendencia en los resultados. En particular, cuanto mayor es el tamaño de la muestra, mayor es la mejora. Además, el escenario con nueve inputs es el que presenta mayores diferencias entre los dos enfoques considerados en este análisis. En cuanto al sesgo, EAT podado supera a FDH en todas las experiencias computacionales realizadas, con una reducción que va del 12% al 93%, en el caso de la primera medida de sesgo calculada, y del 10% al 49% en el caso del sesgo basado en el valor absoluto. Por último, la medida de la discrepancia entre EAT profundo y FDH muestra que es cero en el contexto de considerar sólo un input (escenario 1), algo que se esperaba a raíz de la Proposición 4.4; relativamente baja en el caso del escenario 2 basado en dos inputs; y es más alta en el caso del tercer y cuarto escenarios con más inputs. En todos los casos, la discrepancia observada aumenta a medida que aumenta el tamaño de la muestra. En la Figura 14 se muestra un ejemplo del resultado de una de nuestras simulaciones.

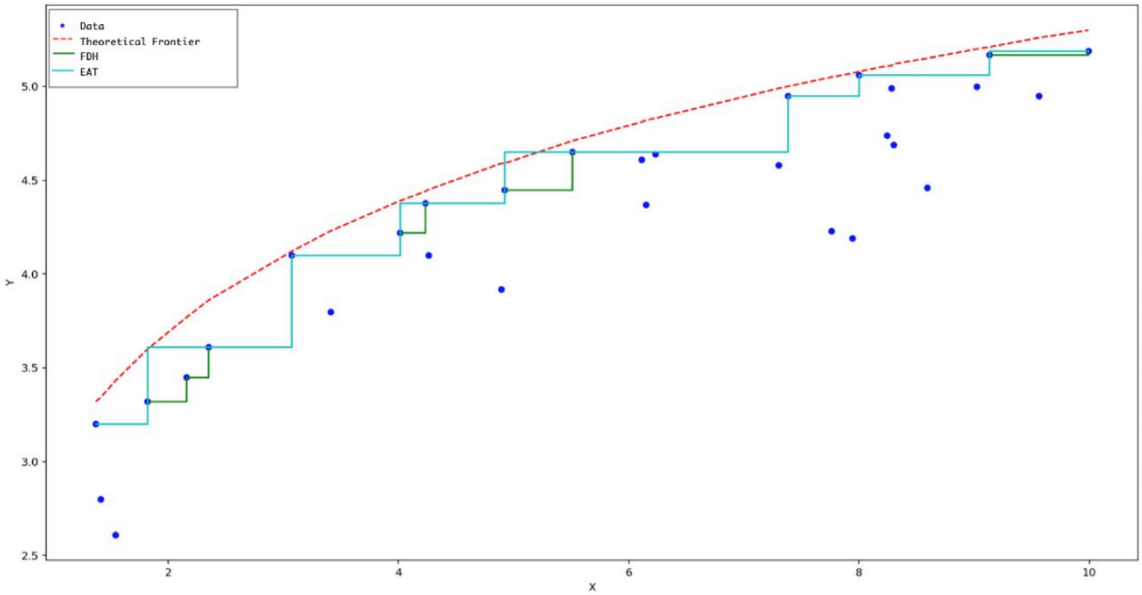


Figura 14. Ejemplo ilustrativo de los resultados obtenidos para una de las simulaciones de FDH vs EAT.

Otra característica de la técnica EAT es que la frontera estimada puede ser ilustrada fácilmente de manera gráfica aprovechando su estructura de árbol. Esto es importante desde el punto de vista de la visualización de los datos, y esta característica contrasta con las limitaciones de FDH en más de dos o tres dimensiones. Por ejemplo, en la Figura 15 se muestra el árbol final obtenido después de ejecutar el algoritmo correspondiente a la técnica EAT (podada) para uno de los experimentos asociados al escenario 3, en el que se consumen tres inputs para producir un output. En cada nodo, se puede encontrar un número de identificación, el error cuadrático medio, el tamaño de la muestra, la división ejecutada y la estimación del output. Una etiqueta como "X3 < 7.69 | X3 ≥ 7.69" significa que el nodo hijo izquierdo está asociado a los ejemplos del nodo padre que satisfacen la condición $x_3 < 7.69$, mientras que el nodo hijo derecho está relacionado con los ejemplos que cumplen $x_3 \geq 7.69$. En la parte inferior del árbol están los nodos terminales (las hojas) donde aparece la estimación final de la variable respuesta (el output). De esta manera, se puede visualizar la frontera escalonada visitando las diferentes ramas del árbol de arriba a abajo.

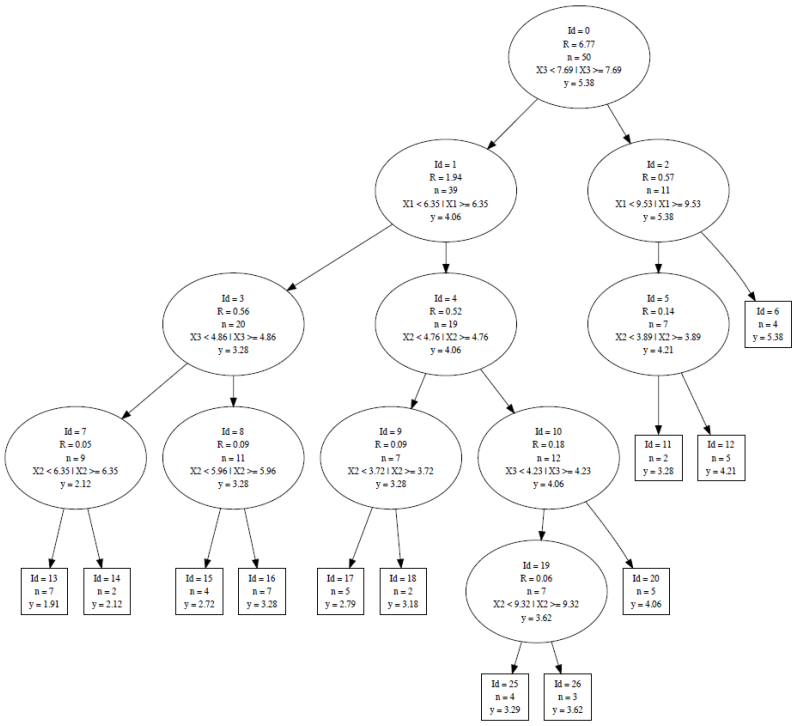


Figura 15. Ejemplo gráfico de la estructura de árbol de uno de los experimentos ejecutados para FDH vs EAT.

Las Tabla 3 y Tabla 4 muestran los resultados asociados a la simulación de múltiples inputs y outputs. La estructura de estas tablas es similar a la de la Tabla 2, excepto por el hecho de que ahora se considera el porcentaje de DMUs en la frontera teórica y la posibilidad de ruido aleatorio. Una vez más, se observa que EAT supera a FDH con respecto al MSE y al sesgo.

Tabla 3. Resultados sin ruido de las simulaciones multi-output para FDH vs EAT.

Núm. Obs.	%. de obs. en la front. teórica	Error cuadrático medio			Sesgo absoluto		
		FDH	EAT profundo	EAT	FDH	EAT profundo	EAT
50	25	0.394	0.279	0.346 (12%)	0.398	0.327	0.339 (15%)
50	10	0.481	0.361	0.368 (24%)	0.480	0.403	0.371 (23%)
50	0	0.567	0.431	0.405 (29%)	0.549	0.473	0.411 (25%)
100	25	0.309	0.187	0.190 (38%)	0.357	0.265	0.258 (28%)
100	10	0.396	0.253	0.240 (39%)	0.440	0.341	0.304 (31%)
100	0	0.506	0.336	0.250 (51%)	0.525	0.419	0.341 (35%)
200	25	0.221	0.111	0.121 (45%)	0.301	0.203	0.205 (32%)
200	10	0.311	0.179	0.159 (49%)	0.394	0.284	0.248 (37%)
200	0	0.398	0.237	0.175 (56%)	0.474	0.357	0.287 (39%)

Tabla 4.Resultados con ruido de las simulaciones multi-output para FDH vs EAT.

Núm. Obs.	%. de obs. en la front. teórica	Error cuadrático medio			Sesgo absoluto		
		FDH	EAT profundo	EAT	FDH	EAT profundo	EAT
50	25	0.398	0.263	0.380 (4%)	0.398	0.318	0.342 (14%)
50	10	0.506	0.371	0.368 (27%)	0.478	0.396	0.369 (23%)
50	0	0.596	0.440	0.429 (28%)	0.555	0.468	0.418 (25%)
100	25	0.301	0.182	0.208 (31%)	0.349	0.258	0.265 (24%)
100	10	0.413	0.257	0.261 (37%)	0.447	0.341	0.313 (30%)
100	0	0.482	0.306	0.258 (46%)	0.515	0.400	0.342 (34%)
200	25	0.481	0.361	0.368 (24%)	0.480	0.403	0.371 (23%)
200	10	0.313	0.171	0.137 (56%)	0.390	0.271	0.240 (38%)
200	0	0.400	0.228	0.198 (50%)	0.471	0.343	0.291 (38%)

Cabe mencionar que un inconveniente del nuevo enfoque en comparación con la técnica de FDH es el tiempo de computación empleado. En esta sección, los experimentos se llevaron a cabo en un ordenador con un procesador Intel Core i5 de doble núcleo con CPU 2.6 GHz y RAM 8 GB y sistema operativo OS X 10.12.1. El algoritmo se implementó en código Python (disponible como software libre bajo la licencia GNU General Public License version 3 en el repositorio GitHub https://github.com/MiriamEsteve/EATpy). Se utilizó CPLEX v12.8 como núcleo para resolver los problemas de optimización; se utilizaron las opciones predeterminadas. En lo que respecta al tiempo de ejecución, para una instancia que constaba de dos outputs, dos inputs y 200 DMU, la técnica FDH utilizó 20.04 segundos para obtener todos los puntajes de eficiencia, mientras que la técnica EAT utilizó 85.55 segundos (aproximadamente cuatro veces más). Por este motivo, en el Capítulo 4 Sección 4.3. se propone una mejora algorítmica que, en un trabajo futuro y presente, permita su paralelización y, por tanto, reducir el tiempo computacional empleado para obtener los resultados.

Finalmente, se centra nuestra atención en la "maldición de la dimensionalidad" (Charles et al., 2019, Ruggiero, 2005 y Nataraja and Johnson, 2011). En el caso de las medidas de eficiencia técnica, este problema está relacionado con la falta de discriminación entre las DMUs eficientes e ineficientes. Es un inconveniente que suele estar relacionado con el uso de demasiados inputs y outputs en comparación con el tamaño de la muestra del problema. A continuación, se muestra cómo funciona el algoritmo EAT cuando se aplica a un conjunto de datos reales de 15 DMUs con el uso de 6 inputs para producir 6 outputs, aunque este es un tema que merece una investigación más profunda y por ello se aborda en el Capítulo 4 Sección 4.2. con la adaptación del EAT al Random Forest de Breiman (2001). Para este ejemplo, se retoma el conjunto de datos de las 15 mejores ciudades de los EE.UU. de la revista Fortune en 1996 (véase Charles et al., 2019). En el ejemplo, en la Tabla 5, la técnica FDH muestra todas las DMUs como técnicamente eficientes, mientras que el nuevo enfoque puede determinar una puntuación que ayude a discriminar entre las ciudades eficientes e ineficientes (sólo 5 de 15 son técnicamente eficientes).

Tabla 5. Resultados de eficiencia de las 15 mejores ciudades de los EE.UU. de la revista Fortune en 1996.

DMU	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4	y_5	y_6	**FDH score**	**EAT score**
Seattle	586,000	581	1.45	4.50	21	542.3	46,928	0.297	4.49	7	117	22	1.00	1.00
Denver	475,000	558	0.97	4.00	14	595.6	42,879	0.291	2.79	5	60	71	1.00	1.09
Philadelphia	201,000	600	1.50	4.75	21	693.6	43,576	0.227	3.64	25	216	166	1.00	1.00
Minneapolis	299,000	609	1.49	4.00	24	496.5	45,673	0.270	2.67	6	131	125	1.00	1.03
Ral-Durham	318,000	613	0.99	4.50	18	634.7	40,990	0.319	4.94	7	33	47	1.00	1.00
St. Louis	265,000	558	0.89	3.00	18	263.0	39,079	0.206	3.40	10	104	62	1.00	1.20
Cincinnati	467,000	580	1.25	3.75	20	551.5	38,455	0.199	2.80	4	71	94	1.00	1.22
Washington	583,000	625	1.29	3.75	33	714.5	54,291	0.373	3.35	30	148	105	1.00	1.00
Pittsburgh	347,000	535	0.99	3.75	17	382.1	34,534	0.188	3.66	8	124	112	1.00	1.35
Dallas-FW	296,000	650	1.50	5.00	18	825.4	41,984	0.271	1.96	3	98	77	1.00	1.12
Atlanta	600,000	740	1.19	6.75	20	846.6	43,249	0.263	2.23	9	118	102	1.00	1.09
Baltimore	575,000	775	0.99	3.99	18	1296.3	43,291	0.233	4.02	8	102	45	1.00	1.08
Boston	351,000	888	1.09	4.25	34	686.6	46,444	0.325	5.69	25	240	55	1.00	1.00
Milwaukee	283,000	727	1.53	3.50	26	518.9	41,841	0.214	3.11	6	52	50	1.00	1.12
Nashville	431,000	695	1.19	4.00	26	1132.5	40,221	0.215	3.25	4	37	37	1.00	1.17

4.2. RANDOM FOREST PARA LOS ÁRBOLES DE ANÁLISIS DE EFICIENCIA

En este capítulo de tesis, se aplica el enfoque de Esteve et al. (2020) al contexto del uso de conjuntos (*ensembles*, en inglés) para proporcionar estimaciones en el campo del aprendizaje automático. En particular, se introduce cómo adaptar la técnica estándar de Random Forest (Breiman, 2001) para estimar las fronteras de producción en microeconomía e ingeniería satisfaciendo ciertos postulados estándar, como la libre disponibilidad. En lugar de utilizar árboles entrenados por CART (Breiman et al., 1984), nuestro enfoque utilizará árboles entrenados por EAT (Esteve et al., 2020), lo que permitirá estimar las fronteras de producción. Por la naturaleza del Random Forest, esta nueva propuesta será robusta al remuestreo de datos y variables de entrada.

La robustez ha sido un tema de gran interés en la literatura sobre la medición de la eficiencia técnica. En cuanto a la robustez de los datos observados, Simar y Wilson (1998, 2000a, 2000b) fueron los primeros en adaptar la metodología de bootstrapping (Efron, 1979) al contexto de DEA y FDH. En el caso de estos autores, el objetivo principal era aproximar la distribución muestral de las puntuaciones de eficiencia. En este caso particular, se recurre a una técnica denominada Bagging en Aprendizaje Estadístico, que significa Agregación Bootstrap (LeBlanc y Tibshirani, 1996; Mojirsheibani, 1997, 1999; Berk, 2016), con el objetivo de determinar buenas predicciones de las variables respuesta a partir de un conjunto de predictores y una muestra de datos de aprendizaje. Concretamente, el Bagging es un procedimiento que explota conjuntos de valores entrenados sobre muestras aleatorias de los datos. El Bagging no es una forma de llegar a un modelo en particular. En su lugar, es un algoritmo que puede ayudar a mejorar el rendimiento de los valores entrenados a partir de un enfoque estadístico determinado. El algoritmo detrás de Bagging funciona de la siguiente manera. El algoritmo realiza varias pasadas sobre los datos (muestras bootstraps) en las que, en cada pasada, los predictores se vinculan a las variables respuesta a través de un determinado enfoque estadístico, por ejemplo, entrenando los Árboles de Decisión. El procedimiento vinculado al Bagging

termina cuando se recogen/agrupan las predicciones de todas las muestras individuales para obtener una predicción final de las variables respuesta a partir de un vector de los predictores. En los problemas de regresión, en los que las variables respuesta son de naturaleza continua, las predicciones individuales suelen agregarse aplicando la media aritmética. La agregación de los resultados de muchas pasadas de los datos puede tener importantes beneficios desde una perspectiva estadística. El promedio sobre conjuntos de valores entrenados puede aumentar su estabilidad. El proceso de promediación tiende a anular los resultados respaldados por las características idiosincrásicas de los datos. Además, el Bagging puede tener consecuencias beneficiosas para el equilibrio entre el sesgo y la varianza. Aunque, por lo general, el objetivo principal de esta técnica es la reducción de la varianza de los valores entrenados, en determinadas circunstancias puede haber también una disminución del sesgo. La gran diferencia entre Random Forest, que también explota el Bagging, y el Bagging "puro" es que el primero también utiliza el remuestreo en los predictores, mientras que el segundo no. En otras palabras, la principal diferencia entre estas dos técnicas es la elección del tamaño del subconjunto de predictores en cada nodo que se va a dividir en cada árbol a lo largo del procedimiento. De hecho, si Random Forest se construye utilizando todos los predictores disponibles, entonces equivale al Bagging “puro”.

En cuanto a la importancia de la robustez de las variables de entrada y salida en la medición no paramétrica de la eficiencia, desde el comienzo de DEA y FDH, los investigadores han sido conscientes de que la selección de las variables de entrada y salida que deben considerarse en el análisis de la eficiencia es uno de los temas cruciales de la especificación del modelo. En la práctica, la experiencia previa de los investigadores puede llevar a la selección de algunos inputs y outputs considerados esenciales para representar la tecnología subyacente. Sin embargo, puede haber otras variables de cuya inclusión en el modelo el analista no siempre esté seguro (Pastor et al., 2002). Esta situación se ha abordado de diferentes maneras en la literatura. Por una parte, algunos enfoques se centran en proponer un mecanismo para seleccionar los inputs y outputs adecuados a partir de un conjunto de variables. La idea principal es equilibrar la

experiencia de los investigadores con la información proporcionada por las observaciones (véase, por ejemplo, Banker, 1993, 1996; Pastor et al., 2002). Por otro lado, otro método se basa en la determinación de las puntuaciones de eficiencia que sean robustas frente a la selección de las variables. En esta línea, se puede encontrar un documento reciente de Landete et al. (2017), en el que los autores consideran sistemáticamente todos los posibles escenarios asociados a todas las especificaciones de los inputs y outputs que podrían definirse a partir del conjunto original de inputs y outputs. La inclusión de un input/output en el conjunto de variables seleccionadas se modela a través de la probabilidad de que esa variable sea considerada en el modelo DEA. En Landete et al. (2017), la puntuación robusta de la eficiencia para una unidad dada se define entonces como el valor esperado de esa variable aleatoria. La consideración de todas las combinaciones de inputs/outputs da lugar a un número exponencial de problemas que deben ser resueltos.

Hasta donde sabemos, este trabajo abordado en la tesis es el primero que se centra en la introducción de una metodología para medir la eficiencia técnica que es robusta frente al remuestreo de datos y, al mismo tiempo, frente a la especificación de las variables de entrada. Esta doble robustez se logrará adaptando Random Forest a la estimación de las fronteras de producción. En comparación con la técnica de Bagging, el Random Forest comparte el mismo conjunto de predictores para ajustar cada árbol individual de cada submuestra de bootstrap. Este conjunto común de variables predictoras disponibles puede crear una dependencia adicional, lo que podría producir que el proceso de promediación vinculado al Bagging no fuera tan eficaz como se esperaba. German et al. (1992) fueron los primeros en mostrar cómo el error de generalización esperado de un algoritmo de aprendizaje automático puede descomponerse en ruido puro, sesgo y varianza del predictor. Como señaló Breiman (2001), un enfoque sensato para disminuir el error de generalización consiste en reducir la varianza de la predicción, siempre que el sesgo respectivo pueda fijarse o, al menos, no se aumente excesivamente. En particular, los métodos de conjunto, como el de Random Forest, es una forma de hacer precisamente eso. La idea que subyace a Random Forest es introducir perturbaciones aleatorias en el proceso de aprendizaje para obtener diferentes modelos a partir de una única muestra de

aprendizaje y luego combinar las predicciones de cada modelo individual para formar la predicción final del conjunto (Louppe y Geurts, 2012, Louppe, 2014). Como muestra Louppe (2014), la varianza del predictor, cuando el número de elementos del conjunto es aleatoriamente grande, es igual al producto de la correlación entre las predicciones de dos elementos aleatorios cualesquiera del conjunto y la varianza de la predicción de cualquier elemento aleatorio, bajo la hipótesis de independencia e idéntica distribución de las variables aleatorias que controlan la aleatoriedad del algoritmo de aprendizaje. Además, el ruido y los términos de sesgo en la descomposición del error de generalización permanecen constantes e idénticos al ruido y al sesgo de cualquiera de los modelos aleatorios que componen el conjunto. Por lo tanto, la clave en los métodos de conjunto es introducir perturbaciones aleatorias para descorrelacionar en la mayor medida posible las predicciones de los modelos individuales ya que, de esta manera, la correlación entre las predicciones de dos elementos aleatorios cualesquiera del conjunto tomarán un valor negativo, lo que significa que la varianza del conjunto será estrictamente menor que la varianza de un modelo individual. En nuestros contextos, esto implica que, bajo el principio de incorrelación, Random Forest presentará un error de generalización estrictamente menor que CART ejecutado en una muestra particular (bootstrap) (ver, también, Geurts, 2002, para una descomposición alternativa de la varianza). En este capítulo de la tesis, se explota este resultado para mejorar la potencia de EAT para estimar las fronteras eficientes, adaptando la técnica de Random Forest al contexto de la producción. Véanse también Kwok y Carter (1990), Breiman (1994), Dietterich y Kong (1995), Amit et al. (1997), Ho (1998), Cutler y Zhao (2001), Geurts et al. (2006) y Rodríguez et al. (2006), donde se han propuesto diferentes conjuntos y árboles de decisión para mejorar el error de generalización de los modelos individuales.

A continuación, el nuevo enfoque Random Forest para EAT se divide en dos secciones, una para introducir esta adaptación y otra para mostrar los resultados obtenidos a partir de una experiencia computacional. Tras demostrar con estos resultados que la nueva técnica RF+EAT es adecuada se aborda la forma de crear una lista ordenada con la importancia de las variables de entrada y la resolución para el problema de la maldición

de la dimensionalidad, a través de ejemplos numéricos, que puede lograr Random Forest estándar y, por tanto, RF+EAT.

4.2.1. RANDOM FOREST PARA EAT (RF+EAT)

En esta sección se presenta el algoritmo asociado a la adaptación de la técnica del Random Forest (Breiman, 2001) al mundo de la evaluación de la eficiencia técnica, llamado aquí RF+EAT.

Input of the algorithm:

p, number of trees

$\aleph$, original data with m inputs and s outputs

$mtry$, number of inputs to selected for each split

n_{min}, max number of observations in a leaf nodes

Output of the algorithm:

$RF^{EAT} := \{ T^{EAT}(\aleph_q): q = 1, \ldots, p \}$, Random Forest + EAT model

$\{ \aleph_q : q = 1, \ldots, p \}$

$RF^{EAT} := \{\}$

FOR q: = 1 **TO** p

  $\aleph_q :=$ Bootstrap sample of $\aleph$

  $T^{EAT}(\aleph_q) := \{ t_0 : t_0 =$ root node $\}$

  FOR EACH t in $T^{EAT}(\aleph_q)$: t is a leaf node $\Leftrightarrow n(t) \leq n_{min}$

    Randomly selects $mtry$ ($\leq m$) of the original inputs

    Select the best input among the $mtry$ inputs for t and split the data in t_L and t_R nodes

    $T^{EAT}(\aleph_q) := T^{EAT}(\aleph_q) \cup \{ t_L , t_R \}$

  END FOR

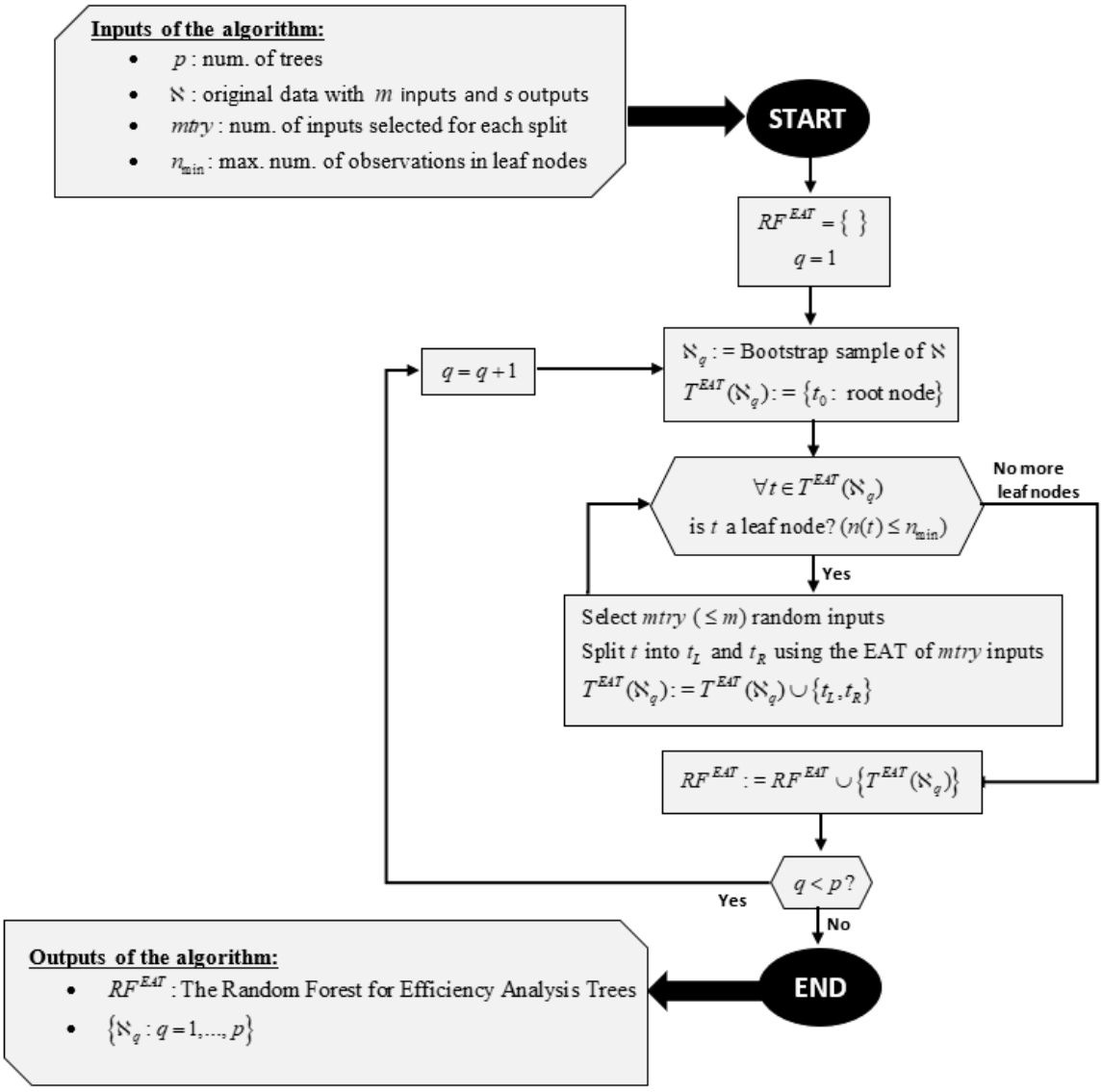
  $RF^{EAT} := RF^{EAT} \cup \{ T^{EAT}(\aleph_q) \}$

END FOR

Algoritmo 2. Algoritmo RF+EAT.

En la primera etapa, se selecciona el número de árboles que conformarán el bosque (*forest*, en inglés) o conjunto de árboles, éste es el hiperparámetro p. Después, se generan p (bootstrap) muestras aleatorias con reemplazamientos a partir de la muestra original

de datos. Seguidamente, se ejecuta el algoritmo EAT sobre cada submuestra, sin podar pero usando la regla de parada $n(t) \leq n_{\min}$. En este algoritmo, $n_{\min}$ se trata como un hiperparámetro que podría ser modificado, tal y como sucede con p (ver *Flowchart 1*).



Flowchart 1. Pasos del algoritmo RF+EAT.

Durante la ejecución del algoritmo EAT, se selecciona aleatoriamente un subconjunto de variables de entrada del conjunto original cada vez que se llama a la subrutina de división. Para ello, se utiliza una de las siguientes cinco reglas generales:

- Regla de Breiman: $mtry = m/3$
- Regla DEA1: $mtry = \frac{n(t)}{2} - s$
- Regla DEA2: $mtry = \frac{n(t)}{3} - s$
- Regla DEA3: $mtry = \frac{n(t)}{2s}$
- Regla DEA4: $mtry = \min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$

La primera regla fue sugerida por Breiman (2001) y es el valor habitual utilizado en Random Forest estándar para el problema de regresión. El número de inputs seleccionados al azar debe ser un tercio del número de inputs totales. Como de costumbre, el valor obtenido debe redondearse hacia abajo, y se aplica lo mismo para todas las reglas restantes. Nótese también que esta fórmula no depende de $n(t)$, es decir, el tamaño de la muestra que contiene el nodo (padre) que va a ser dividido. Por el contrario, se sugieren otras cuatro reglas diferentes que sí dependen de $n(t)$ y todas ellas provienen de la literatura que trata sobre el establecimiento de una buena relación entre el tamaño de la muestra y el número de variables en DEA. De hecho, la literatura indica algunas reglas empíricas con respecto al número de datos frente al número de inputs y outputs. La Regla DEA1 procede de Golany y Roll (1989) y Homburg (2001), que sugieren que el número de observaciones debe ser al menos el doble del número de inputs y outputs. La Regla DEA2 procede de los artículos de Nunamaker (1985), Banker et al. (1989), Friedman y Sinuany-Stern (1998) y Raab y Lichty (2002), que señalan que el número de DMUs debería ser de al menos el triple del número de inputs y outputs. Por otro lado, Dyson et at. (2001) sugirieron que el número de unidades debería ser de al menos el doble del producto del número de inputs y el número de outputs, lo que generó la Regla DEA3. Por último, otra regla empírica es la de Cooper et al. (2007), que afirmaron $n \geq \max\{m \cdot s, 3 \cdot (m+s)\}$, que dio lugar a la regla DEA4. Obviamente, las reglas DEA1-DEA4 son nuevas, y no se han utilizado anteriormente en la literatura de Random Forest.

Una vez ejecutado el Algoritmo 2, se tiene p árboles entrenados para ser agregados con el propósito de obtener una estimación de salida dado un vector de entrada $\boldsymbol{x} \in R_+^m$. En este sentido, se tiene $T^{EAT}(\aleph_1),...,T^{EAT}(\aleph_p)$ estructuras de árbol derivadas de la aplicación del algoritmo EAT con las p submuestras bootstrap $\aleph_1,...,\aleph_p$. Dado un vector de inputs $\boldsymbol{x} \in R_+^m$, la salida del estimador asociado a cada estructura de árbol $T^{EAT}(\aleph_q)$ es denotada como $\boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x})$. Como es habitual en Random Forest, un estimador de producción para un vector $\boldsymbol{x} \in R_+^m$ se determina promediando el estimador individual correspondiente a cada árbol:

$$\boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x}) := \frac{1}{p}\sum_{q=1}^{p} \boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x}) \tag{17}$$

A partir de este estimador, es posible definir el conjunto de tecnología o posibilidad de producción inducida por la técnica de Random Forest como:

$$\hat{\Psi}_{RF+EAT} = \left\{(\boldsymbol{x},\boldsymbol{y}) \in R_+^{m+s} : \boldsymbol{y} \le \boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x})\right\} \tag{18}$$

Recurriendo a ciertos resultados en Esteve et al. (2020), no es difícil demostrar que el conjunto de posibilidades de producción inducido por Random Forest cumple con la propiedad clásica en la teoría de la producción de libre disponibilidad.

Proposición 5.1. $\hat{\Psi}_{RF+EAT}$ satisface la libre disponibilidad en inputs y outputs.

Demostración. Sea $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\Psi}_{RF+EAT}$ y $(\boldsymbol{x}',\boldsymbol{y}') \in R_+^{m+s}$ tal que $\boldsymbol{x}' \ge \boldsymbol{x}$ e $\boldsymbol{y}' \le \boldsymbol{y}$. Se prueba que $(\boldsymbol{x}',\boldsymbol{y}') \in \hat{\Psi}_{RF+EAT}$. Se sabe que la tecnología inducida por la aplicación del algoritmo

EAT sobre cada submuestra de bootstrap $\aleph_q$ es $\left\{(\boldsymbol{x},\boldsymbol{y})\in R_+^{m+s}:\boldsymbol{y}\le \boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x})\right\}$. Por como la estimación de cada componente del vector de salida $\boldsymbol{y}(t)$ se actualiza para cada nodo bajo el algoritmo EAT, se tiene que si $\boldsymbol{x}'\ge \boldsymbol{x}$, entonces $\boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x}')\ge \boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x})$ (ver, Esteve et al., 2020). De esta manera, por (17), se tiene que $\boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x}')\ge \boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x})$. Por consiguiente, $\boldsymbol{y}'\le \boldsymbol{y} \underbrace{\le}_{\text{porque } (\boldsymbol{x},\boldsymbol{y})\in\hat{\Psi}_{RF+EAT}}$ $\boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x})\le \boldsymbol{y}^{RF+EAT(\aleph)}(\boldsymbol{x}')$, que implica, por definición, $(\boldsymbol{x}',\boldsymbol{y}')\in\hat{\Psi}_{RF+EAT}$. ■

Sin embargo, la técnica de Random Forest para estimar las fronteras de producción no cumple la propiedad de la mínima extrapolación, que sí satisface el FDH y que es culpable de su clásico problema de sobreajuste. Además, por su naturaleza, EAT genera un estimador de la frontera de producción que tiene una forma escalonada y, en consecuencia, el conjunto de posibilidades de producción inducido a partir de EAT no cumple con la convexidad. De esta manera, EAT compite con FDH y, por tanto, la adaptación de Random Forest que se introduce en este capítulo de tesis también lo hace.

En RF+EAT, la puntuación (radial) orientada al output de la eficiencia $\phi(\boldsymbol{x}_k,\boldsymbol{y}_k)=\max\{\phi_k\in R:(\boldsymbol{x}_k,\phi_k\boldsymbol{y}_k)\in\Psi\}$ para $\boldsymbol{x}_k\in R_+^m$ e $\boldsymbol{y}_k\in R_{++}^s$ puede ser estimada sustituyendo la tecnología teórica Ψ por su estimación $\hat{\Psi}_{RF+EAT}$, es decir, $\phi^{RF+EAT}(\boldsymbol{x}_k,\boldsymbol{y}_k)=\max\left\{\phi_k\in R:(\boldsymbol{x}_k,\phi_k\boldsymbol{y}_k)\in\hat{\Psi}_{RF+EAT}\right\}$. El siguiente resultado establece la forma de calcular el valor de $\phi^{RF+EAT}(\boldsymbol{x}_k,\boldsymbol{y}_k)$ en la práctica.

Proposición 5.2. Dado $\boldsymbol{x}_k \in R_+^m$ e $\boldsymbol{y}_k \in R_{++}^s$. Entonces, $\phi^{RF+EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right) = \min_{r=1,\ldots,s}\left\{\frac{y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{rk}}\right\}$.

Demostración. El valor máximo en $\max\left\{\phi_k \in R : \left(\boldsymbol{x}_k, \phi_k \boldsymbol{y}_k\right) \in \hat{\Psi}_{RF+EAT}\right\}$ se alcanza realmente ya que (i) $\left\{\phi \in R : \left(\boldsymbol{x}_k, \phi \boldsymbol{y}_k\right) \in \hat{\Psi}_{RF+EAT}\right\} = \left\{\phi \in R : \phi \boldsymbol{y}_k \le \boldsymbol{y}^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)\right\}$ es un politopo y (ii) si una función lineal está acotada superiormente en un politopo, entonces alcanza su supremo en el politopo (Mangasarian, 1994, p. 130). Por consiguiente, $\left(\boldsymbol{x}_k, \phi^{RF+EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right)\boldsymbol{y}_k\right) \in \hat{\Psi}_{RF+EAT}$. Entonces, por (18), $\phi^{RF+EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right)\boldsymbol{y}_k \le \boldsymbol{y}^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)$. De ahí que, $\phi^{RF+EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right) \le \min_{r=1,\ldots,s}\left\{\frac{y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{rk}}\right\}$. Además, se tiene que $\min_{t=1,\ldots,s}\left\{\frac{y_t^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{tk}}\right\} y_{rk} \le \frac{y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{rk}} y_{rk} = y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)$, para todo $r = 1,\ldots,s$. Esto implica que $\left(\boldsymbol{x}_k, \min_{r=1,\ldots,s}\left\{\frac{y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{rk}}\right\}\boldsymbol{y}_k\right) \in \hat{\Psi}_{RF+EAT}$ y, también, $\phi^{RF+EAT}\left(\boldsymbol{x}_k, \boldsymbol{y}_k\right) = \min_{r=1,\ldots,s}\left\{\frac{y_r^{RF+EAT(\aleph)}\left(\boldsymbol{x}_k\right)}{y_{rk}}\right\}$. ■

4.2.2. EXPERIENCIA COMPUTACIONAL

En esta sección se muestra los resultados de las simulaciones que sirven para comparar los métodos: FDH, EAT y RF+EAT. Para ello, se presenta una comparación sistemática de estos tres métodos de estimación de frontera en una experiencia computacional, tal y como se hizo en el Capítulo 4 Sección 4.1.3. Los datos simulados provienen de la típica función Cobb-Douglas de un solo output en microeconomía, jugando con diferentes tamaños de muestra: 50, 75 y 100 unidades, y número de inputs: 6, 9, 12 y 15. Los datos de entrada fueron muestreados aleatoriamente siguiendo una distribución $Uni[1,10]$ con independencia de cada input y observación, y el término de ineficiencia siguió una

distribución normal truncada $\left|N\left(0,0.4\right)\right|$. Se hicieron 100 experimentos para cada combinación de tamaño de muestra y número de inputs. El rendimiento de cada método se evalúa con el error cuadrático medio (MSE) y el sesgo.

La Tabla 6 muestra las estadísticas de MSE y de sesgo de las tres técnicas evaluadas (FDH, EAT y RF+EAT). Las dos primeras columnas indican el número de observaciones y el número de inputs. La tercera columna corresponde al valor del parámetro $n_{\min}$. Se comprueban diferentes valores, siempre mayores o iguales a 5, y se muestran en la tabla el más pequeño que hace que la técnica RF+EAT genere mejores resultados que el enfoque EAT. Las siguientes tres columnas indican el MSE de las técnicas FDH, EAT y RF+EAT. Las tres columnas siguientes muestran el sesgo de las mismas tres metodologías. Además, también se ha calculado (y mostrado entre paréntesis) la diferencia relativa entre FDH y RF+EAT con respecto al MSE y el sesgo para facilitar la comparación de los resultados. Estos valores son los porcentajes de reducción del MSE y el sesgo cuando se aplica la nueva técnica RF+EAT en lugar de FDH. Merece la pena mencionar que, durante la experiencia de cálculo, se utilizaron diferentes configuraciones para el hiperparámetro p en el algoritmo RF+EAT. En concreto, se probaron con $p=500$ y $p=1000$ árboles, obteniendo resultados similares con respecto al MSE y el sesgo. Además, buscando la simplicidad, se muestran exclusivamente los resultados derivados de la aplicación de la regla de Breiman para definir $mtry$. También se probaron las otras cuatro reglas, obteniendo cifras similares.

Tabla 6. Resultados de las simulaciones para FDH vs EAT vs RF+EAT.

Núm. Obs.	#Inputs	n_{min}	Error cuadrático medio			Sesgo		
			FDH	EAT	RF+EAT	FDH	EAT	RF+EAT
50	6	5	3.565	0.920	0.814 (77.16%)	1.571	0.749	0.707 (55.01%)
50	9	5	3.836	0.861	0.760 (80.19%)	1.618	0.741	0.795 (50.86%)
50	12	5	4.120	1.224	0.541 (86.87%)	1.721	0.857	0.594 (65.51%)
50	15	5	3.903	1.427	0.369 (90.55%)	1.661	0.906	0.482 (70.99%)
75	6	5	2.998	1.258	1.129 (62.35%)	1.455	0.917	0.896 (38.40%)
75	9	5	4.036	1.126	1.075 (73.35%)	1.662	0.849	0.839 (49.51%)
75	12	5	3.677	0.921	0.644 (82.48%)	1.617	0.141	0.638 (60.56%)
75	15	5	3.583	1.300	0.425 (88.14%)	1.599	0.943	0.520 (67.48%)
100	6	10	3.917	1.401	1.181 (69.84%)	1.697	0.989	0.810 (52.30%)
100	9	15	4.229	1.585	1.294 (69.41%)	1.720	1.074	0.875 (49.10%)
100	12	5	3.566	1.032	0.766 (78.51%)	1.571	0.844	0.689 (56.12%)
100	15	5	3.923	1.137	0.495 (87.40%)	1.642	0.818	0.553 (66.32%)

En cuanto a los resultados, el MSE y el sesgo del método RF+EAT fueron claramente más pequeños que las mismas medidas de rendimiento para la técnica de FDH. Las mejoras oscilaron entre el 62% y el 91% en el caso de MSE, y entre el 38% y el 71% en el caso del sesgo. Se observa una cierta tendencia en los resultados. En particular, dado el tamaño de la muestra, cuanto mayor es el número de inputs, mayor es la mejora. Además, el nuevo enfoque, es decir, RF+EAT, supera la técnica EAT, mostrando, en general, mejores resultados tanto en MSE como en el sesgo. Una vez más, se puede identificar un patrón en este sentido. Con respecto a los valores observados para EAT en la Tabla 6, la mejora exhibida por RF+EAT es más intensa a medida que aumenta el número de variables. Por otra parte, se observa que el parámetro $n_{\min}$ cobra importancia en estas experiencias computacionales. Para los casos en los que la muestra contiene pocas DMUs (hasta 75 DMUs) la regla de parada se mantiene a 5 datos en los nodos hoja,

pero a medida que aumenta este tamaño $n_{\min}$ también debe de aumentar para lograr una mejora superior respecto al FDH.

Por último, en la Figura 16 se muestran las correspondientes curvas MSE y en la Figura 17 se muestran los gráficos de contorno de algunas de las simulaciones anteriores. Las curvas MSE permiten ver cómo converge el error del modelo con respecto al parámetro *p*.

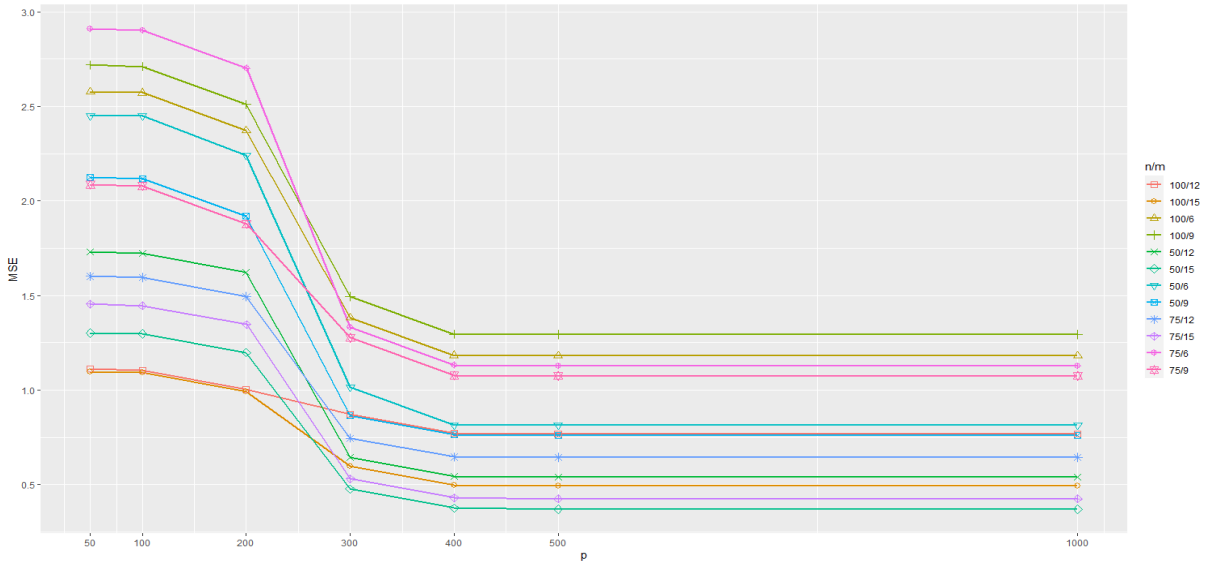


Figura 16. Curvas MSE.

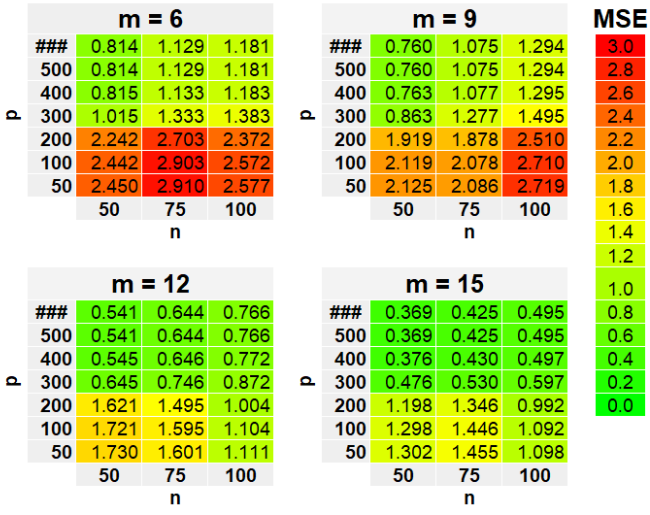


Figura 17. Gráfico de contorno del MSE.

4.2.3. CÁLCULO DE LA IMPORTANCIA DE LAS VARIABLES DE ENTRADA

En esta sección, se introduce una medida de la importancia de las variables de entrada, calculada a partir de un conjunto de árboles aleatorios de EAT. Se presenta primero cómo la adaptación de Random Forest al contexto de la teoría de la producción puede ser utilizada para evaluar la importancia de las variables de entrada. Seguidamente, se ilustra el uso de esta medida de importancia mediante un ejemplo empírico.

Una misión relevante en el aprendizaje automático es la predicción de una o varias variables respuesta basada en un conjunto de variables de predicción. Sin embargo, el objetivo principal no es sólo hacer la predicción más exacta de las variables respuesta, sino también identificar qué variables predictoras son las más pertinentes para hacer esas predicciones; principalmente con el objetivo de comprender el proceso fundamental que generó los datos (Louppe et al., 2013). En este sentido, Random Forest estándar es una técnica que permite calcular ese tipo de medida. En nuestro contexto, se adapta de nuevo esta medida para tratar la medición de la relevancia de los inputs cuando el foco de atención sea la estimación de las fronteras de producción. La medida permitirá determinar una clasificación de la importancia de las variables de entrada.

En cierto sentido, el cálculo de una medida de la importancia de los inputs está vinculado a la determinación de la significación estadística de esas variables en el ámbito de la medición de la eficiencia técnica basada en técnicas paramétricas, en las que se consideran ciertas distribuciones de probabilidad y permiten calcular *p*-valores e intervalos de confianza. En el mundo no paramétrico, como en el aprendizaje automático, es habitual no asumir ninguna distribución de probabilidad particular en la generación de los datos, lo que tiene como consecuencia la imposibilidad de determinar la medida del error típico de muestreo. En cambio, al tratar de imitar el mundo paramétrico, las técnicas de aprendizaje automático pueden dar lugar alternativamente a una clasificación de la

importancia de los predictores (inputs en nuestro contexto de producción) para estimar las variables respuesta (los outputs).

En el contexto de los árboles de decisión, como en CART (Breiman et al., 1984), ya existe una medida de la importancia de las variables predictoras. Se basa en la noción de divisiones subrogadas para una determinada variable. Se trata de la división más cercana definida en un predictor que puede imitar la división real definida en un nodo determinado. Las divisiones subrogadas se introdujeron para considerar los efectos de enmascaramiento. Por ejemplo, puede ocurrir en la construcción de un árbol que alguna variable $x_{j'}$ nunca sea seleccionada en ningún nodo como la variable de división, porque lleva a nodos hijos que son ligeramente peores en cuanto al error cuadrático medio en comparación con otro predictor $x_{j''}$. Sin embargo, si $x_{j''}$ se elimina y se genera un nuevo árbol, entonces $x_{j'}$ puede aparecer como el predictor más relevante en las divisiones, dando un árbol tan bueno como el original en cuanto a precisión. La medida de la importancia introducida por Breiman et al. (1984) sí tiene en cuenta estas situaciones mediante divisiones subrogadas (alternativas).

Seguidamente, se adapta al marco de la producción la medida de importancia que se suele utilizar en el estándar de Random Forest, véase la subsección 2.3. Breiman (2001) introdujo una forma de evaluar el papel de cada predictor en Random Forest que se basa en la aleatoriedad y no en divisiones subrogadas. En particular, se basa en la reducción de la exactitud de la predicción, cuando un predictor se "baraja" de manera que no puede hacer una contribución sistemática a la predicción de la variable de respuesta. En el problema de la regresión, el enfoque ordinario consiste en utilizar el aumento del error cuadrático medio para los datos del *Out-Of-Bag* (fuera de la muestra), que desempeña un papel fundamental en la evaluación de la calidad de las estimaciones proporcionadas por Random Forest. Mediante el proceso de doble aleatorización en Random Forest, se inhiben los efectos de enmascaramiento dentro del conjunto de árboles aleatorizados.

Siguiendo a Breiman (2001), se sugiere evaluar la importancia del input x_j determinando el Error de Incremento Medio del conjunto de árboles cuando los valores de x_j son permutados aleatoriamente. Para ello, se recurre al uso de las muestras *Out-Of-Bag*. En particular, para calcular la importancia del input x_j, se siguen los siguientes pasos. Generar una nueva base de datos, $\aleph^j$, idéntica a la original $\aleph$, donde las variables de entrada x_j se han permutado aleatoriamente. Este proceso de "barajado" hace que este input, en promedio, no esté relacionada con otros inputs y outputs. Aplicar la técnica adaptada de Random Forest (RF+EAT) sobre la muestra de aprendizaje $\aleph^j$. Determinar el valor del error de generalización vinculado a este último bosque (conjunto de árboles) como $err^{RF+EAT(\aleph^j)}$, donde

$$err^{RF+EAT(\aleph^j)} = \frac{1}{n}\sum_{(\boldsymbol{x}_i, y_i)\in\aleph^j}\sum_{r=1}^{s}\left(y_{ri} - y_r^{RF+EAT(\aleph^j)}(\boldsymbol{x}_i)\right)^2 \tag{19}$$

En (19), $y_r^{RF+EAT(\aleph^j)}(\boldsymbol{x}_i)$ es el *r*-ésimo componente del vector $\boldsymbol{y}^{RF+EAT(\aleph^j)}(\boldsymbol{x}_i) = \frac{1}{\left|K_i(\aleph^j)\right|}\sum_{q\in K_i(\aleph^j)} \boldsymbol{d}_{T^{EAT}(\aleph_q)}(\boldsymbol{x}_i)$ y, como antes, $K_i(\aleph^j)$ representa el conjunto de índices de los p árboles entrenados mediante EAT tal que las submuestras correspondientes no incluyen $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. Además, de manera similar, es necesario determinar el error de generalización del bosque aleatorio original $err^{RF+EAT(\aleph)}$. Por último, es posible calcular cuánto aumenta en porcentajes el error de generalización del modelo cuando la variable de x_j es permutada aleatoriamente de la siguiente manera:

$$\%Inc^{RF+EAT}(x_j) = 100\cdot\left(\frac{err^{RF+EAT(\aleph^j)} - err^{RF+EAT(\aleph)}}{err^{RF+EAT(\aleph)}}\right) \tag{20}$$

A continuación, se ilustra cómo funciona este enfoque a través de un ejemplo empírico. En particular, basamos esta ilustración se basa en una base de datos de 102 almacenes

que operan en la zona del Benelux en 2017 (véase Kaps y Koster, 2019). Siguiendo a Balk et al (2017), se utilizan tres inputs y cuatro outputs. Los inputs son: el tamaño del almacén en m^2 (superficie); el número de empleados a tiempo completo (ETC); y el número de unidades de mantenimiento de existencias (SKU). Los outputs son: número de líneas de pedido (líneas de pedido enviadas por día); porcentaje de líneas de pedido sin errores (porcentaje sin errores); flexibilidad de los pedidos (por día); y número de procesos especiales (manejados por día). Véase en Aparicio y Zofio (2020) una descripción de la muestra de datos.

Después de ejecutar el modelo estándar FDH (4) y el nuevo algoritmo RF+EAT (Algoritmo 2), se obtienen estimaciones de las puntuaciones de las 102 unidades evaluadas. La Figura 18 es la representación gráfica de las densidades vinculadas a las puntuaciones determinadas por ambas metodologías. Los resultados muestran claras diferencias entre las puntuaciones estimadas, siguiendo diferentes patrones. Como es habitual, FDH identifica muchas unidades como técnicamente eficientes, algo que ya no ocurre con el nuevo enfoque.

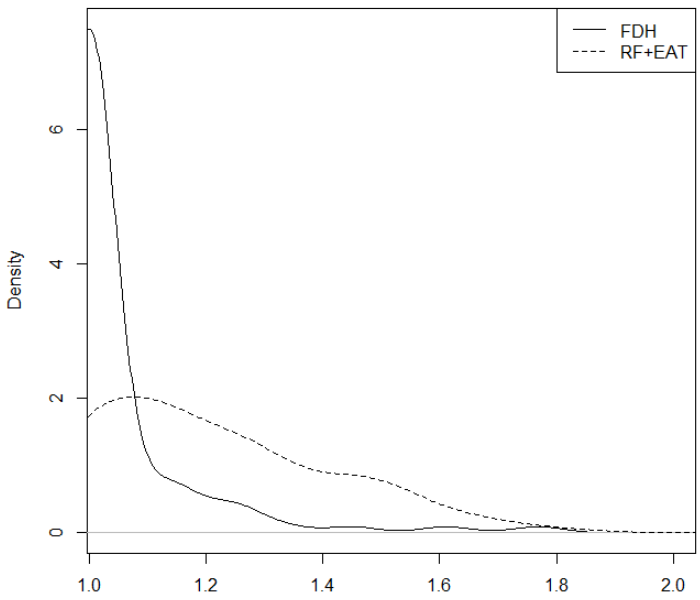


Figura 18. Densidades de las puntuaciones asociados con FDH y RF+EAT

En cuanto a la importancia de las variables, cuando lo que preocupa es la determinación de los resultados de los inputs, el enfoque RF+EAT es capaz de producir una clasificación como la que se discutió anteriormente. En la Figura 19 se ilustra gráficamente el valor de importancia de cada input, calculado a partir de la expresión (20). Según los resultados, la superficie y el número de unidades de mantenimiento de existencias son muy similares en cuanto a su nivel de importancia, siendo el número de empleados equivalentes a tiempo completo la variable de entrada más importante. En particular, el error de generalización aumenta más del 80% cuando este último input se permuta aleatoriamente, lo que, en cierto sentido, equivale a eliminarla del modelo.

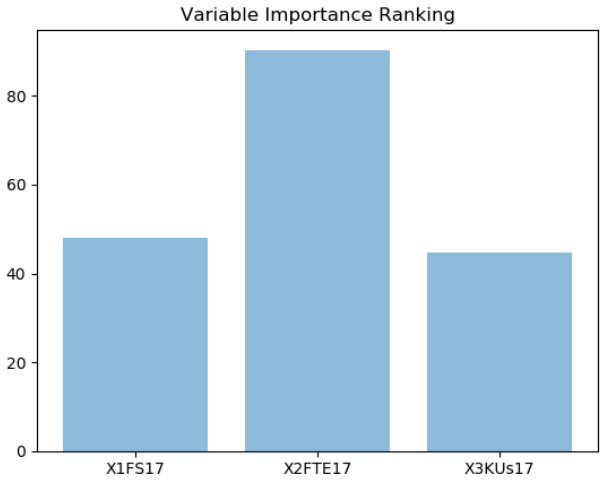


Figura 19. Representación gráfica del ranking de la importancia de las variables de entrada.

4.2.4. RESOLUCIÓN DEL PROBLEMA DE LA MALDICIÓN DE LA DIMENSIONALIDAD

Cualquier profesional podría creer que a medida que aumenta el número de predictores utilizados para ajustar un modelo, la precisión del modelo ajustado también aumentará. Sin embargo, este no es necesariamente el caso. En términos generales, la adición de predictores adicionales que estén fuertemente relacionados con la respuesta mejorará la calidad del modelo ajustado, al reducir el error de generalización. Sin embargo, la inclusión de variables que no están realmente asociadas con la respuesta dará lugar a un deterioro del modelo ajustado, y a un aumento del error de generalización. La razón es

que la adición de variables de "ruido" (aquellas que no tienen una relación real con la variable respuesta) aumenta la dimensionalidad del problema, agravando el riesgo de sobreajuste (ya que las variables de ruido pueden parecer relevantes en el modelo como consecuencia del azar relacionado con la estructura de los valores de la variable respuesta y los predictores de los datos de entrenamiento) sin ningún beneficio potencial en términos de reducción real del error de generalización. Así pues, el manejo de muchas variables puede conducir a modelos mejorados si los predictores son de hecho pertinentes para el problema, pero de otro modo conducirá a peores resultados (James et al., 2013). En particular, en el caso del análisis envolvente de datos o del *Free Disposal Hull*, si la relación entre el tamaño de la muestra y el número de variables (inputs y outputs) es baja o moderadamente baja, los modelos de eficiencia pueden dar lugar a que se evalúe un número considerable de unidades como técnicamente eficientes; especialmente en el caso de FDH. Esta falta de discriminación se denomina en la literatura "maldición de la dimensionalidad". En esta sección del Capítulo 4 Sección 4.2., se muestra que RF+EAT puede considerarse como una solución para este tipo de problemas.

La maldición de la dimensionalidad ha atraído la atención de muchos investigadores durante los últimos años en el análisis de la eficiencia no paramétrica, ya que en numerosas contribuciones empíricas se observa en los resultados una falta de diferenciación entre las unidades evaluadas. Por lo general, se debe a un número excesivo de variables con respecto al número total de observaciones. En la literatura, algunos autores han optado por aplicar el análisis de componentes principales (PCA) de diferentes maneras para mejorar el poder de discriminación en DEA (véase, por ejemplo, Ueda y Hoshiai, 1997; Adler y Berechman, 2001; Adler y Golany, 2002; Araujo et al., 2014). Una posibilidad es reducir el número de inputs y outputs en los factores mediante PCA. Si la mayor parte de la varianza de los datos puede atribuirse a los primeros factores, entonces los inputs y outputs originales en DEA pueden ser reemplazadas por estos factores sin mucha pérdida de información. Otra posibilidad es utilizar PCA como medio de ponderar las variables de entrada y salida y agregarlas. Otros investigadores prefieren el uso de técnicas asociadas con la reducción directa de las variables en lugar de la

agregación de la información por PCA. Probablemente, el primero en esta línea fue Pastor et al. (2002), centrado en analizar el papel marginal de una determinada variable "candidata", con respecto a la eficiencia medida por medio de un modelo DEA. Estos autores desarrollaron un test estadístico específico que permite evaluar la importancia de la contribución de la eficiencia observada del candidato. En última instancia, esta técnica puede proporcionar ciertas señales para decidir la incorporación o la eliminación de una variable del modelo DEA correspondiente. Otras contribuciones relacionadas son Ruggiero (2005) y Nataraja y Johnson (2011). Más recientemente, Charles et al. (2019) han proporcionado un enfoque sencillo para aumentar el poder de discriminación entre las DMUs eficientes e ineficientes utilizando el conocido modelo puro de DEA, que considera sólo los inputs o sólo los outputs. Véase también Shen et al. (2016). Además, se han introducido en la literatura algunas reglas empíricas relativas al número de DMUs frente al número de inputs y outputs, que hoy en día son bien conocidas. Por ejemplo, Golany y Roll (1989) y Homburg (2001) sugieren que el número de DMUs debería ser al menos el doble del número de inputs y outputs. Nunamaker (1985), Banker et al. (1989), Friedman y Sinuany-Stern (1998) y Raab y Lichty (2002) sugieren que el número de DMUs debería ser al menos tres veces el número de inputs y outputs; y Dyson et al. (2001) sugieren que el número de DMUs debería ser al menos el doble del producto del número de inputs y el número de outputs. Además, otra regla empírica que puede servir de orientación es, en consonancia con Cooper et al. (2007), $n \geq \max\left\{m \cdot s, 3 \cdot (m+s)\right\}$. Las reglas empíricas 2-5 de la Sección 4.2.1. se definieron a partir de estas últimas relaciones entre el tamaño de la muestra y el número de inputs y outputs.

En esta sección, se ilustra la utilidad del enfoque RF+EAT para hacer frente a la maldición de la dimensionalidad mediante tres bases de datos tomadas de la literatura. En particular, se utilizan los datos explotados en Sarkis (2000), Jenkins y Anderson (2003) y Homburg (2001).

De los resultados (Tabla 7, Tabla 8 y Tabla 9), se puede concluir que el poder de discriminación de EAT es mayor que el poder de discriminación de FDH pero, al mismo tiempo, el poder de discriminación de la nueva técnica, RF+EAT, es claramente mejor que el poder de discriminación de EAT. Se puede ver un caso extremo en la Tabla 7, donde FDH identifica todas las unidades como técnicamente eficientes (la puntuación es igual a uno). Sólo cuatro de las veintidós unidades evaluadas se muestran como ineficientes por EAT, mientras que en el caso de la técnica RF+EAT, hasta doce DMUs se identifican como técnicamente ineficientes (una puntuación superior a uno), dependiendo de la regla utilizada para determinar el parámetro *mtry* en el Algoritmo 2. Además, los resultados obtenidos al aplicar las cinco reglas para seleccionar aleatoriamente un subconjunto de variables de entrada muestran, en general, patrones similares. Por último, cabe mencionar que, en el caso del enfoque RF+EAT, la puntuación de eficiencia puede tomar valores inferiores a uno, algo que no puede suceder con FDH y EAT. Este resultado se debe a que la frontera de producción estimada por Random Forest no envuelve por arriba necesariamente todas las observaciones. El proceso de Bagging que forma parte de Random Forest implica que muchos modelos individuales (árboles) se ajustan utilizando diferentes muestras de datos, seleccionadas al azar del conjunto de datos original. Por consiguiente, cada DMU sólo aparece en un cierto número de estas muestras aleatorias (el muestreo con sustitución en promedio hace que se excluya alrededor del 37% de las unidades de una muestra determinada) y las fronteras de producción individuales estimadas no siempre pueden envolver todas las unidades. Por lo tanto, el proceso final de agregación, que se aplica en RF+EAT, podría dar una puntuación inferior a uno, especialmente en el caso de las unidades que muestran un patrón relacionado con un buen rendimiento. En el caso de estas unidades, cuando la puntuación de eficiencia de RF+EAT toma un valor inferior a uno, la puntuación debe interpretarse como una puntuación de supereficiencia en la línea de la literatura anterior, como en Andersen y Petersen (1993). FDH y DEA evalúa la eficiencia relativa de las DMUs pero no permite una clasificación de las unidades eficientes. Andersen y Petersen (1993) fueron los primeros en introducir una versión modificada de DEA basada en la comparación de las unidades eficientes en relación con una tecnología de referencia construida por todas las demás unidades, es decir, sin considerar la unidad objeto de

evaluación. RF+EAT funciona en esta misma línea, permitiendo detectar unidades supereficientes.

Tabla 7. Resultados usando los datos de Sarkis (2000).

DMU	X1	X2	X3	X4	X5	Y1	Y2	Y3	FDH	EAT	RF+EAT Breiman	RF+EAT DEA1	RF+EAT DEA2	RF+EAT DEA3	RF+EAT DEA4
1	656	552678100	609	1190	670	5	14	13900	1	1	1.504	1.271	1.274	1.267	1.294
2	786	539113200	575	1190	682	4	18	23600	1	1	1.170	0.989	1.016	0.986	1.007
3	912	480565400	670	1222	594	4	24	39767	1	1	0.938	0.968	0.967	0.962	0.965
4	589	559780715	411	1191	443	9	10	13900	1	1	0.991	1	1	1	1
5	706	532286214	325	1191	404	7	14	23600	1	1	1.263	1	1.010	1	1
6	834	470613514	500	1226	384	6.5	18	40667	1	1.027	1.051	1.054	1.085	1.059	1.058
7	580	560987877	398	1191	420	9	10	13900	1	1	1	1	1	1	1
8	682	532224858	314	1191	393	7	14	23600	1	1	1.263	1	0.984	1	1
9	838	466586058	501	1229	373	6.5	22	41747	1	1	0.991	1.056	1.053	1.047	1.053
10	579	561555877	373	1191	405	9	9	13900	1	1	1	1	1	1	1
11	688	532302258	292	1191	370	7	13	23600	1	1	1.263	1	0.984	1	1
12	838	465356158	499	1230	361	6.5	17	42467	1	1.064	1.006	1.056	1.049	1.044	1.032
13	595	560500215	500	1191	538	9	12	13900	1	1	1	1	1	1	1
14	709	532974014	402	1191	489	7	17	23600	1	1	1.148	1	1.037	1	1
15	849	474137314	648	1226	538	6.5	20	40667	1	1.027	1.099	1.103	1.106	1.090	1.089
16	604	560500215	500	1191	538	9	12	13900	1	1	1	1	1	1	1
17	736	532974014	402	1191	489	7	17	23600	1	1	1.148	1	1.037	1	1
18	871	474137314	648	1226	538	6.5	20	40667	1	1.027	1.099	1.103	1.106	1.100	1.089

19	579	568674539	495	1193	558	9	7	13900	1	1	1	1	1	1	1
20	695	536936873	424	1195	535	6	18	23600	1	1	1.106	0.989	0.998	0.986	1
21	827	457184239	651	1237	513	7	16	45167	1	1	0.973	0.828	0.866	0.790	0.872
22	982	457206173	651	1239	513	7	16	45167	1	1	0.982	0.993	0.996	0.991	0.990

Tabla 8. Resultados usando los datos de Jenkins y Anderson (2003).

DMU	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**Y1**	**Y2**	**FDH**	**EAT**	**RF+EAT Breiman**	**RF+EAT DEA1**	**RF+EAT DEA2**	**RF+EAT DEA3**	**RF+EAT DEA4**
1	15	27	70	23	18	33	85	82	1.129	1.129	1.108	1.112	1.126	1.103	1.123
2	5	2	70	15	11	5	96	93	1	1	0.978	0.982	0.989	0.975	0.990
3	25	26	75	22	24	32	78	87	1.069	1.069	1.066	1.068	1.073	1.062	1.075
4	18	15	75	18	16	23	87	88	1.057	1.057	1.052	1.054	1.058	1.049	1.062
5	9	4	80	5	14	26	89	94	1	1	0.995	0.996	0.994	0.995	0.995
6	6	2	80	13	9	28	93	93	1	1.011	1.004	1.006	1.005	1.006	1.003
7	14	6	85	14	13	21	92	91	1	1.033	1.026	1.028	1.025	1.028	1.026
8	17	17	90	3	17	18	97	92	1	1	0.988	0.994	0.993	0.984	0.988

Tabla 9. Resultados usando los datos de Homburg (2001).

DMU	X1	X2	X3	X4	X5	X6	Y1	Y2	FDH	EAT	RF+EAT Breiman	RF+EAT DEA1	RF+EAT DEA2	RF+EAT DEA3	RF+EAT DEA4
1	300	75	60	42	45	75	500	2.5	1	1.040	1.015	1.012	1.017	1.019	1.019
2	250	66	40	30	30	73	500	2.3	1	1	1.001	1	1.005	0.997	1.003
3	310	80	65	41	47	85	510	2.1	1	1.020	1.011	1.004	1.008	1.008	1.005
4	270	77	54	48	40	80	490	2.7	1	1	0.959	0.970	0.973	0.971	0.969
5	239	74	62	41	54	79	480	2	1	1.042	1.045	1.046	1.058	1.035	1.060
6	280	73	57	47	60	77	490	2.5	1	1.020	1.019	1.035	1.040	1.024	1.046
7	340	76	70	45	50	84	470	1.9	1.106	1.106	1.102	1.092	1.096	1.097	1.097
8	290	72	55	43	39	80	520	2.5	1	1	0.993	0.985	0.987	0.990	0.989

4.3. EFICIENCIA COMPUTACIONAL

En esta sección de la tesis, en comparación con la literatura descrita hasta ahora sobre las fronteras de producción, la principal contribución es la mejora de la precisión del estimador de la función de producción generada a partir de EAT, recurriendo al *backtracking* y a técnicas heurísticas, al tiempo que se reduce la carga computacional del algoritmo. Tal y como se ha descrito anteriormente, el algoritmo utilizado por EAT se basa en una técnica heurística para seleccionar el siguiente nodo que se va a dividir durante el proceso de crecimiento del árbol de decisión correspondiente. Sin embargo, como se mostrará en este capítulo de tesis, esta heurística no siempre produce el mínimo error cuadrático medio entre todos los árboles posibles que podrían desarrollarse. Por consiguiente, uno de los principales objetivos de este trabajo es mejorar la precisión del estimador de la función de producción generada a partir de EAT recurriendo a técnicas de *backtracking* (Baase, 2009 y Horowitz y Sahni, 1978). En particular, se combina la idea que subyace en el enfoque heurístico con la potencialidad del *backtracking* (Pearl, 1984 y Tarjan, 1972) para mejorar la calidad del estimador de la función de producción basada en EAT. Además, a través de nuestro enfoque, se disminuye la carga computacional de las técnicas estándar de *backtracking* aplicadas a la metodología del árbol de decisión basado en EAT, como se muestra en las experiencias simuladas. Hasta donde sabemos, este trabajo es el primero que contempla los aspectos computacionales de la técnica de Efficiency Analysis Trees (Esteve et al., 2020). Por un lado, hay algunos trabajos anteriores dedicados a la medición de la eficiencia y relacionados con las características computacionales del problema DEA (véase, por ejemplo, Dulà y Thrall, 2001 y Dulà, 2008). Sin embargo, explotan características vinculadas a la estructura del problema como un modelo de Programación Matemática. Por consiguiente, estas contribuciones están muy alejadas de la estructura de árbol que se estudia en este capítulo. Por otra parte, hay otros trabajos anteriores que se centran en las técnicas de aprendizaje automático, como los árboles de decisión, y la eficiencia. A pesar de ello, estas contribuciones no combinan realmente estas dos facetas, sino que aplican cada una en una etapa diferente: en una primera etapa se determina la eficiencia técnica y, en una segunda etapa, se utiliza la eficiencia como variable respuesta con los árboles de decisión

o Random Forest, para dar detalles de los factores relacionados con la eficiencia técnica (ver, por ejemplo, Emrouznejad y Anouze, 2010 y Rebai, et al., 2019). Por el contrario, EAT, conecta metodológicamente estos dos temas clave, es decir, los árboles de decisión y la medición de la eficiencia, tal y como se ha podido comprobar a lo largo del desarrollo de la presente tesis.

Hasta este momento, el desarrollo de la tesis se ha abordado desde un punto de vista exclusivamente metodológico y no desde uno computacional. EAT es un algoritmo voraz cuya heurística de división de un nodo no obtiene el árbol con el menor error cuadrático medio (MSE). Existe, por tanto, la posibilidad de mejorarlo aplicando la técnica algorítmica bien conocida de *backtracking*. Sin embargo, esta técnica computacionalmente es muy costosa de aplicar puesto que calcula todas las posibles combinaciones de división de todos los nodos de un árbol EAT. En este sentido, en esta sección, se proponen dos algoritmos adicionales al EAT heurístico (el estándar) y al EAT con *backtracking* "puro", que combinan ambos enfoques, es decir, la heurística y el *backtracking* con el propósito de alcanzar precisiones cercanas a EAT con *backtracking* en tiempos computaciones (en segundos) más próximos a EAT heurístico. Se ha encontrado que EAT heurístico es más rápido con un MSE peor que EAT con *backtracking* que obtiene mejor MSE, pero en un tiempo excesivo. La implementación de estos cuatro algoritmos se ha realizado en el lenguaje de programación C debido a que, junto a Python y R, es uno de los lenguajes de programación que se usa en el campo del aprendizaje automático. Además, a diferencia de Python y R, el lenguaje C, sin lugar a dudas, es el más rápido, aunque, del mismo modo, es el más complejo de utilizar. Hasta ahora, todos los experimentos computacionales que se han presentado en esta tesis se han realizado a través de los algoritmos implementados en Python, pero, alcanzado este capítulo en el que la eficiencia computacional toma gran relevancia, todos los experimentos que se muestran se han implementado en el lenguaje de programación C. Los algoritmos implementados se encuentran disponibles como software libre bajo la licencia GNU General Public License version 3 en el repositorio GitHub https://github.com/MiriamEsteve/EATpy, en el caso del lenguaje de programación

Python, y en https://github.com/MiriamEsteve/BackEATC, en el caso de los algoritmos que se presentarán en este capítulo en el lenguaje C. La siguiente subcción se enfocará exclusivamente en el código implementado en el lenguaje de programación R, debido a que se ha desarrollado un paquete de R bajo la Licencia Pública General de GNU versión 3 que se puede descargar desde Comprehensive R Archive Network (CRAN) en https://cran.r-project.org/web/packages/eat. Este paquete incorpora las principales funcionalidades de EAT, Random Forest para EAT, DEA y FDH.

El Capítulo 4 Sección 4.3. se divide en dos secciones. En la primera parte se abordan los cuatro algoritmos propuestos y en la segunda parte se muestran los resultados a partir de simulaciones.

4.3.1. ESQUEMAS DE LOS ALGORITMOS DESARROLLADOS PARA ESTIMAR LAS FUNCIONES DE PRODUCCIÓN

En esta sección, se analiza el rendimiento de cuatro algoritmos vinculados a la metodología EAT con respecto a cuestiones de computación y precisión.

4.3.1.1. ALGORITMO BACKTRACKING BASADO EN EL MÉTODO EAT

En este trabajo se propone un algoritmo *backtracking* para obtener el árbol que da los valores más pequeños de la suma del MSE de las diferentes combinaciones de (x_j, S_j).

El árbol heurístico, vinculado al algoritmo estándar de EAT, sólo explora una solución factible mientras que *backtracking* explora todas ellas, obteniendo la mejor.

El desarrollo del algoritmo *backtracking* en el presente trabajo considera la estructura de descomposición del árbol en la función de división, obteniendo esta estructura de árbol según el principio de monotonía (o *free disposability*, en inglés). El árbol se desarrolla de forma similar a la subsección anterior, aunque la profundidad del árbol viene marcada por

el número de nodos que pueden ser creados mediante la partición recursiva hasta alcanzar un nodo hoja.

4.3.1.2. COMBINACIÓN DE LAS TÉCNICAS DE LA HEURÍSTICA Y DEL BACKTRACKING

El algoritmo heurístico propuesto como método estándar en EAT obtiene valores adecuados para el MSE. Sin embargo, estos valores son peores que los obtenidos por el algoritmo *backtracking* propuesto en la Sección 4.3.1.1. a expensas de utilizar un alto coste de cálculo. Por lo tanto, en esta subsección se proponen nuevos algoritmos con ambos esquemas para obtener valores adecuados de MSE en un tiempo razonable. De hecho, en general, el coste computacional consumido por el algoritmo *backtracking* es inviable. En este sentido, nuestras propuestas combinan ambos enfoques para obtener mejores resultados que utilizando ambos por separado.

Se estudian dos enfoques. El primero combina en primer lugar la heurística y en segundo lugar el *backtracking*. El árbol se construye mediante la heurística hasta que se alcanza una condición (denotada por *numStopH*). Cuando esta parte del árbol está terminada, el *backtracking* continúa construyendo el árbol hasta que se alcanza una condición final (denotada por *numStopB*). En ese caso, *numStopH* debe ser mayor que *numStopB*, de lo contrario el árbol se completará, y el *backtracking* no podrá continuar su expansión. El cumplimiento de estas condiciones para *numStopH* y *numStopB*, están determinadas por la matriz de datos asociada al nodo. Así, cuando se alcanzan las condiciones finales en el proceso de división, los nodos obtenidos se marcan como nodos hojas. Por esta razón, cuando se cumple la primera de ellas, es decir, *numStopH*, es necesario redefinir los nodos que son hojas y, luego, el estado del árbol le permite seguir expandiendo los nodos.

El segundo enfoque es similar al anterior, pero implica cambiar el orden de los algoritmos, es decir, en primer lugar se emplea el del *backtracking* y en segundo lugar el del

heurístico. En este caso, el árbol comienza a construirse a través del *backtracking* hasta que se alcanza el *numStopB* y continúa expandiéndose a través de la heurística hasta que se satisface la condición vinculada al *numStopH*.

La Figura 20 muestra los algoritmos propuestos: Algoritmo A (la heurística), Algoritmo B (el *backtracking*), Algoritmo C (primero la heurística y luego el *backtracking*) y, finalmente, Algoritmo D (primero el *backtracking* y luego la heurística). Los nodos con forma de círculo significan que han sido construidos siguiendo el enfoque heurístico. Los nodos triangulados son todos los posibles nodos que pueden ser construidos. Los parámetros *numStopH* y *numStopB* indican la condición que deben cumplir los nodos, tanto los representados por círculos como los representados por triángulos, para completar la construcción del árbol mediante el enfoque heurístico o de *backtracking*, respectivamente.

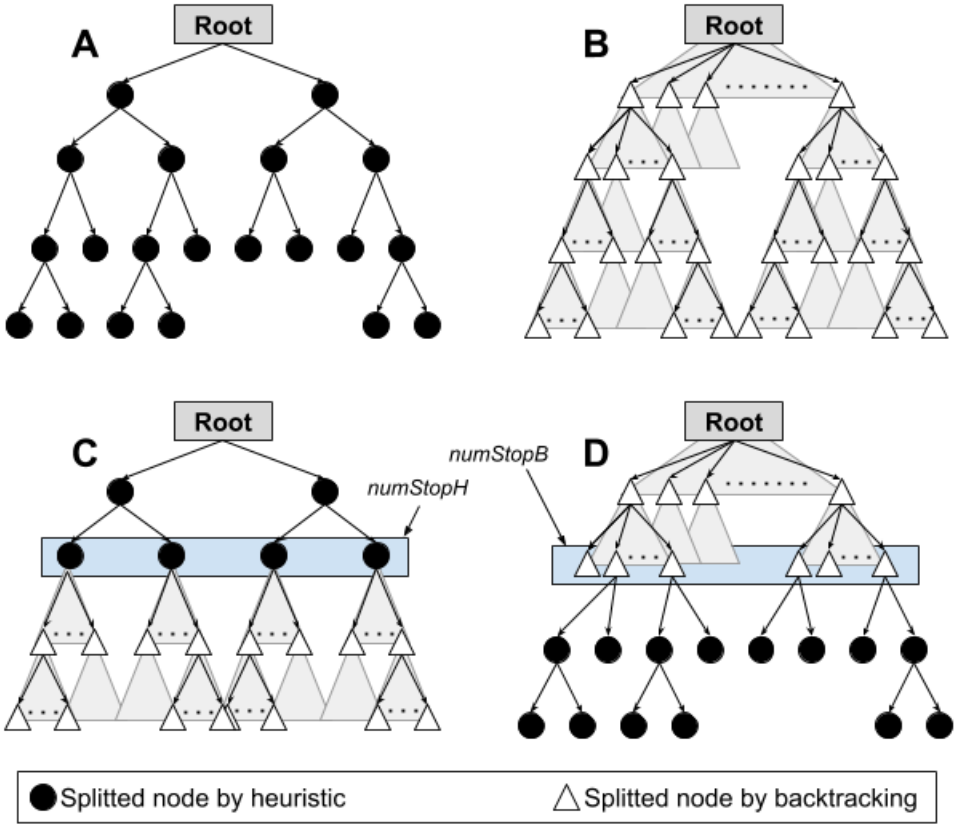


Figura 20. Desarrollo de las estructuras de árboles de los diferentes enfoques.

4.3.2. SIMULACIONES

Los experimentos fueron llevados a cabo en un ordenador con Intel(R) Core (TM) i7-10510U con CPU 1.80 GHz y RAM 16 GB. El código fue implementado en C (disponible como software libre bajo la licencia GNU General Public License version 3 en el repositorio GitHub https://github.com/MiriamEsteve/BackEATC), tal y como se ha explicado al principio de este capítulo. Se realizaron algunos experimentos para afinar ciertos parámetros del algoritmo. En todos ellos, los datos se simularon a partir de una función de producción teórica típica de Cobb-Douglas en microeconomía. Los datos de las entradas fueron generados al azar siguiendo una distribución $Uni(0,1)$ independiente de cada input y observación. Entonces, el nivel de output "eficiente", es decir, la frontera de producción teórica, se calcula como $f(x_1,...,x_n) = x_1^{\alpha_1} + ... + x_n^{\alpha_n}$, siendo $\sum_{i=1}^{n} \alpha_i = 0.5$. Por último, el término de "ineficiencia" se determina aleatoriamente como $u \sim Exp\left(\frac{1}{3}\right)$.

En este trabajo, los experimentos se han llevado a cabo utilizando los valores mostrados en la Tabla 10.

Las simulaciones se realizaron en 100 experimentos para cada combinación del tamaño del conjunto de datos y el número de inputs. El segundo experimento estudia el coste de la ejecución al variar el tamaño del conjunto de datos y el número de inputs. En estos experimentos, *numStopH* y *numStopB* toman el valor 2.

Tabla 10. Funciones de producción teóricas utilizadas en los experimentos en función del número de inputs.

#Inputs	**Forma functional** $f(x)$

2	$f(x) = x_1^{0.1} + x_2^{0.4} + e^{-u}$
3	$f(x) = x_1^{0.1} + x_2^{0.1} + x_3^{0.4} + e^{-u}$
4	$f(x) = x_1^{0.1} + x_2^{0.1} + x_3^{0.2} + x_4^{0.2} + e^{-u}$
5	$f(x) = x_1^{0.05} + x_2^{0.1} + x_3^{0.05} + x_4^{0.1} + x_5^{0.2} + e^{-u}$
6	$f(x) = x_1^{0.05} + x_2^{0.1} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.15} + e^{-u}$
7	$f(x) = x_1^{0.05} + x_2^{0.05} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.05} + x_7^{0.15} + e^{-u}$
8	$f(x) = x_1^{0.05} + x_2^{0.05} + x_3^{0.05} + x_4^{0.05} + x_5^{0.1} + x_6^{0.05} + x_7^{0.05} + x_8^{0.1} + e^{-u}$

En la Tabla 11 se indica el tiempo medio de ejecución (en segundos) y la media de la suma de los MSE de los nodos hoja en los diferentes experimentos considerados. En particular, las dos primeras columnas indican el tamaño de la muestra y el número de inputs. El resto de columnas muestran el tiempo de ejecución y la media de la suma del MSE en los nodos hojas obtenida en los cuatro algoritmos propuestos.

En cuanto a los resultados, todos los métodos se ven afectados por el aumento de la complejidad del problema asociado al número de filas en los conjuntos de datos, ya que detectamos que cuanto mayor es el tamaño de la muestra, mayor es el valor de MSE. Además, el aumento de la complejidad relacionado con el número de inputs en los conjuntos de datos afecta a todos los métodos, ya que observamos que MSE disminuye. El tiempo necesario para ejecutar los experimentos sólo se ve afectado por el aumento del número de filas.

Además, el Algoritmo A presenta el valor más alto para MSE, el Algoritmo B el más bajo y los otros tienen valores de MSE entre estos dos. Por un lado, el Algoritmo C, que tiene un coste de ejecución mayor que el del Algoritmo A, tiene un valor de MSE que se

aproxima al correspondiente del Algoritmo A. Por otro lado, el Algoritmo D, que tiene un coste de ejecución menor que el del Algoritmo B, tiene un valor de MSE que se aproxima al del Algoritmo B. Además, en la Tabla 11, parece que el Algoritmo D es el mejor algoritmo para obtener un MSE atractivo en un tiempo de ejecución aceptable.

Tabla 11. Coste de ejecución y error de precisión cuando los cuatro algoritmos se utilizan para varios problemas y tamaño de inputs

Núm. Obs.	#Inputs	Algoritmo A		Algoritmo B		Algoritmo C		Algoritmo D	
		Tiempo	MSE	Tiempo	MSE	Tiempo	MSE	Tiempo	MSE
10	2	0.000	0.26	309.204	0.07	32.524	0.26	35.789	0.08
15	2	0.001	1.96	318.757	0.95	17.664	1.62	79.800	1.06
20	2	0.002	4.59	330.005	4.04	19.544	4.33	157.196	4.04
10	3	0.001	0.15	82.775	0.05	6.585	0.15	48.600	0.13
15	3	0.001	1.25	131.355	0.73	10.309	1.25	84.235	1.07
20	3	0.002	0.50	183.638	0.45	17.700	0.50	163.225	0.49
10	4	0.001	0.20	91.175	0.07	10.531	0.21	42.593	0.18
15	4	0.001	0.12	276.829	0.07	12.563	0.12	88.602	0.10
20	4	0.002	0.74	176.645	0.63	18.283	0.74	180.085	0.69
10	5	0.001	0.91	62.846	0.33	9.027	0.91	41.844	0.77
15	5	0.002	0.19	100.198	0.11	16.003	0.19	106.907	0.16
20	5	0.002	0.62	141.652	0.58	18.689	0.62	185.002	0.56
10	6	0.001	0.24	58.681	0.09	8.346	0.23	39.893	0.20
15	6	0.001	0.76	90.44	0.44	12.02	0.76	93.025	0.64
20	6	0.002	0.19	118.405	0.18	16.827	0.19	194.903	0.18

10	7	0.001	0.45	61.117	0.17	8.642	0.45	40.443	0.38
15	7	0.001	0.34	91.154	0.20	12.987	0.33	92.246	0.28
20	7	0.003	0.11	124.467	0.10	17.445	0.11	187.113	0.10
10	8	0.001	0.01	65.492	0.01	9.028	0.01	40.777	0.01
15	8	0.002	0.01	114.744	0.01	16.823	0.01	99.984	0.01
20	8	0.002	0.21	126.868	0.20	19.734	0.21	191.372	0.19

El algoritmo D es el que obtiene los valores de MSE más cercanos a los observados para el Algoritmo B, que presenta los mínimos, en un tiempo de ejecución factible.

Se ejecutaron nuevos experimentos usando el Algoritmo D y variando la primera condición final, es decir, *numStopB*. La Tabla 12 muestra resultados similares a los de la Tabla 11, pero sólo con el Algoritmo D. Las dos primeras columnas son el tamaño del problema y el tamaño del input, y los bloques de columnas restantes muestran el valor aplicado de *numStopB*. Además, cada uno de estos bloques muestra tres columnas con el tiempo de ejecución, el valor de MSE y el porcentaje de mejora con respecto al Algoritmo B.

Al igual que en la Tabla 11, los valores de MSE crecen cuando el tamaño del problema aumenta y estos resultados están notoriamente influenciados por el crecimiento de la primera condición final. Este efecto puede observarse en la columna "*%B*", que representa la mejora del algoritmo D con respecto al Algoritmo B. El porcentaje de mejora aumenta del 30% al 94%. El coste de ejecución también se ve afectado por el incremento del *numStopB*, que se acerca cada vez más a los tiempos de ejecución obtenidos en el Algoritmo A. Así pues, el mejor *numStopB* es el que equilibra por igual el trabajo de *backtracking* y la heurística. De esta manera, se puede obtener un resultado aceptable en un tiempo de ejecución factible.

Tabla 12. Coste de ejecución y error de precisión del Algoritmo D cuando la primera condición final varía en 4, 6, 8 y 9.

Núm. Obs.	#Inputs	*numStopB* = 4			*numStopB* = 6			*numStopB* = 8			*numStopB* = 9		
		Tiempo	**MSE**	**%B**	**Tiempo**	**MSE**	**%B**	**Tiempo**	**MSE**	**%B**	**Tiempo**	**MSE**	**%B**
10	2	24.434	0.08	94.6	13.115	0.08	92.2	5.390	0.20	33.9	4.002	0.20	31.3
15	2	57.733	1.26	68.7	35.471	1.27	68.3	9.183	1.77	18.9	5.374	1.85	10.2
20	2	124.601	1.41	31.8	73.625	1.44	27.4	26.403	1.43	28.9	9.2643	1.44	27.1
10	3	23.855	0.14	5.2	15.813	0.15	1.5	6.209	0.15	0.8	4.328	0.15	0.4
15	3	68.680	1.07	35.7	46.821	1.07	35.2	8.381	1.19	11.7	5.7634	1.21	9.1
20	3	134.692	0.48	43.0	104.128	0.50	11.7	24.974	0.50	5.2	9.634	0.50	1.2
10	4	26.775	0.19	15.5	15.810	0.19	14.6	6.154	0.20	9.7	4.836	0.20	7.6
15	4	72.738	0.11	23.2	65.130	0.11	21.0	15.746	0.12	3.4	5.897	0.12	0.7
20	4	139.782	0.72	20.5	105.389	0.72	18.7	20.193	0.72	12.3	10.006	0.74	0.8

10	5	27.374	0.78	22.9	17.479	0.79	21.3	10.842	0.83	13.3	4.982	0.90	2.5
15	5	81.548	0.16	34.2	50.558	0.16	31.5	12.384	0.16	30.3	6.008	0.16	30.7
20	5	149.646	0.59	56.1	114.125	0.59	50.1	19.193	0.60	32.3	10.235	0.61	11.4
10	6	31.683	0.20	21.9	16.266	0.20	20.5	13.573	0.22	4.2	5.002	0.23	0.7
15	6	68.658	0.63	40.3	51.365	0.63	39.4	28.197	0.72	12.7	6.735	0.73	7.4
20	6	139.837	0.18	61.6	112.087	0.19	25.2	32.645	0.19	25.2	10.935	0.19	22.8
10	7	27.936	0.38	23.6	18.906	0.40	16.4	7.203	0.42	8.9	5.009	0.42	8.2
15	7	73.134	0.29	37.8	58.651	0.29	33.5	17.342	0.31	18.0	6.892	0.31	17.2
20	7	142.477	0.10	48.7	127.912	0.10	21.6	23.201	0.10	21.6	11.281	0.10	16.5
10	8	28.105	0.00	10.0	18.513	0.00	2.0	10.240	0.01	0	5.321	0.01	0
15	8	82.828	0.01	30.2	61.398	0.01	22.6	23.420	0.01	5.3	6.746	0.01	7.5
20	8	143.567	0.20	55.4	122.804	0.20	36.9	27.293	0.21	13.8	10.386	0.21	9.2

La Figura 21 muestra el rendimiento del Algoritmo D para diferentes problemas de *numStopB* cuando el tamaño del problema cambia. El eje X describe el tamaño de la muestra en relación con el número de las variables de entradas (n/tamaño de las variables de entrada) y el eje Y muestra el porcentaje de mejora de MSE con respecto al algoritmo de *backtracking* (Algoritmo B). La Figura 21 revela que el porcentaje de mejora es mayor cuando *numStopB* es igual a 4 y 6.

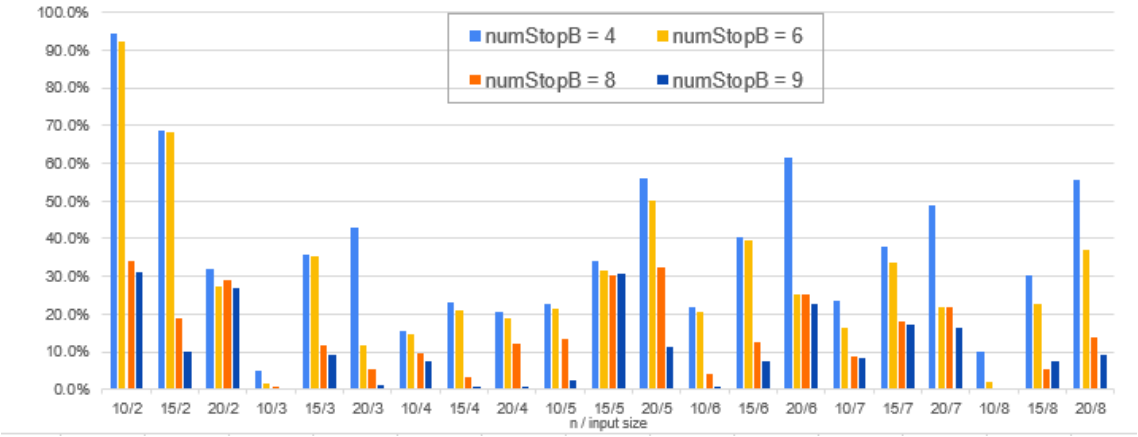


Figura 21. Porcentaje de mejora en la precisión entre el Algoritmo D y el Algoritmo B cuando el tamaño del problema, el número de inputs y *numStop* se modifica.

En el caso de las aplicaciones reales, nuestros resultados podrían ser útiles para determinar ciertos parámetros, como el *numStopB*. En particular, la forma de determinar el valor más adecuado del parámetro *numStopB* para aplicar el Algoritmo D depende directamente del coste de cálculo y, por lo tanto, del tiempo de cálculo disponible del usuario. Por ejemplo, siguiendo nuestros resultados, en un problema con un tamaño de muestra de 10 y un tamaño de variables de entrada de 5; si el usuario tiene un tiempo aproximado de 5 segundos, el *numStopB* más adecuado es 9. Si, por el contrario, el usuario tiene hasta 30 segundos disponibles, entonces el valor adecuado para el *numStopB* sería 4, obteniendo el menor MSE.

4.4. PAQUETE DE R (EAT)

En esta sección de la tesis, se presenta el paquete de R (CRAN), bautizado como *eat*, que incorpora las principales funcionalidades de Efficiency Analysis Trees (EAT), Random Forest para Efficiency Analysis Trees (RF+EAT), Data Envelopment Analysis (DEA) y Free Disposal Hull (FDH). El paquete *eat* está disponible como software libre, bajo la Licencia Pública General GNU versión 3, y puede descargarse de la Comprehensive R Archive Network (CRAN) en https://CRAN.R-project.org/package=eat, incluyendo material suplementario como conjuntos de datos o viñetas para replicar todos los resultados presentados en este trabajo. Además, *eat* está alojado en un repositorio de código abierto en GitHub en https://github.com/MiriamEsteve/EAT.

Hoy en día se pueden encontrar diversos paquetes de R para estimar la eficiencia técnica, como por ejemplo, *Benchmarking* (Bogetoft y Otto, 2010), para estimar tecnologías y medir la eficiencia técnica usando DEA y *Stochastic Frontier Analysis* (SFA); *nonparaeff* (Oh y Suh, 2013), para medir la eficiencia y la productividad usando DEA y sus variaciones; *npbr* (Daouia, Laurent y Noh, 2017), que abarca las técnicas de envoltura de datos basadas en polinomios a trozos (o *splines*, en inglés), ajuste lineal local, valores extremos y suavizado de núcleos; *snfa* (McKenzie, 2018), que se ajusta tanto al suavizado del DEA como al no paramétrico de SFA; o *semsfa* (Ferrara y Vidoli, 2018), que, en una primera etapa, estima los modelos de frontera estocástica mediante técnicas de regresión semiparamétricas o no paramétricas para relajar las restricciones paramétricas y, en una segunda etapa, aplica una técnica basada en la pseudoverosimilitud o el método de los momentos para estimar los parámetros de varianza. Además, existen otros paquetes sobre medición de la eficiencia desarrollados para plataformas alternativas. En MATLAB (The MathWorks Inc., 2021), se puede encontrar el Data Envelopment Analysis Toolbox (Álvarez, Barbero y Zofío, 2020), que implementa los principales modelos DEA y resuelve medidas como la función de distancia direccional (con outputs deseables e indeseables), el modelo aditivo ponderado y el índice de Malmquist-Luenberger; o el Total Factor Productivity (TFP) Toolbox para MATLAB (Balk, Barbero y Zofío, 2018),

que incluye funciones para calcular los principales índices de Productividad Total de Factores y su descomposición mediante modelos DEA. En Stata (StataCorp, 2021), es posible encontrar un paquete similar en Ji y Lee (2010).

Sin embargo, ninguna de estas herramientas tiene el objetivo principal de estimar una frontera de producción mediante árboles de regresión satisfaciendo los principios microeconómicos de libre disponibilidad (o *free disposability*, en inglés), convexidad y datos deterministas. En cuanto a las medidas de eficiencia técnica implementadas en el nuevo paquete *eat*, cabe mencionar que se proporcionan puntuaciones numéricas y gráficos de barras para los modelos radiales BCC orientados al output y al input (Banker, Charnes y Cooper, 1984), la función de distancia direccional (Chambers, Chung y Färe, 1998), el modelo aditivo ponderado (Lovell y Pastor, 1995; y Charnes, Cooper, Golany, Seiford y Stutz, 1985) y las medidas Russell orientadas al output y al input (Färe y Lovell, 1978). Además, la adaptación de Random Forest (Breiman, 2001) para tratar conjuntos de árboles de análisis de eficiencia RF+EAT también se ha incorporado al nuevo paquete *eat*. Por otra parte, los modelos estándar FDH y DEA han sido incluidos en el paquete para facilitar la comparación con las puntuaciones de eficiencia determinadas por la técnica EAT. Asimismo, la convexificación de la estimación de la tecnología proporcionada por EAT, Convexified Efficiency Analysis Trees (CEAT) (Aparicio et. al, 2021), se implementa en el paquete *eat*.

4.4.1. ESTRUCTURA DE LOS DATOS

Los datos son manejados como un tipo de dato usual en R, denominado *Data Frame.* Todas las funciones devuelven una estructura objeto que contiene los resultados de la estimación y los argumentos de la función de estimación.

Algunos de los campos de las estructuras de las principales funciones del paquete son los siguientes[1]:

[1] Para obtener una lista completa es necesario escribir *"help"* más el nombre de la función en R.

- *data*: Contiene las variables de entrada y de salida.
  - x e *y*: Contienen las variables de entrada y de salida.
  - input_name y output_name: Contienen los índices de entrada y de salida que les corresponde dentro del conjunto de datos.
  - row_names: Nombre de las observaciones.

- Control: Contiene los hiperparámetros seleccionados por el usuario:
  - numStop: Valor mínimo de observaciones que contiene un nodo.
  - fold: Número de trozos en los que se divide el conjunto de datos al aplicar *cross validation*.
  - na.rm: Variable lógica que indica si las variables vacías deben ser ignoradas.
  - max.depth: Hiperparámetro que limita el número de nodos entre el nodo raíz y el nodo hoja más lejano (no se incluye la raíz).
  - max.leaves: Hiperparámetro que determina el número máximo de nodos hoja.

- tree: Lista que contiene los nodos de EAT. Está construida por los siguientes elementos:
  - id: Node index.
  - F: Índice del nodo padre.
  - SL: Índice del nodo hijo izquierdo.
  - SR: Índice del nodo hijo derecho.
  - index: Conjunto de índices correspondientes a las observaciones que contiene el nodo.
  - R: Error del nodo.
  - xi: Índice de la variable de entrada por el que se produce la división del nodo.
  - s: Valor numérico de la variable xi.
  - a: Componentes del vector $\boldsymbol{a}^t$.

- `b`: Componentes del vector $\boldsymbol{b}^t$.

- `nodes_df`: Contiene la información siguiente relacionada con todos los nodos que componen la estructura `tree`:
  - `id`: Índice del nodo.
  - `N`: Número de observaciones en el nodo.
  - `Proportion`: Proporción de observaciones en el nodo.
  - `y`: Valores predichos.
  - `R`: Error cuadrático medio (MSE) del nodo.
  - `Index`: Índices de las observaciones del nodo.

- `model`: Contiene la información siguiente relacionada con el modelo entrenado:
  - `nodes`: Número de nodos en el árbol.
  - `leaf_nodes`: Número de nodos hoja en el árbol.
  - `a`: Componentes del vector $\boldsymbol{a}^t$.
  - `y`: La estimación del output de cada nodo hoja.

En concreto, el objeto `RFEAT` contiene alguna información más:

- `forest`: Lista que contiene cada uno de los EAT.
- `Error`: Error Out-Of-Bag del conjunto de árboles.
- `OOB`: Lista que contiene el conjunto de Out-Of-Bag para cada árbol.

4.4.2. CONJUNTO DE DATOS Y FUENTES ESTADÍSTICAS

En este apartado de la tesis se ilustran todos los modelos presentados en el paquete *eat* recurriendo a un único conjunto de datos. Los datos han sido recogidos por la encuesta PISA (Programme for International Student Assessment) 2018. Este conjunto de datos consta de 72 países con tres outputs: dominio de lectura, matemáticas y ciencias; y trece inputs relacionados con el entorno socioeconómico de estos países recogidos del Índice

de Progreso Social (Social Progress Index, 2020). Estos inputs se pueden clasificar en cuatro bloques como se muestra a continuación:

- Campo de necesidades humanas básicas: NBMC (Nutrición y Atención Médica Básica), WS (Agua y Saneamiento), S (Vivienda) y PS (Seguridad Personal).
- Ámbito de los fundamentos del bienestar: ABK (Acceso a los conocimientos básicos), AIC (Acceso a la información y las comunicaciones), HW (Salud y bienestar) y EQ (Calidad ambiental).
- Campo de oportunidades: PR (Derechos Personales), PFC (Libertad y Elección Personal), I (Inclusión) y AAE (Acceso a la Educación Avanzada).
- Campo económico: GDP_PPP (Producto Interior Bruto basado en la Paridad de Poder Adquisitivo).

La Tabla 13 recoge las estadísticas descriptivas de todas estas variables (outputs e inputs):

Tabla 13. Estadísticas descriptivas de las variables incluidas en los datos.

Variable	Tipo	Media	Desv. Estand.	Mín.	Máx.
Reading	Output	450.9	50.5	340.0	549.0
Science	Output	455.1	48.3	336.0	551.0
Mathematics	Output	454.8	52.2	325.0	569.0
NBMC	Input	95.6	3.9	79.2	98.9
WS	Input	94.5	6.6	72.7	99.8
S	Input	93.6	5.5	67.8	98.9
PS	Input	77.0	11.7	53.6	96.7
ABK	Input	90.1	7.9	61.6	99.1
AIC	Input	82.9	9.3	60.4	98.3
HW	Input	74.8	10.9	47.0	90.8
EQ	Input	79.3	11.9	34.9	93.6
PR	Input	81.6	18.0	21.1	98.1
PFC	Input	75.4	11.0	47.2	91.7

I	Input	54.2	17.1	12.4	81.9
AAE	Input	71.7	10.7	48.4	90.4
GDP_PPP	Input	36.0	21.9	7.4	114.1

4.4.3. FUNCIONES DEL PAQUETE

Las funciones incluidas en el paquete *eat* se resumen en la Tabla 14. Esta tabla se compone de tres columnas divididas en cinco subsecciones para la técnica EAT, RF+EAT, FDH, DEA y CEAT. La primera columna es el nombre de la función y la segunda es la descripción de la función.

Tabla 14. Funciones del paquete *eat*.

Función	Descripción
`EAT`	Genera un modelo EAT.
`bestEAT`	Produce el modelo EAT con los mejores valores de los hiperparámetros que minimizan el error calculado en una muestra de prueba de un modelo EAT ajustado con una muestra de entrenamiento.
`efficiencyEAT`	El resultado es un conjunto de datos con las variables de entrada, las puntuaciones de eficiencia obtenidas mediante un modelo EAT y, opcionalmente, las puntuaciones FDH. Las alternativas disponibles son: • "BCC.OUT": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo radial BCC orientado al output. • "BCC.INP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo radial BCC orientado al input.

	• "DDF": Vector numérico con las puntuaciones de eficiencia calculadas mediante la función de distancia direccional. • "RSL.OUT": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo Russell orientado a la producción. • "RSL.INP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo Russell orientado a los inputs. • "WAM.MIP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo aditivo ponderado con Medida de la Proporción de Ineficiencia. • "WAM.RAM": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo aditivo ponderado con la medida de ineficiencia ajustada al rango.
`efficiencyJitter`	Gráfico de fluctuación que representa la dispersión de las puntuaciones de eficiencia de las DMU en un nodo determinado. Se muestran la media y la desviación estándar.
`efficiencyDensity`	Gráfico de densidad para comparar la distribución de las puntuaciones de eficiencia entre dos modelos determinados.
`predictEAT`	Conjunto de datos con los datos originales y los valores predichos por un modelo EAT dado un conjunto de perfiles de entrada.
`rankingEAT`	Conjunto de datos con la importancia de las variables obtenidas por un modelo EAT o una lista con las puntuaciones y un gráfico de barras que representa la importancia de las variables.

`plotEAT`	Gráfico de un modeloEAT mediante una estructura de árbol.
`frontier`	En el caso de un escenario de un input y output, esta función dibuja la función de producción estimada obtenida por EAT. La frontera FDH es opcional.
`RFEAT`	Genera un modelo RF+EAT.
`bestRFEAT`	Produce el modelo RF+EAT con los mejores valores de los hiperparámetros que minimizan el error calculado en una muestra de prueba de un modelo RF+EAT ajustado con una muestra de entrenamiento.
`efficiencyRFEAT`	Conjunto de datos con las variables de entrada y las puntuaciones de eficiencia de RF+EAT y, opcionalmente, las puntuaciones FDH. Sólo está disponible el modelo radial de BCC orientación output.
`predictRFEAT`	Conjunto de datos con los datos originales, los valores predichos por un modelo RF+EAT dado un conjunto de perfiles de entrada.
`rankingRFEAT`	Conjunto de datos con las puntuaciones de la importancia de las variables obtenidas por un modelo RF+EAT o una lista con las puntuaciones y un gráfico de barras que las representa.
`plotRFEAT`	Gráfico con el error Out-of-Bag (OOB) para un RF+EAT formado por *k* árboles.
`efficiencyCEAT`	Produce un conjunto de datos con las variables de entrada, las puntuaciones de eficiencia obtenidas a través de un modelo CEAT y, opcionalmente, las puntuaciones DEA.

	• "BCC.OUT": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo radial BCC orientado al output. • "BCC.INP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo radial BCC orientado al input. • "DDF": Vector numérico con las puntuaciones de eficiencia calculadas mediante la función de distancia direccional. • "RSL.OUT": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo Russell orientado a la producción. • "RSL.INP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo Russell orientado a los inputs. • "WAM.MIP": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo aditivo ponderado con Medida de la Proporción de Ineficiencia. • "WAM.RAM": Vector numérico con las puntuaciones de eficiencia calculadas mediante el modelo aditivo ponderado con la medida de ineficiencia ajustada al rango.
`Y1.sim`	Simula un conjunto de datos en un escenario de un único output. Se pueden generar 1, 3, 6, 9, 12 y 15 inputs.
`X2Y2.sim`	Simula un conjunto de datos en un escenario con 2 inputs y 2 outputs.

4.4.3.1. EL MODELO EAT

El modelo Efficiency Analysis Trees se implementa en R utilizando la función `EAT(data, x, y, …)`. Esta función es la más importante del paquete. Los argumentos mínimos de esta función son los datos (`data`) que contienen las variables de estudio, los índices de las variables predictoras o inputs (`x`) y los índices de las variables predichas o outputs (`y`). Además, se incluyen los argumentos `numStop`, `fold`, `max.depth` y `max.leaves` para los usuarios más experimentados en el campo del aprendizaje automático y los modelos basados en árboles. La modificación de estos cuatro parámetros permite obtener diferentes estimaciones de la frontera y, por tanto, seleccionar la que mejor se adapte a las necesidades del análisis. La descripción de estos parámetros es la siguiente:

- `numStop` se refiere al número mínimo de observaciones de un nodo que se va a dividir y está directamente relacionado con el tamaño del árbol. Cuanto mayor sea el valor de `numStop`, menor será el tamaño del árbol.
- `fold` se refiere al número de partes en que se divide el conjunto de datos para aplicar la técnica de validación cruzada. Las variaciones en el argumento `fold` no están directamente relacionadas con el tamaño del árbol.
- `max.depth` limita el número de nodos entre el nodo raíz y el nodo hoja más lejano (raíz no incluida). Cuando se introduce este argumento, no se lleva a cabo el típico proceso de poda. En este caso, se permite que el árbol crezca hasta la profundidad requerida.
- `max.leaves` determina el número máximo de nodos hoja. Al igual que en `max.depth`, no se realiza el proceso de poda, sino que el árbol crece hasta alcanzar el número requerido de nodos hoja y, entonces, se devuelve el árbol.

Hay que tener en cuenta que la inclusión de los argumentos `max.depth` o `max.leaves` reduce el tiempo de cálculo al eliminar el procedimiento de poda. Sin embargo, el proceso de poda es preferible si el objetivo del estudio es inferencial en lugar de simplemente descriptivo. Si se incluyen ambos al mismo tiempo, se muestra un mensaje de advertencia y sólo se utiliza `max.depth`.

Como ejemplo de esta función se usa los datos PISA anteriormente explicados para crear un árbol multi respuesta usando el siguiente código. Como resultado de esta función se devuelve la estructura del objeto explicado también en la subsección anterior.

```
modelEAT <- EAT(data = PISAindex,
                x = 6:18, # input
                y = 3:5 # output)
```

4.4.3.2. EL MODELO RF+EAT

El modelo de Random Forest for Efficiency Analysis Treesse puede implementar en R utilizando la función `RFEAT(data, x, y, …)`. La función `RFEAT` también se ha desarrollado con el objetivo de proporcionar mayor robustez estadística a los resultados obtenidos por la función `EAT`. La función `RFEAT` requiere los datos que contienen las variables para el análisis, `x` e `y` correspondientes a los índices de entradas y salidas respectivamente, el número mínimo de observaciones en un nodo para que se intente una división (`numStop`) y `na.rm` para ignorar las observaciones con celdas NA, es decir, vacías. Todos estos argumentos se utilizan para la construcción de los *p* árboles individuales EAT que componen el bosque aleatorio RF+EAT. Por último, el argumento `s_mtry` indica el número de inputs que se pueden seleccionar aleatoriamente en cada división. Puede establecerse como cualquier número entero, aunque también hay ciertos valores predefinidos. Por lo tanto, siendo *m* el número de inputs, *s* el número de outputs y $n(t)$ el número de observaciones que contiene el nodo, las opciones disponibles para `s_mtry` son:

- `BRM` = $\frac{m}{3}$
- `DEA1` = $\frac{n(t)}{2} - s$
- DEA2 = $\frac{n(t)}{3} - s$

- DEA3 = $\frac{n(t)}{2s}$
- DEA4 = $\min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$

Como ejemplo se crea el modelo `RFEAT` con 30 árboles que devuelve la estructura de objeto explicada en la anterior subsección:

```
forest <- RFEAT(data = PISAindex,
                x = 6:18, # input
                y = 3:5, # output
                numStop = 5,
                p = 30,
                s_mtry = "BRM",
                na.rm = TRUE)
```

4.4.3.3. PREDICCIONES

Los estimadores de Efficiency Analysis Trees y Random Forest for Efficiency Analysis Trees pueden usarse en R utilizando `predictEAT(object, data, x)` y `predictRFEAT(object, data, x)`, respectivamente. Estas funciones devuelven un *Data Frame* con los datos y el resultado esperado para un conjunto de observaciones. En ambos casos, `newdata` se refiere al *Data Frame* y `x` al conjunto de inputs usados. En cuanto al argumento `object`, en el primer caso, se corresponde a un objeto `EAT` y, en el segundo caso, a un objeto RF`EAT`.

Para las predicciones con un objeto `EAT` sólo es necesario un árbol. Sin embargo, para el modelo RFEAT, el output viene dada por cada uno de los *p* árboles individuales entrenados que, posteriormente, se calcula el valor medio de todas las predicciones.

Como ejemplo, a continuación, se evalúan las últimas 10 DMUs del ejemplo utilizado en la Sección 4.4.2.. Los resultados se devuelven en una estructura *Data Frame* con las predicciones del output.

```
predictEAT(object = modelEAT, newdata = tail(data, 10), x = x)
-----------------------------------------------------------
       S_PISA_pred R_PISA_pred M_PISA_pred
1              396         399         393
2              429         420         423
3              429         420         423
4              487         479         496
5              396         377         379
6              396         399         393
7              396         377         379
8              396         377         379
9              396         377         379
10             396         377         379
```

```
predictRFEAT(object = forest, newdata = tail(data, 10), x = x)
-------------------------------------------------------------
       S_PISA_pred R_PISA_pred M_PISA_pred
1        400.02      393.18      401.92
2        421.78      418.16      413.56
3        420.74      417.80      412.44
4        448.74      447.32      443.68
5        382.60      368.40      371.38
6        412.02      411.96      412.58
7        384.54      371.28      373.22
8        392.58      382.12      382.94
9        379.28      365.68      370.52
10       376.30      364.94      365.42
```

Del mismo modo, el usuario puede crear nuevos conjuntos de datos y calcular su predicción de la siguiente manera:

```
new <- data.frame(NBMC = c(90, 95, 93),
                  WS = c(87, 92, 99),
                  S = c(93, 90, 90),
                  HW = c(90, 91, 92),
                  AAE = c(88, 91, 89))
```

```
predictions_EAT <- predictEAT(object = modelEAT,
                              newdata = new,
                              x = 1:5)
```

4.4.3.4. IMPORTANCIA DE LAS VARIABLES

La forma de calcular la importancia de los valores de los predictores mediante los *Efficiency Analysis Trees* es utilizando la función `rankingEAT(objeto, ...)`. La puntuación de la importancia representa la influencia de cada variable en el modelo. El usuario puede especificar el número de unidades decimales mediante el parámetro `digits`, incluir un gráfico de barras con las puntuaciones de la importancia usando `barplot` y mostrar una línea horizontal en el gráfico para facilitar el punto de corte entre las variables importantes e irrelevantes con `threshold`.

Esta función devuelve el nombre de las variables predictoras, la puntuación de importancia y el gráfico de barras (ver Figura 22), tal y como se puede ver en el siguiente ejemplo.

```
rankingEAT(object = modelEAT, threshold = 70, barplot = TRUE,
digits = 2)
------------------------------------------------------------
                Importance
AAE             100.00
WS               97.65
NBMC             82.55
HW               82.49
S                82.24
ABK              67.16
GDP_PPP          64.54
AIC              64.08
EQ               56.29
PR               56.22
I                56.22
PS               45.38
PFC              29.35
```

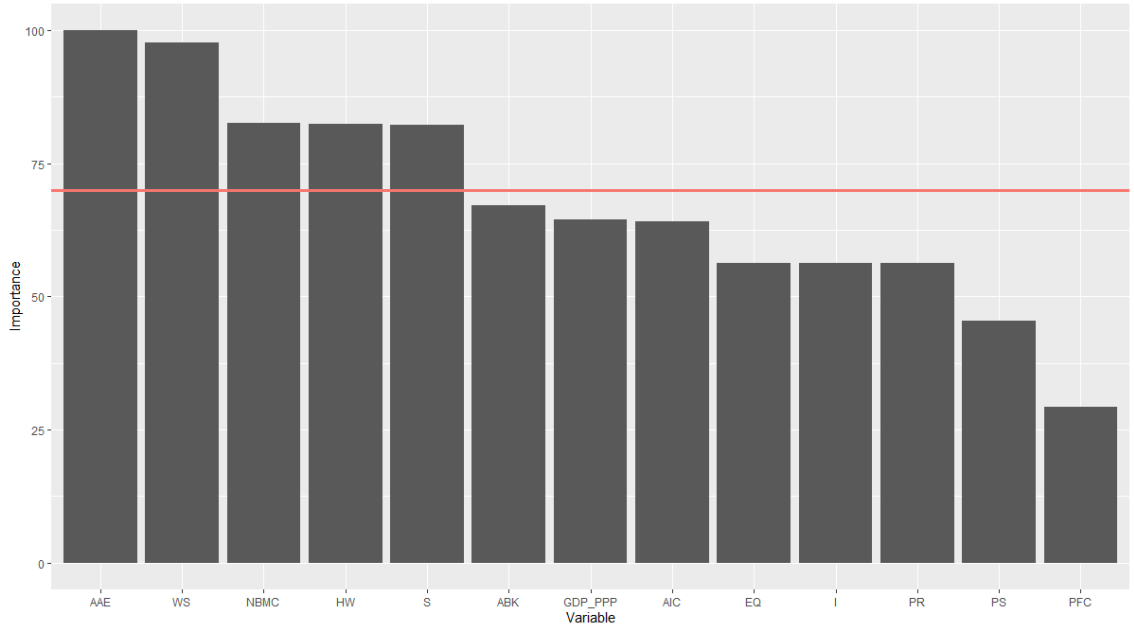


Figura 22. Gráfico de barras de la importancia de las variables para los EAT en los datos de `PISAindex` mediante la función `rankingEAT()`.

Al igual que en `rankingEAT()`, la función `rankingRFEAT()` permite calcular una puntuación de la importancia para las variables utilizando un objeto `RFEAT`. Es posible que puedan aparecer puntuaciones negativas al calcular la importancia de las variables mediante `rankingRFEAT()`. La aparición de este tipo de puntuación (negativa) puede entenderse como que, si se eliminara esa variable del modelo, se produciría una mejora en la capacidad predictiva del mismo.

```
rankingRFEAT(object = modelRFEAT, barplot = TRUE)
------------------------------------------------------------
        Importance
PFC      7.1977380
HW       2.8658000
EQ       2.2628482
I        1.6523514
AIC      0.9298688
PR      -0.5867114
AAE     -0.6801898
S       -0.8349269
PS      -1.0268028
GDP_PPP -1.2789753
```

```
NBMC     -1.8264092
ABK      -2.8606368
WS       -4.0611915
```

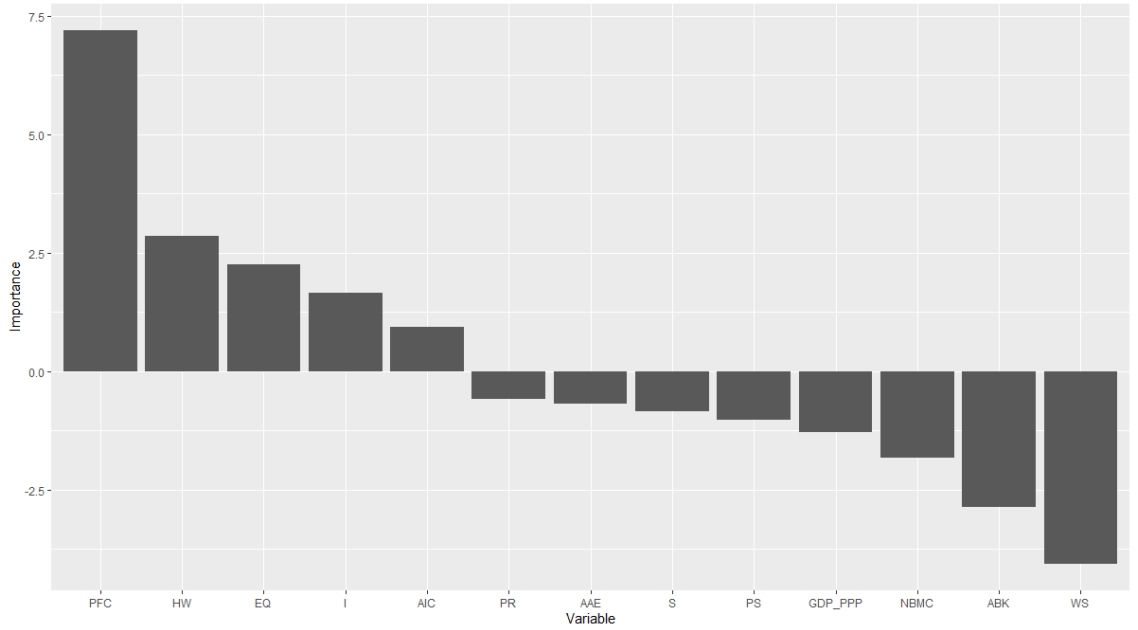


Figura 23. Gráfico de barras de la importancia de la variable para los EAT en los datos de `PISAindex` mediante `rankingRFEAT()`.

4.4.4. MODELOS BÁSICOS DE EAT Y RFEAT

Las puntuaciones de eficiencia son valores numéricos que indican el grado de eficiencia de un conjunto de unidades de decisión (DMU). Se debe introducir un conjunto de datos (`data`) y los correspondientes índices de inputs(s) (`x`) y outputs(s) (`y`). Se recomienda que el conjunto de datos con las DMU cuya eficiencia se va a calcular coincida con los utilizados para estimar la frontera. Sin embargo, también es posible calcular las puntuaciones de eficiencia para un nuevo conjunto de datos. Las puntuaciones de eficiencia se calculan utilizando el modelo de programación matemática incluido en el argumento `score_model`. En el paquete de R están disponibles los siguientes modelos:

- `BCC.OUT`: El modelo radial orientado al output (Banker et al., 1984).
- `BCC.INP`: El modelo radial orientado al input (Banker et al., 1984).
- `RSL.OUT`: El modelo Russell orientado al output (Färe and Lovell, 1978).
- `RSL.INP`: El modelo Russell orientado al input (Färe and Lovell, 1978).
- `DDF`: La función de distancia direccional (Chambers et al., 1998).

- `WAM.MIP`: La medida de las proporciones de ineficacia como un tipo de modelo aditivo ponderado (Cooper et al., 1999).
- `WAM.RAM`: La medida de ineficacia ajustada al rango como un tipo de modelo aditivo ponderado (Cooper et al., 1999).

Las puntuaciones de FDH pueden calcularse opcionalmente estableciendo `FDH = TRUE` en `efficiencyEAT()` y `efficiencyREAT()`. Las puntuaciones DEA también pueden calcularse estableciendo `DEA = TRUE` en la función `efficiencyCEAT()`. Por último, se incluye un breve resumen de la distribución de las puntuaciones calculadas a partir de cada modelo.

4.4.4.1. EL MODELO RADIAL ORIENTADO AL OUTPUT

El modelo radial orientado al output puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` con el parámetro `scores_model` establecido en "`BCC.OUT`". Los resultados se devuelven en una estructura *Data Frame* con los valores de input y output y la puntuación de eficiencia BCC orientada al output. Se muestra un ejemplo de este código:

```
eff <- efficiencyEAT(data = PISAindex,
                     x = c(6:18),
                     y = c(3,4,5),
                     object = modelEAT,
                     scores_model = "BCC.OUT",
                     digits = 3,
                     FDH=TRUE,
                     na.rm = TRUE)
-----------------------------------------------------------
      EAT_BCC_out FDH_BCC_out
SGP       1.000       1.000
JPN       1.042       1.000
KOR       1.000       1.000
…
PAN       1.000       1.000
PHL       1.074       1.000
DOM       1.102       1.000
```

```
 Model Mean Std. Dev. Min Q1 Median   Q3  Max
   EAT 1.02      0.03   1  1   1.01 1.01 1.16

 Model Mean Std. Dev. Min Q1 Median Q3  Max
   FDH    1      0.01   1  1      1  1 1.04
```

La convexificación de EAT puede calcularse en R utilizando la función `efficiencyCEAT(data, x, y, ...)`. Siguiendo con el mismo ejemplo:

```
eff <- efficiencyCEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = "BCC.OUT",
                      digits = 3,
                      DEA=TRUE,
                      na.rm = TRUE)
-------------------------------------------------------------
       CEAT_BCC_out DEA_BCC_out
SGP         1.000       1.000
JPN         1.042       1.000
KOR         1.060       1.000
…
PAN         1.371       1.084
PHL         1.199       1.000
DOM         1.326       1.000

 Model Mean Std. Dev. Min  Q1 Median   Q3  Max
  CEAT 1.17      0.09   1 1.1   1.15 1.15 1.37

 Model Mean Std. Dev. Min Q1 Median   Q3  Max
   DEA 1.03      0.04   1  1   1.01 1.01 1.17
```

Como en `efficiencyEAT()`, la función `efficiencyRFEAT()` devuelve las puntuaciones de eficiencia bajo Random Forest para un conjunto de DMUs. Sin embargo, en este caso, sólo está disponible para el modelo BCC orientado al output. Las puntuaciones FDH también pueden calcularse estableciendo `FDH = TRUE`. Los resultados

se devuelven en una estructura *Data Frame* con los valores de los inputs y outputs y las puntuaciones de eficiencia de la BCC orientadas al output. Continuando con el ejemplo.

```
eff <- efficiencyRFEAT(data = PISAindex,
                       x = c(6:18),
                       y = c(3,4,5),
                       object = forest,
                       digits = 3,
                       FDH=TRUE)
-------------------------------------------------------------
       RFEAT_BCC_out FDH_BCC_out
SGP            0.968       1.000
JPN            1.005       1.000
KOR            0.993       1.000
…
PAN            1.016       1.000
PHL            1.034       1.000
DOM            1.066       1.000

 Model Mean Std. Dev.  Min   Q1 Median Q3 Max
 RFEAT 1.01      0.03 0.96 0.99      1  1 1.1

 Model Mean Std. Dev. Min Q1 Median Q3  Max
   FDH    1      0.01   1  1      1  1 1.04
```

4.4.4.2. EL MODELO RADIAL ORIENTADO AL INPUT

El modelo radial orientado al input puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` estableciendo `score_model = "BCC.INP"`. Los resultados se devuelven en una estructura *Data Frame* con los valores de input y output y las puntuaciones de eficiencia orientada al input de BCC. Viendo un ejemplo:

```
eff <- efficiencyEAT(data = PISAindex,
                       x = c(6:18),
                       y = c(3,4,5),
                       object = modelEAT,
                       scores_model = “BCC.INP”,
                       digits = 3,
                       FDH=TRUE,
                       na.rm = TRUE)
```

```
-----------------------------------------------------------
       EAT_BCC_in FDH_BCC_in
SGP      1.000      1.000
JPN      1.000      1.000
KOR      0.989      1.000
…
PAN      0.884      1.000
PHL      1.000      1.000
DOM      1.000      1.000

 Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
   EAT 0.98      0.03 0.83 0.98   0.99 0.99   1

 Model Mean Std. Dev.  Min Q1 Median Q3 Max
   FDH    1      0.01 0.97  1      1  1   1
```

Para el caso de la convexificación, utilizamos la función `efficiencyCEAT(data, x, y, ...)`. Se muestra un ejemplo del código:

```
eff <- efficiencyCEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = “BCC.INP”,
                      digits = 3,
                      DEA=TRUE,
                      na.rm = TRUE)

-----------------------------------------------------------
       CEAT_BCC_in DEA_BCC_in
SGP       1.000      1.000
JPN       0.937      1.000
KOR       0.898      1.000
…
PAN       0.884      0.961
PHL       1.000      1.000
DOM       1.000      1.000

 Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
  CEAT 0.89      0.06 0.81 0.85   0.87 0.87   1

 Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
   DEA 0.98      0.02 0.92 0.96   0.99 0.99   1
```

4.4.4.3. EL MODELO RUSSELL ORIENTADO AL OUTPUT

La medida de Russell orientada al output puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` con el parámetro `scores_model` establecido en `"RSL.OUT"`. Los resultados se devuelven en una estructura *Data Frame* con los valores de input y output y las puntuaciones de eficiencia de Russell orientadas al output. Se muestra un ejemplo:

```
eff <- efficiencyEAT(data = PISAindex,
                     x = c(6:18),
                     y = c(3,4,5),
                     object = modelEAT,
                     scores_model = "RSL.OUT",
                     digits = 3,
                     FDH=TRUE,
                     na.rm = TRUE)
------------------------------------------------------------

       EAT_RSL_out FDH_RSL_out
SGP       1.000       1.000
JPN       1.070       1.000
KOR       1.006       1.000
…
PAN       1.053       1.000
PHL       1.097       1.000
DOM       1.149       1.000

 Model Mean Std. Dev. Min   Q1 Median   Q3  Max
   EAT 1.04      0.03   1 1.01   1.03 1.03 1.17

 Model Mean Std. Dev. Min Q1 Median Q3  Max

   FDH 1.01      0.01   1  1      1  1 1.07
```

Finalmente, los árboles de análisis de eficiencia convexos pueden calcularse utilizando `efficiencyCEAT(data, x, y, ...)` con `score_model` establecido en `"RSL.OUT"`. Continuando con el ejemplo:

```
eff <- efficiencyCEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = “RSL.OUT”,
                      digits = 3,
                      DEA=TRUE,
                      na.rm = TRUE)
-----------------------------------------------------------
       CEAT_RSL_out DEA_RSL_out
SGP           1.000       1.000
JPN           1.070       1.004
KOR           1.068       1.000
…
PAN           1.439       1.140
PHL           1.209       1.000
DOM           1.374       1.000

 Model Mean Std. Dev. Min   Q1 Median   Q3  Max
  CEAT 1.19       0.1   1 1.11   1.16 1.16 1.44

 Model Mean Std. Dev. Min Q1 Median   Q3  Max
   DEA 1.04      0.04   1  1   1.02 1.02 1.19
```

4.4.4.4. EL MODELO RUSSELL ORIENTADO AL INPUT

La medida de Russell orientada al input puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` con el parámetro `scores_model` establecido en `"RSL.INP"`. Los resultados se devuelven en una estructura *Data Frame* con los valores de input y output y las puntuaciones de eficiencia de Russell orientadas al input. A continuación, se muestra un ejemplo del código a utilizar.

```
eff <- efficiencyEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = “RSL.INP”,
                      digits = 3,
                      FDH=TRUE,
                      na.rm = TRUE)
-----------------------------------------------------------

       EAT_RSL_in FDH_RSL_in
```

```
SGP      0.588      1.000
JPN      0.571      1.000
KOR      0.576      1.000
…
PAN      0.624      0.789
PHL      0.764      0.992
DOM      0.707      0.898

 Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
   EAT 0.62      0.06 0.54 0.57   0.61 0.61 0.78

 Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
   FDH 0.93      0.06 0.77 0.89   0.93 0.93   1
```

Bajo el supuesto de la convexidad de la tecnología, la función `efficiencyCEAT(data, x, y, ...)` debe utilizarse con `score_model` ajustado a `"RSL.INP"`. Viendo un ejemplo.

```
eff <- efficiencyCEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = “RSL.INP”,
                      digits = 3,
                      DEA=TRUE,
                      na.rm = TRUE)
------------------------------------------------------------
       CEAT_RSL_in DEA_RSL_in
SGP       0.588      1.000
JPN       0.558      0.956
KOR       0.555      0.933
…
PAN       0.624      0.789
PHL       0.764      0.992
DOM       0.707      0.898

 Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
  CEAT 0.61      0.07 0.52 0.56   0.59 0.59 0.78

 Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
   DEA 0.88      0.07 0.76 0.84   0.88 0.88   1
```

4.4.4.5. LA FUNCIÓN DE DISTANCIA DIRECCIONAL

La función distancia direccional puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` con `scores_model` establecido como `"DDF"`. Los resultados se devuelven en una estructura *Data Frame* con los valores de input y output y las puntuaciones de eficiencia DDF. Mostramos un ejemplo del código abajo.

```
eff <- efficiencyEAT(data = PISAindex,
                     x = c(6:18),
                     y = c(3,4,5),
                     object = modelEAT,
                     scores_model = "DDF",
                     digits = 3,
                     FDH=TRUE,
                     na.rm = TRUE)
------------------------------------------------------------
     EAT_DDF FDH_DDF
SGP    0.000   0.000
JPN    0.000   0.000
KOR    0.000   0.000
…
PAN    0.000   0.000
PHL    0.000   0.000
DOM    0.000   0.000


 Model Mean Std. Dev. Min Q1 Median Q3  Max
   EAT 0.01      0.01   0  0      0  0 0.09

 Model Mean Std. Dev. Min Q1 Median Q3  Max
   FDH    0         0   0  0      0  0 0.02
```

Además, la convexificación de EAT puede calcularse en R utilizando la función `efficiencyCEAT(data, x, y, ...)` con `score_model` ajustado a `"DDF"`. Se muestra un ejemplo de esta posibilidad.

```
eff <- efficiencyCEAT(data = PISAindex,
                     x = c(6:18),
                     y = c(3,4,5),
                     object = modelEAT,
                     scores_model = "DDF",
                     digits = 3,
```

```
                    DEA=TRUE,
                    na.rm = TRUE)
-------------------------------------------------------------
      CEAT_DDF DEA_DDF
SGP    0.000   0.000
JPN    0.025   0.000
KOR    0.041   0.000
…
PAN    0.092   0.028
PHL    0.000   0.000
DOM    0.000   0.000

 Model Mean Std. Dev. Min   Q1 Median   Q3  Max
  CEAT 0.06      0.04   0 0.04   0.07 0.07 0.15

 Model Mean Std. Dev. Min Q1 Median Q3  Max
   DEA 0.01      0.01   0  0      0  0 0.05
```

4.4.4.6. LA MEDIDA DE LAS PROPORCIONES DE INEFICIENCIA Y LA MEDIDA DE INEFICIENCIA AJUSTADA AL RANGO COMO TIPOS DE MODELOS ADITIVOS PONDERADOS

El modelo aditivo ponderado puede calcularse en R utilizando la función `efficiencyEAT(data, x, y, ...)` con `scores_model` ajustado a `"WAM.MIP"` para la Medida de Proporciones de Ineficiencia o `"WAM.RAM"` para la Medida de Ineficiencia Ajustada al Rango (ver Cooper et al., 1999). Los resultados se devuelven en una estructura *Data Frame* con los valores de los inputs y outputs y las puntuaciones correspondientes al modelo aditivo ponderado. Se muestra un ejemplo del código correspondiente.

```
eff <- efficiencyEAT(data = PISAindex,
                    x = c(6:18),
                    y = c(3,4,5),
                    object = modelEAT,
                    scores_model = “WAM.MIP”,
                    digits = 3,
                    FDH=TRUE,
                    na.rm = TRUE)
-------------------------------------------------------------
```

```
      EAT_WAM    FDH_WAM
SGP   5.359      0.000
JPN   5.782      0.000
KOR   4.945      0.000
…
PAN   5.041      0.000
PHL   3.362      0.000
DOM   4.259      0.000

 Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
   EAT 4.71      0.81 2.72 4.11   4.86 4.86 6.11

 Model          Mean    Std. Dev.    Min Q1 Median Q3  Max
   FDH -5.633803e+28 2.322144e+29 -1e+30  0      0  0 2.73
```

El modelo aditivo ponderado asociado al Convexified Efficiency Analysis Trees se calcula mediante `efficiencyCEAT(data, x, y, ...)` con el parámetro `score_model` establecido en `"WAM.MIP"` o `"WAM.RAM"`. A continuación, se muestra un ejemplo.

```
eff <- efficiencyCEAT(data = PISAindex,
                      x = c(6:18),
                      y = c(3,4,5),
                      object = modelEAT,
                      scores_model = "WAM.MIP",
                      digits = 3,
                      DEA=TRUE,
                      na.rm = TRUE)
------------------------------------------------------------
      CEAT_WAM DEA_WAM
SGP    5.359   0.000
JPN    5.821   0.000
KOR    5.789   0.000
…
PAN    5.556   2.835
PHL    3.483   0.000
DOM    4.665   0.000

Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
  CEAT 5.27      0.75 3.32 4.75   5.45 5.45 6.24

 Model Mean Std. Dev. Min Q1 Median   Q3  Max
   DEA 1.05      1.07   0  0   1.08 1.08 3.13
```

4.4.5. OPCIONES AVANZADAS, VISUALIZACIÓN Y EXPORTACIÓN DE RESULTADOS

4.4.5.1. OPCIONES AVANZADAS DE OPTIMIZACIÓN

A veces es necesario encontrar el valor de los hiperparámetros que minimizan el error calculado a partir de la muestra de prueba (*test*, en inglés) mediante EAT "profundo" (sin podar) utilizando una muestra de entrenamiento (*train*, en inglés). En esta sección, la base de datos `PISAindex`, explicada al principio de este capítulo de tesis, se divide en un subconjunto de entrenamiento con el 70% de las DMUs y un subconjunto de prueba con el 30% restante. A continuación, se aplica la función `bestEAT()` para encontrar el valor de los hiperparámetros que minimizan el error calculado en la muestra de prueba a partir de EAT ajustado con la muestra de entrenamiento.

Primero se genera la muestra de entrenamiento y prueba:

```
n <- nrow(PISAindex)                # Observations in the dataset
selected <- sample(1:n, n * 0.7)    # Training indexes
training <- PISAindex[selected, ]   # Training set
test <- PISAindex[- selected, ]     # Test set
```

La función `bestEAT()` requiere un conjunto de entrenamiento (`training`) sobre el que ajustar un modelo EAT y un conjunto de prueba (`test`) sobre el que calcular el error. El número de árboles construidos viene dado por el número de combinaciones diferentes que pueden dar los argumentos `numStop`, `fold`, `max.depth` y `max.leaves`. Hay que tener en cuenta que no es posible introducir `NULL` y un valor determinado en los argumentos `max.depth` o `max.leaves` al mismo tiempo. La función `bestEAT()` devuelve un *Data Frame* con las siguientes columnas:

- `numStop`: Valor del hiperparámetro `numStop`.
- `fold`: Valor del hiperparámetro `fold`.

- `max.depth`: Valor del hiperparámetro `max.depth`. Si no se establece en `NULL`.
- `max.leaves`: Valor del hiperparámetro `max.leaves`. Si no se establece en `NULL`.
- `RMSE`: Error cuadrático medio calculado en la muestra de prueba con un árbol construido con la muestra de entrenamiento y el conjunto de hiperparámetros dados.
- `leaves`: Número de nodos hoja en el árbol.

El código de la función es el siguiente:

```
bestEAT(training, test,
        x, y,
        numStop = 5,
        fold = 5,
        max.depth = NULL,
        max.leaves = NULL,
        na.rm = TRUE)
```

Por ejemplo, si los argumentos introducidos son `numStop = {3, 5, 7}` y `fold = {5, 7}`, se construyen seis modelos diferentes con `{numStop = 3, fold = 5}`, `{numStop = 3, fold = 7}`, `{numStop = 5, fold = 5}`, `{numStop = 5, fold = 7}`, `{numStop = 7, fold = 5}` y `{numStop = 7, fold = 7}`. Se muestra un ejemplo numérico:

```
bestEAT(training = training,
        test = test,
        x = c(6, 7, 8, 12, 17),
        y = 3:5,
        numStop = c(3, 5, 7),
        fold = c(5, 7))
------------------------------------------------------------
     numStop    fold  RMSE       leaves
1    3          7     56.82      24
2    3          5     58.81      13
```

```
3     7          5     59.14      10
4     7          7     59.14      10
5     5          7     60.13      12
6     5          5     60.24      11
```

El mejor modelo devuelto es el dado por los hiperparámetros `{numStop = 3, fold = 7}` con un `RMSE = 56.82` y 24 nodos hojas. Sin embargo, este modelo es muy complejo. Por lo tanto, alternativamente, podríamos seleccionar el modelo con parámetros `{numStop = 7, fold = 5}` que tiene un `RMSE = 59.14`, pero con solo 10 nodos hojas. Con este resultado, se ajusta el modelo EAT final utilizando todos los datos originales.

```
bestEAT_model <- EAT(data = PISAindex,
                     x = c(6, 7, 8, 12, 17),
                     y = 3:5,
                     numStop = 7,
                     fold = 5)
```

4.4.5.2. VISUALIZACIÓN PERSONALIZADA

Después de calcular los modelos `EAT` y `CEAT`, se muestran en pantalla algunas funciones de visualización personalizadas. La función `efficiencyJitter()` devuelve un gráfico de fluctuaciones mediante la librería `ggplot2`. Este gráfico muestra cómo las DMUs se agrupan en nodos hoja en un modelo construido con la función `EAT()`. Cada nodo hoja agrupa DMUs con el mismo nivel de recursos. Una línea de puntos y una línea negra representan, respectivamente, el valor medio y la desviación estándar de las puntuaciones de su nodo correspondiente. Además, las etiquetas de las DMU eficientes se muestran siempre en función del modelo introducido en el argumento `score_model`. Por último, el usuario puede especificar un límite superior (`upb`) y un límite inferior (`lwb`) para mostrar también las etiquetas cuya puntuación de eficiencia se encuentra entre ellos. También se pueden utilizar las puntuaciones de un modelo CEAT.

Como ejemplo de esta función, se evalúan las puntuaciones de eficiencia de EAT correspondientes al modelo radial orientado al output y las representamos mediante `efficiencyJitter()`.

```
efficiencyJitter(object = modelEAT,
                 df_scores = scores_EAT$EAT_BCC_OUT,
                 scores_model = "BCC.OUT",
                 lwb = 1.2)
-----------------------------------------------------------
```

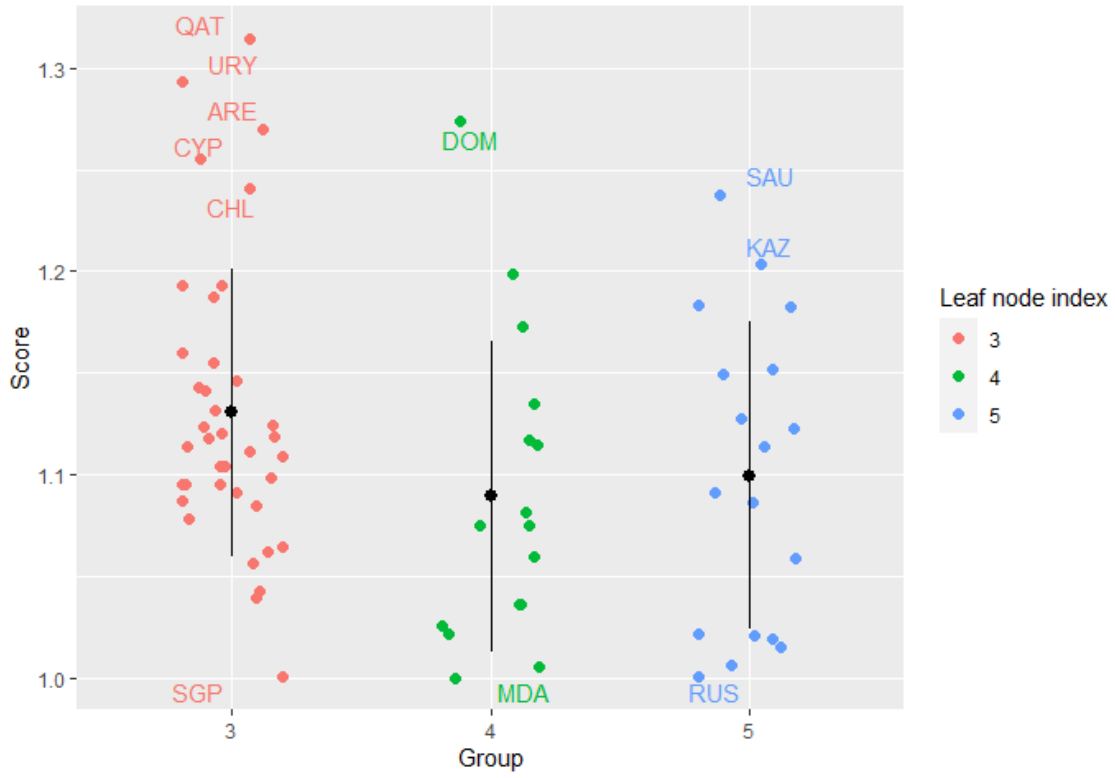


Figura 24. Gráfico de fluctuación que muestra cómo se agrupan las DMU en nodos hoja en un modelo construido con la función `EAT()`.

Otra función de visualización personalizada es `efficiencyDensity(df_scores, model)` que devuelve un gráfico de densidad usando la librería `ggplot2`. De esta manera, se puede comprobar la similitud entre las puntuaciones obtenidas por las diferentes metodologías disponibles.

En este ejemplo:

```
efficiencyDensity(df_scores = scores_EAT[, 3:4],
                  model = c("EAT", "FDH"))
```

```
------------------------------------------------------------
```

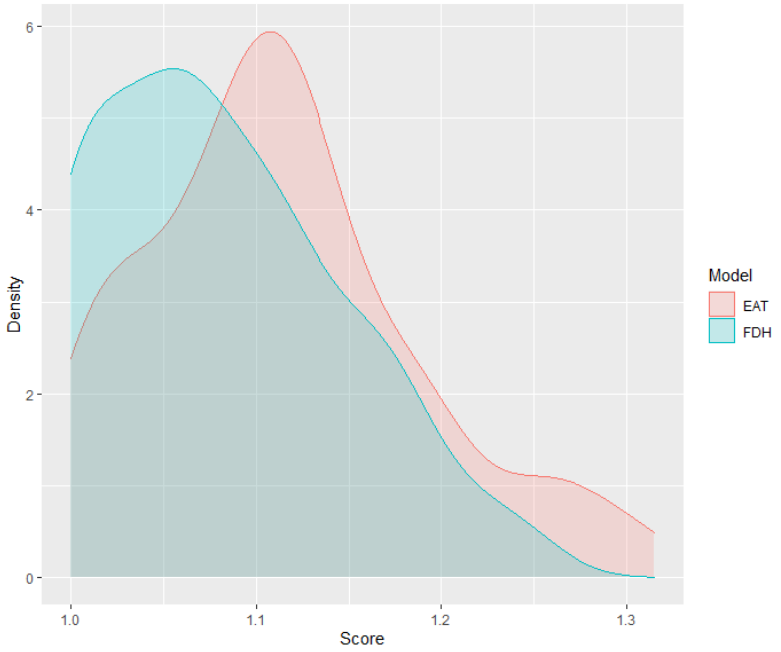


Figura 25. Gráfico de densidad que muestra las diferencias entre las puntuaciones obtenidas por las funciones `EAT()` y `FDH()`.

```
efficiencyDensity(df_scores = scores_CEAT[, 3:4],
                  model = c("CEAT", "DEA"))
```

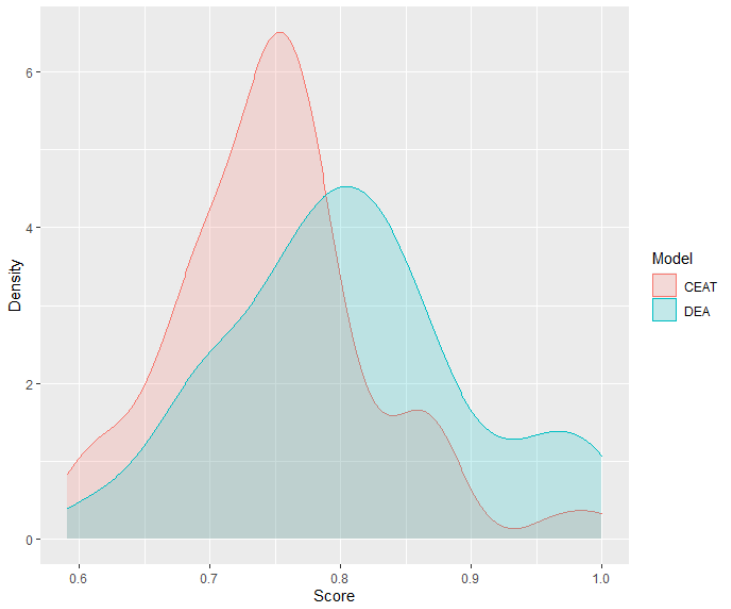


Figura 26. Gráfico de densidad que muestra las diferencias entre las puntuaciones obtenidas por las funciones `CEAT()` y `DEA()`.

Además, en el caso limitado de utilizar sólo un input para producir sólo un output, podemos mostrar la frontera estimada por la función `EAT()` a través de un gráfico de `ggplot2`. La frontera estimada por FDH también se puede representar si se establece `FDH = TRUE`. Las DMUs observadas pueden mostrarse mediante un gráfico de dispersión si `observed.data = TRUE` y, además, su color, forma y tamaño pueden modificarse con `observed.color`, `pch` y `size`, respectivamente. Finalmente, los nombres de las filas pueden incluirse con `rwn = TRUE`.

Como ejemplo, se utilizan los datos simulados del paquete *eat* para generar un input y un output mediante la función `Y1.sim(N = 50, nX = 1)`. El parámetro `N` corresponde al número de DMUs y el parámetro `nX` es el número de inputs.

```
simulated <- Y1.sim(N = 50, nX = 1)

modelEAT <- EAT(data = simulated, x = 1,y = 2)
```

```
frontier <- frontier(object = modelEAT, FDH = TRUE, observed.data
= TRUE, rwn = TRUE)

plot(frontier)

-------------------------------------------------------------
```

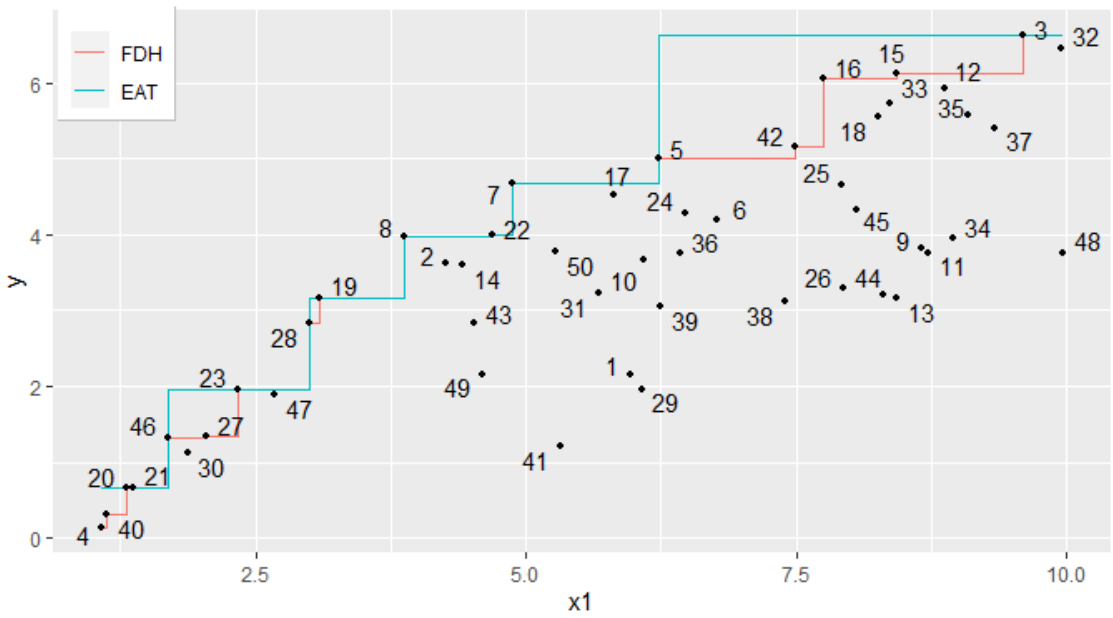


Figura 27. Gráfico de la estimación de la frontera mediante FDH y EAT.

4.4.5.3. EXPORTACIÓN DE RESULTADOS

La función `frontier()` mostrada anteriormente sólo funciona para el caso simple con un input y un output. Para escenarios de múltiples inputs y/o outputs, la estructura típica de árbol que muestra las relaciones entre los outputs y los inputs viene dada por la función `plotEAT()`.

Para cada nodo, se puede obtener la siguiente información:

- `id`: Índice del nodo.
- `R`: Error en el nodo.
- `n(t)`: Número de DMUs en el nodo.
- `Input name`: Variable de entrada asociada a la división.

- `y`: Vector de predicciones de output.

Además, los nodos se colorean según la variable por la que se realiza la división o son negros en el caso de ser un nodo hoja.

A continuación, se muestra cómo funciona la función `plotEAT()` vinculada a tres formas diferentes de controlar el tamaño de un modelo de árbol: `numStop`, `max.depth` y `max.leaves`:

1. Control del tamaño por `numStop`

```
reduced_model1 <- EAT(data = PISAindex,
                      x = c(6, 7, 8, 12, 17),   # input
                      y = 3:5,                  # output
                      numStop = 9)
plotEAT(object = reduced_model1)
-----------------------------------------------------------
```

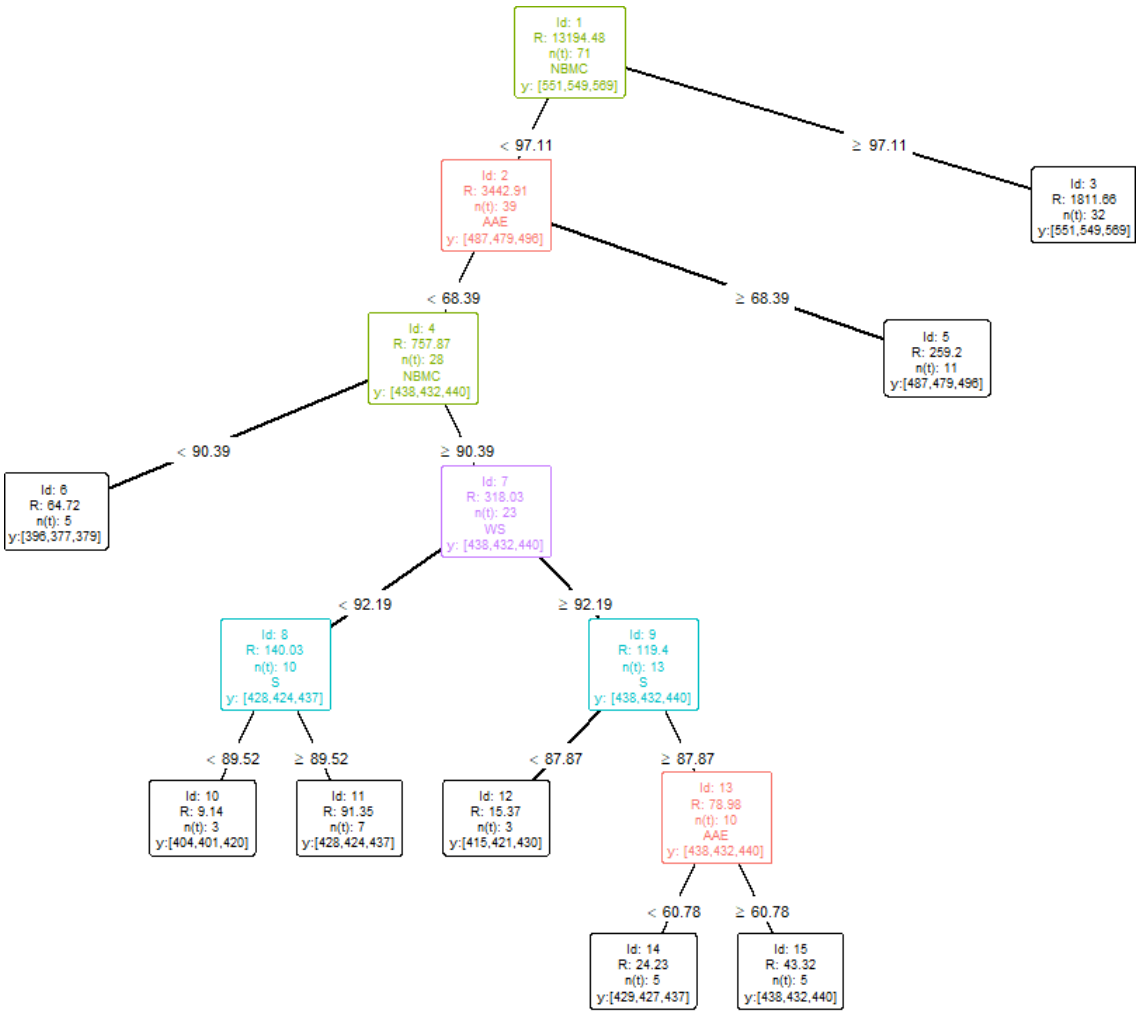


Figura 28. Gráfico del árbol construido por EAT utilizando `numStop`.

2. Control del tamaño por `max.depth`

```
reduced_model2 <- EAT(data = PISAindex,
                      x = c(6, 7, 8, 12, 17), # input
                      y = 3:5, # output
                      numStop = 9,
                      max.depth = 5)

plotEAT(reduced_model2)
-----------------------------------------------------------
```

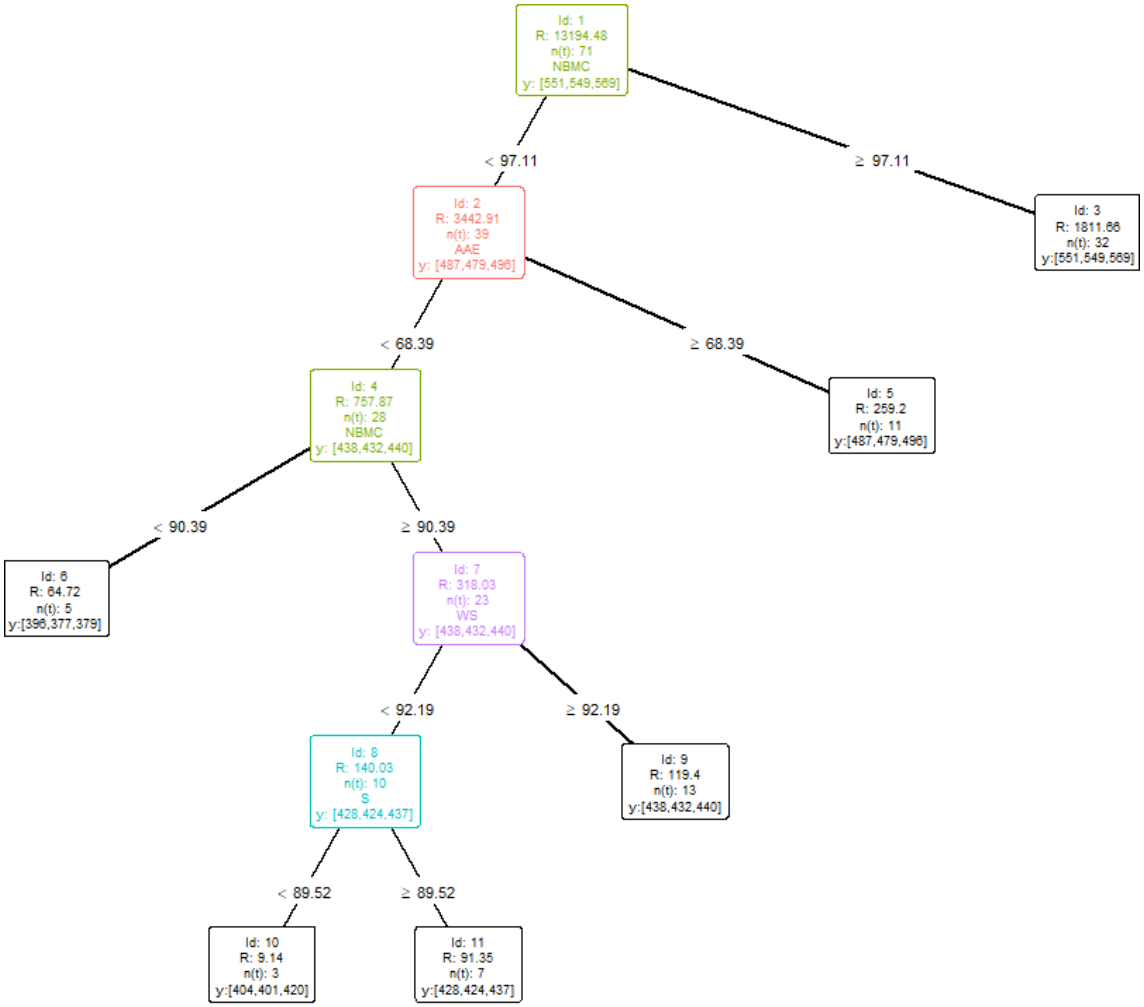


Figura 29. Gráfico del árbol construido por EAT utilizando `max.depth`.

3. Control del tamaño por `max.leaves`

```
reduced_model3 <- EAT(data = PISAindex,
                      x = c(6, 7, 8, 12, 17), # input
                      y = 3:5, # output
                      numStop = 9,
                      max.leaves = 4)
plotEAT(object = reduced_model3)

-------------------------------------------------------------
```

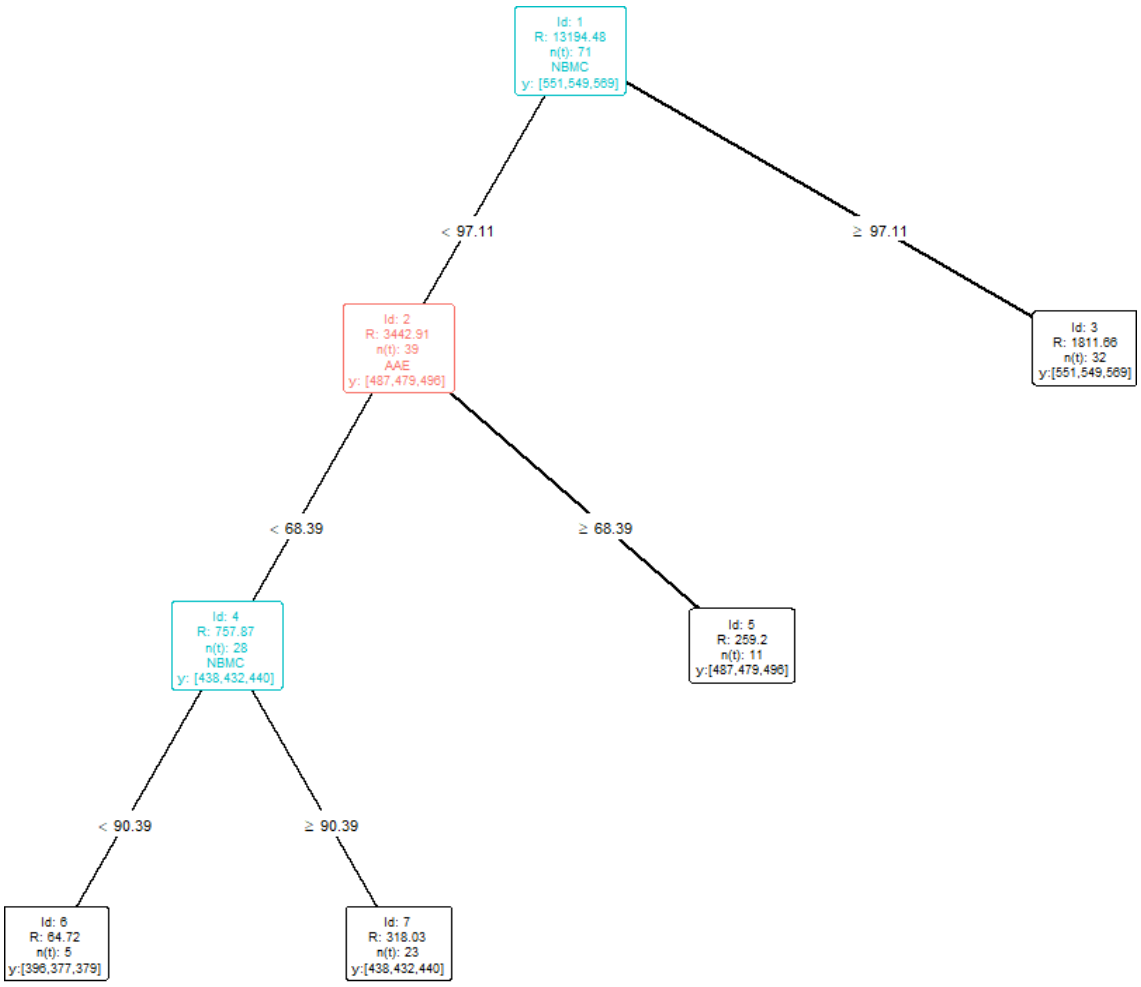


Figura 30. Gráfico del árbol construido por EAT utilizando `max.leaves`.

Finalmente, la función `plotRFEAT()` devuelve el error Out-Of-Bag para un conjunto de datos de entrenamiento y un bosque formado por *p* árboles, como se muestra a través de un ejemplo numérico.

```
plotRFEAT(forest)
-----------------------------------------------------------
```

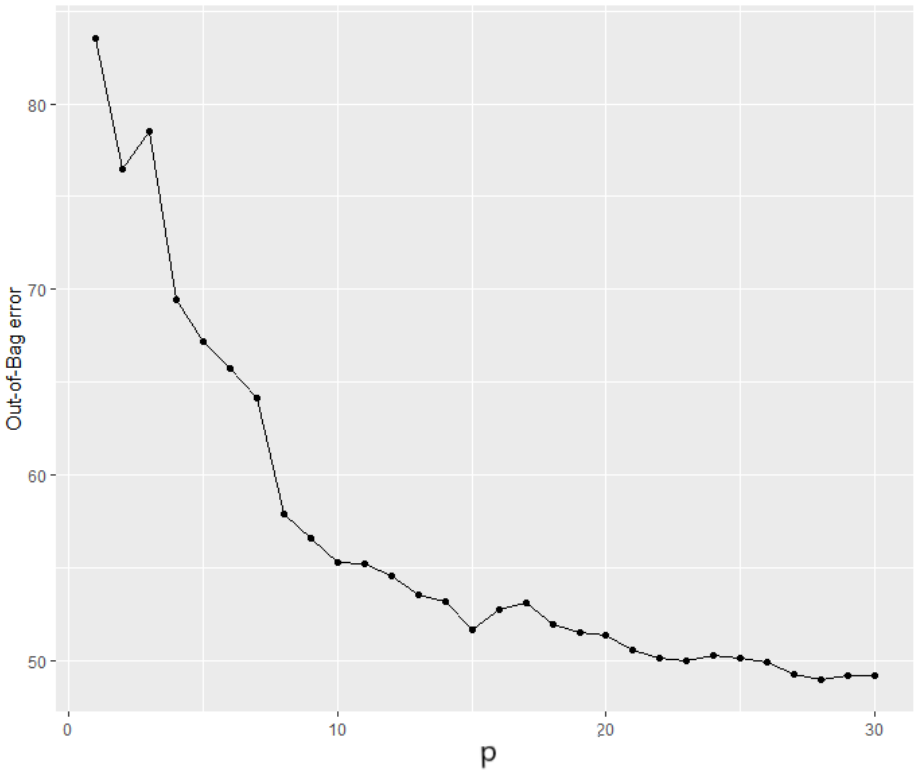


Figura 31. Gráfico del error Out-Of-Bag para el conjunto de datos de entrenamiento generado por un bosque formado por *p* árboles.

CAPÍTULO 5: DISCUSIÓN

En este capítulo de tesis se discuten los resultados obtenidos en los capítulos anteriores, relacionando estos con la literatura previa sobre la temática en cuestión.

Desde un punto de vista puramente metodológico, existe toda una literatura reciente que tiene como objetivo fundamental el uso de técnicas de aprendizaje automático o similar para la predicción de eficiencia y la determinación de la frontera de producción. En este sentido, Kuosmanen y Johnson (2010, 2017) mostraron cómo DEA puede ser interpretado como un modelo de regresión basado en mínimos cuadrados condicionado a restricciones de forma sobre la superficie a ajustar (la frontera de producción) y restricciones de signo de los residuos. Además, estos autores introdujeron en la literatura el modelo denominado CCNLS (*Corrected Concave Non-parametric Least Squares*, en inglés). Bajo este enfoque, muchos hiperplanos, a priori uno diferente para cada dato, deben de ser ajustados mediante la nube de puntos de las observaciones muestrales, mientras que la condición de convexidad de la tecnología así generada viene garantizada por el sistema de desigualdades de Afriat (1972). Por otro lado, Hernderson y Parmeter (2009), Du et al. (2013) y Parmeter y Racine (2013) introdujeron estimadores de fronteras de producción basados en *kernels* no paramétricos. Estos métodos proporcionan una aproximación suavizada de la frontera de producción. Además, Daouia et al. (2016) introdujeron un método que puede ser utilizado para estimar funciones de producción a través de 'trozos' de polinomios de grado dos y tres y teniendo en cuenta restricciones de forma (monotonía y concavidad). El modelo en cuestión es regularizado mediante la determinación de una cuadrícula de puntos óptimos para la subdivisión del espacio de los inputs. Tsionas (2022)

recientemente definió árboles de regresión probabilísticos, también condicionados a restricciones de forma (concavidad y suavizado) para estimar eficiencia. Valero-Carreras et al. (2021, 2022) adaptaron las máquinas de soporte vectorial para la determinación del grado de eficiencia absoluta (en lugar de relativa), introduciendo, a su vez, el concepto de eficiencia robusta denominado "épsilon-insensibilidad". Guerrero et al. (2022) han introducido recientemente una metodología que permite estimar tecnologías poliédricas siguiendo el principio de minimización del riesgo estructural. Por último, Olesen y Ruggiero (2018, 2022) propusieron el uso de hiperplanos articulados (Breiman, 1993) como una representación flexible no paramétrica de una función de producción. Comparando la nueva metodología, introducida en esta tesis doctoral, con la literatura previa, cabe indicar que las anteriores aproximaciones están basadas en estimadores tecnológicos que cumplen con la propiedad de convexidad (exceptuando el caso de Olesen y Ruggiero, 2022, que pretenden estimar una función en forma de S), mientras que, en el caso de los EAT, la base fundamental es el uso de estimadores no convexos. Esto cumple con las críticas recientemente formuladas en la literatura hacia un uso excesivo, e incluso en ocasiones innecesario, del postulado de convexidad del conjunto de posibilidades de producción (véase, por ejemplo, Kerstens y Van de Woestyne, 2021). Además, sólo nosotros y Tsionas proponemos la adaptación de métodos basados en árboles de regresión para trabajar en el contexto de la medición de la eficiencia. En el caso de Tsionas, se requiere del uso de árboles probabilísticos. En nuestro caso, la adaptación está fundamentada en el algoritmo original CART de Breiman et al. (1984).

Desde un punto de vista computacional, nuestros resultados incluyen la comparación sistemática de las estimaciones de frontera mono-output y multi-output obtenidas por la técnica FDH, la nueva técnica EAT y la adaptación de EAT al Random Forest (RF+EAT). También se incluyen los resultados del EAT profundo, es decir, EAT sin poda con la regla de parada de los nodos hoja finales a $n_{\min} = 1$ para probar su relación con el FDH. Por otra parte, se incluyen los resultados obtenidos tras la mejora computacional del EAT tras su adaptación a la típica técnica de árboles llamada *Backtracking*.

La primera comparación establecida fue la realizada entre el EAT profundo y el FDH donde se encontró que la discrepancia entre estas técnicas al hallar la estimación de la frontera es del 0%, es decir, ambas técnicas obtienen la misma estimación cuando se trabaja con un solo input y un solo output (y se dispone de un resultado teórico que avala estos resultados). Sin embargo, a medida que la muestra va aumentando el número de observaciones (DMUs) y el número de inputs, el porcentaje de discrepancia va aumentando (alcanzando un 65.33%). Ante estos resultados se observó que existe una relación entre el FDH y la nueva técnica EAT que merecía seguir siendo estudiada para poder superar el problema del sobreajuste de la técnica FDH. Esto se consiguió incorporando la poda de CART de Breiman et al. (1984) al EAT, debido a que incorpora una medida de complejidad de costes y un proceso de validación cruzada. Por otro lado, al igual que sucede en el CART, las estimaciones que devuelve el EAT no son lo suficientemente robustas. Por este motivo, se adaptó la técnica Random Forest de Breiman (2001) al EAT dando lugar a una nueva técnica bautizada como RF+EAT. Además, mediante esta adaptación se consiguió que la nueva técnica RF+EAT determinase la importancia de las variables de entrada del modelo y ofreciera una solución al conocido problema de la "maldición de la dimensionalidad" que sufren las técnicas FDH y DEA. Para la comparación de estos dos métodos, EAT (podado) y RF+EAT, con FDH se empleó distintos escenarios donde se muestra la mejora del error cuadrático medio (MSE) y del sesgo al aumentar el tamaño muestral y el número de inputs y outputs. En concreto, se observó que el EAT mejora al FDH en MSE entre el 13% y el 93% y en sesgo entre el 10% y el 49% al aumentar el número de DMUs y el número de inputs. Respecto al RF+EAT las mejoras oscilaron entre 62% y el 91% en MSE y entre el 38% y el 71% en sesgo. Además, el RF+EAT mostró en general mejores resultados respecto al EAT.

Ante la posible mejora de la precisión del estimador del EAT se recurrió a la técnica del *backtracking* y a técnicas heurísticas. Como se mostró en las experiencias simuladas, la incorporación del *backtracking* al EAT permitió mejorar el MSE hasta el 73%; aunque para conseguir estos resultados el consumo de tiempo computacional resultó ser

excesivamente elevado. Por este motivo, se recurrió, finalmente, a técnicas heurísticas que permitieron obtener mejoras del MSE hasta del 69% en un tiempo de cálculo mucho más razonable.

CAPÍTULO 6: REFERENCIAS

Adler N, Berechman J (2001) Measuring airport quality from the airlines' viewpoint: an application of data envelopment analysis. Transport Policy 8(3):171–181

Adler, N., & Golany, B. (2002). Including principal component weights to improve discrimination in data envelopment analysis. Journal of the Operational Research Society, 53(9), 985-991.

Afriat, S. N. (1972). Efficiency estimation of production functions. International economic review, 568-598.

Aigner, D. J., & Chu, S. F. (1968). On estimating the industry production function. The American Economic Review, 58(4), 826-839.

Aigner, D., Lovell, C. K., & Schmidt, P. (1977). Formulation and estimation of stochastic frontier production function models. Journal of econometrics, 6(1), 21-37.

Álvarez, I., Barbero, J., & Zofío, J. (2020). A data envelopment analysis toolbox for MATLAB. Journal of Statistical Software (Online), 95(3).

Amit, Y., & Geman, D. (1997). Shape quantization and recognition with randomized trees. Neural computation, 9(7), 1545-1588.

Aparicio, J., & Zofío, J. L. (2020). Economic Cross-Efficiency. Omega, 102374.

Aparicio, J., Esteve, M., Rodriguez-Sala, J. J. & Zofio, J. L. (2021). The estimation of productive efficiency through machine learning techniques: Efficiency Analysis Trees. In Data-Enabled Analytics: DEA for Big Data, Zhu, Joe, Charles, Vincent (Eds.). Springer.

Aparicio, J., Pastor, J. T., Vidal, F., & Zofío, J. L. (2017). Evaluating productive performance: A new approach based on the product-mix problem consistent with Data Envelopment Analysis. Omega, 67, 134-144.

Appice, A., & Džeroski, S. (2007). Stepwise induction of multi-target model trees. In European Conference on Machine Learning (pp. 502-509). Springer, Berlin, Heidelberg.

Aragon, Y., Daouia, A., & Thomas-Agnan, C. (2005). Nonparametric frontier estimation: a conditional quantile-based approach. Econometric Theory, 21(2), 358-389.

Araujo, C., Barros, C. P., & Wanke, P. (2014). Efficiency determinants and capacity issues in Brazilian for-profit hospitals. Health care management science, 17(2), 126-138.

Arnaboldi, M., Azzone, G., & Giorgino, M. (2014). Performance measurement and management for engineers. Academic Press.

Baase, S. (2009). Computer algorithms: introduction to design and analysis. Pearson Education India.

Badiezadeh, T., Saen, R. F., & Samavati, T. (2018). Assessing sustainability of supply chains by double frontier network DEA: A big data approach. Computers & Operations Research, 98, 284-290.

Balk, B. M., Barbero, J., & Zofío, J. L. (2018). A Total Factor Productivity Toolbox for MATLAB. Available at SSRN 3178911.

Balk, B. M., Barbero, J., & Zofío, J. L. (2020). A toolbox for calculating and decomposing Total Factor Productivity indices. Computers & Operations Research, 115, 104853.

Balk, B. M., de Koster, M.B.M., Kaps, C. & Zofío, J.L. (2017). An Evaluation of Cross-Efficiency Methods, Applied to Measuring Warehouse Performance. ERIM Report Series Research in Management ERS-2017-015-LIS, Erasmus University Rotterdam, The Netherlands. https://repub.eur.nl/pub/103185.

Banker, R. D. (1993). Maximum likelihood, consistency and data envelopment analysis: a statistical foundation. Management science, 39(10), 1265-1273.

Banker, R. D. (1996). Hypothesis tests using data envelopment analysis. Journal of productivity analysis, 7(2-3), 139-159.

Banker, R. D., & Maindiratta, A. (1992). Maximum likelihood estimation of monotone and concave production frontiers. Journal of Productivity Analysis, 3(4), 401-415.

Banker, R. D., Charnes, A., & Cooper, W. W. (1984). Some models for estimating technical and scale inefficiencies in data envelopment analysis. Management science, 30(9), 1078-1092.

Banker, R. D., Charnes, A., Cooper, W. W., & Clarke, R. (1989). Constrained game formulations and interpretations for data envelopment analysis. European Journal of Operational Research, 40(3), 299-308.

Barbosa, F., Rampazzo, P. C. B., Yamakami, A., & Camanho, A. S. (2019). The use of frontier techniques to identify efficient solutions for the Berth Allocation Problem solved with a hybrid evolutionary algorithm. Computers & Operations Research, 107, 43-60.

Berk, R. A. (2016). Statistical learning from a regression perspective. New York: Springer.

Bogetoft, P., & Otto, L. (2010). Benchmarking with dea, sfa, and r (Vol. 157). Springer Science & Business Media.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(5), 216-233.

Breiman, L. (1993). Hinging hyperplanes for regression, classification, and function approximation. IEEE Transactions on Information Theory, 39(3), 999-1013.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees. Taylor & Francis.

Browne, M. W. (2000). Cross-validation methods. Journal of mathematical psychology, 44(1), 108-132.

Cazals, C., Florens, J. P., & Simar, L. (2002). Nonparametric frontier estimation: a robust approach. Journal of Econometrics, 106(1), 1-25.

Chambers, R. G. (1988). Applied production analysis: a dual approach. Cambridge University Press.

Chambers, R. G., Chung, Y., & Färe, R. (1998). Profit, directional distance functions, and Nerlovian efficiency. Journal of optimization theory and applications, 98(2), 351-364.

Charles, V., Aparicio, J., & Zhu, J. (2020). Data Science and Productivity Analytics. Springer.

Charnes A, Cooper WW, Golany B, Seiford L, Stutz J (1985) Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions. J Econom 30(1–2): 91–107.

Charnes, A., Cooper, W. W., & Rhodes, E. (1978). Measuring the efficiency of decision making units. European journal of operational research, 2(6), 429-444.

Cobb, C. W., & Douglas, P. H. (1928). A theory of production. The American Economic Review, 18(1), 139-165.

Coelli, T. J., Rao, D. S. P., O'Donnell, C. J., & Battese, G. E. (2005). An introduction to efficiency and productivity analysis. Springer Science & Business Media.

Cook, W. D., & Zhu, J. (Eds.). (2014). Data envelopment analysis: A handbook of modeling internal structure and network (Vol. 208). Springer.

Cooper, W. W., Seiford, L. M. & Tone, K. (2007). Data envelopment analysis: a comprehensive text with models, applications, references and DEA-solver software (2nd ed.). New York, NY: Springer.

Cutler, A., & Zhao, G. (2001). Pert-perfect random tree ensembles. Computing Science and Statistics, 33, 490-497.

Daouia, A., Laurent, T., & Noh, H. (2017). npbr: A Package for Nonparametric Boundary Regression in R. Journal of Statistical Software, 79(9), 1-43.

Daraio, C., & Simar, L. (2005). Introducing environmental variables in nonparametric frontier models: a probabilistic approach. Journal of Productivity Analysis, 24(1), 93-121.

De'Ath, G. (2002). Multivariate regression trees: a new technique for modeling species–environment relationships. Ecology, 83(4), 1105-1117.

Debreu, G. (1951). The coefficient of resource utilization. Econometrica: Journal of the Econometric Society, 273-292.

Deprins, D., & Simar, L. (1984). Measuring labor efficiency in post offices, The Performance of Public Enterprises: Concepts and Measurements, M. Marchand, P. Pestieau and H. Tulkens.

Deprins, D., Simar, L., & Tulkens, H. (2006). Measuring labor-efficiency in post offices. In Public goods, environmental externalities and fiscal competition (pp. 285-309). Springer, Boston, MA.

Dietterich, T. G., & Kong, E. B. (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Department of Computer Science, Oregon State University.

Du, P., Parmeter, C. F., & Racine, J. S. (2013). Nonparametric kernel regression with multiple predictors and multiple shape constraints. Statistica Sinica, 1347-1371.

Du, Q., Gunzburger, M., Lehoucq, R. B., & Zhou, K. (2013). A nonlocal vector calculus, nonlocal volume-constrained problems, and nonlocal balance laws. Mathematical Models and Methods in Applied Sciences, 23(03), 493-540.

Dulá, J. H. (2008). A computational study of DEA with massive data sets. Computers & Operations Research, 35(4), 1191-1203.

Dulá, J. H., & Thrall, R. M. (2001). A computational framework for accelerating DEA. Journal of Productivity Analysis, 16(1), 63-78.

Dyson, R. G., Allen, R., Camanho, A. S., Podinovski, V. V., Sarrico, C. S., & Shale, E. A. (2001). Pitfalls and protocols in DEA. European Journal of operational research, 132(2), 245-259.

Efron, B. (1979) Bootstrap methods: another look at the Jackknife. Annals of Statistics, vol. 7, no. 1, 1-26.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In Breakthroughs in statistics (pp. 569-593). Springer, New York, NY.

Emrouznejad, A., & Anouze, A. L. (2010). Data envelopment analysis with classification and regression tree–a case of banking efficiency. Expert Systems, 27(4), 231-246.

Esteve, M., Aparicio, J., Rabasa, A., & Rodriguez-Sala, J. J. (2020). Efficiency Analysis Trees: A New Methodology for Estimating Production Frontiers through Decision Trees. Expert Systems with Applications, 113783.

Esteve, M., Aparicio, J., Rodriguez-Sala, J. J. & Zhu, J. (2021). Estimation of production frontiers through Random Forest: the treatment of lack of robustness, ranking of inputs and curse of dimensionality under Free Disposal Hull. (en revisión)

Esteve, M., España, V. J., Aparicio, J & Barber, X. (2021). eat: An R Package for fitting Efficiency Analysis Trees. (en revisión).

Esteve, M., Rodriguez-Sala, J. J., Lopez-Espin, J. J., & Aparicio, J. (2021). Heuristic and Backtracking algorithms for improving the performance of Efficiency Analysis Trees. IEEE Access, 1-8.

Färe, R., & Lovell, C. K. (1978). Measuring the technical efficiency of production. Journal of Economic theory, 19(1), 150-162.

Färe, R., & Primont, D. (1995). Multi-output production and duality: theory and applications. Boston: Kluwer Academic Publishers.

Farrell, M. J. (1957). The measurement of productive efficiency. Journal of the Royal Statistical Society: Series A (General), 120(3), 253-281.

Ferrara G, Vidoli F (2018). semsfa: Semiparametric Estimation of Stochastic Frontier Models. R package version 1.1, URL https://CRAN.R-project.org/package=semsfa.

Fried, H. O., Schmidt, S. S., & Lovell, C. K. (Eds.). (1993). The measurement of productive efficiency: techniques and applications. Oxford university press.

Friedman, L., & Sinuany-Stern, Z. (1998). Combining ranking scales and selecting variables in the DEA context: The case of industrial branches. Computers & Operations Research, 25(9), 781-791.

Geurts, P. (2002). Contributions to decision tree induction: bias/variance tradeoff and time series classification (Doctoral dissertation, University of Liège Belgium).

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42.

Guerrero, N. M., Aparicio, J., & Valero-Carreras, D. (2022). Combining Data Envelopment Analysis and Machine Learning. Mathematics, 10(6), 909.

Golany, B., & Roll, Y. (1989). An application procedure for DEA. Omega, 17(3), 237-250.

Hall, P., & Huang, L. S. (2001). Nonparametric kernel regression subject to monotonicity constraints. Annals of Statistics, 624-647.

Henderson, D. J., & Parmeter, C. F. (2009). Imposing economic constraints in nonparametric regression: survey, implementation, and extension. Advances in Econometrics, 25(2009), 433-69.

Ho, T. K. (1998). The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence, 20(8), 832-844.

Homburg, C. (2001). Using data envelopment analysis to benchmark activities. International Journal of Production Economics, 73 (1), 51–58.

Horowitz, E., & Sahni, S. (1978). Fundamentals of computer algorithms. Computer Science Press.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

Ji, Y. B., & Lee, C. (2010). Data envelopment analysis. The Stata Journal, 10(2), 267-280.

Kaps, C. & de Koster, M.B.M. (2019). Warehouse Data NL & BE, 2012 and 2017. Erasmus University, The Netherlands. https://doi.org/10.25397/eur.8279426

Kerstens, K., & Van de Woestyne, I. (2021). Cost functions are nonconvex in the outputs when the technology is nonconvex: convexification is not harmless. Annals of Operations Research, 305(1), 81-106.

Kerstens, K., O'Donnell, C., & Van de Woestyne, I. (2019). Metatechnology frontier and convexity: A restatement. European Journal of Operational Research, 275(2), 780-792.

Kerstens, K., & Van de Woestyne, I. (2021). Cost functions are nonconvex in the outputs when the technology is nonconvex: convexification is not harmless. Annals of Operations Research, 305(1), 81-106.

Khezrimotlagh, D., Zhu, J., Cook, W. D., & Toloo, M. (2019). Data envelopment analysis and big data. European Journal of Operational Research, 274(3), 1047-1054.

Koopmans, T. C. (1951). An analysis of production as an efficient combination of activities. Activity analysis of production and allocation.

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26). New York: Springer.

Kuosmanen, T., & Johnson, A. (2017). Modeling joint production of multiple outputs in StoNED: Directional distance function approach. European Journal of Operational Research, 262(2), 792-801.

Kuosmanen, T., & Johnson, A. L. (2010). Data envelopment analysis as nonparametric least-squares regression. Operations Research, 58(1), 149-160.

Landete, M., Monge, J.F. & Ruiz, J.F. (2017). Robust DEA efficiency scores: A probabilistic/combinatorial approach, Expert Systems with Applications, 86: 145–154.

LeBlanc, M., & Tibshirani, R. (1996). Combining estimates in regression and classification. Journal of the American Statistical Association, 91(436), 1641-1650.

Lee, C. Y., & Cai, J. Y. (2020). LASSO variable selection in data envelopment analysis with small datasets. Omega, 91, 102019.

Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2014). Semi-supervised learning for multi-target regression. In International workshop on new frontiers in mining complex patterns, 3-18. Springer, Cham.

Louppe, G. (2014). Understanding random forests: From theory to practice (Doctoral dissertation, University of Liège Belgium).

Louppe, G., & Geurts, P. (2012). Ensembles on random patches. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 346-361). Springer, Berlin, Heidelberg.

Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In Advances in neural information processing systems (pp. 431-439).

Lovell, C. K., & Pastor, J. T. (1995). Units invariant and translation invariant DEA models. Operations research letters, 18(3), 147-151.

Lozano, S., & Khezri, S. (2019). Network DEA smallest improvement approach. Omega, 102140.

Mangasarian, O. L. (1994), Nonlinear Programming, vol. 10 of Classics in Applied Mathematics, SIAM, Philadelphia, Pa, USA.

McKenzie T (2018). snfa: Smooth Non-Parametric Frontier Analysis. R package version 0.0.1, URL https://CRAN.R-project.org/package=snfa.

Meeusen, W., & van Den Broeck, J. (1977). Efficiency estimation from Cobb-Douglas production functions with composed error. International economic review, 435-444.

Misiunas, N., Oztekin, A., Chen, Y., & Chandra, K. (2016). DEANN: A healthcare analytic methodology of data envelopment analysis and artificial neural networks for the prediction of organ recipient functional status. Omega, 58, 46-54.

Mojirsheibani, M. (1997). A consistent combined classification rule. Statistics & Probability Letters, 36(1), 43-47.

Mojirsheibani, M. (1999). Combining classifiers via discretization. Journal of the American Statistical Association, 94(446), 600-609.

Nataraja, N. R., & Johnson, A. L. (2011). Guidelines for using variable selection techniques in data envelopment analysis. European Journal of Operational Research, 215(3), 662-669.

Nunamaker, T. R. (1985). Using data envelopment analysis to measure the efficiency of non-profit organizations: A critical evaluation. Managerial and decision Economics, 6(1), 50-58.

O'Donnell, C. J. (2018). Productivity and Efficiency Analysis. Springer Singapore.

OECD. (2019). PISA 2018 assessment and analytical framework. OECD publishing.

Oh, D. H., & Suh, D. (2013). Nonparaeff: Nonparametric methods for measuring efficiency and productivity. R package version 0.5-8.

Olesen, O. B., & Ruggiero, J. (2022). The hinging hyperplanes: An alternative nonparametric representation of a production function. European Journal of Operational Research, 296(1), 254-266.

Olesen, O. B., & Ruggiero, J. (2022). The hinging hyperplanes: An alternative nonparametric representation of a production function. European Journal of Operational Research, 296(1), 254-266.

Olfati, M., Yuan, W., Khan, A., & Nasseri, S. H. (2020). A New Approach to Solve Fuzzy Data Envelopment Analysis Model Based on Uncertainty. IEEE Access, 8, 167300-167307.

Orea, L., & Zofío, J. L. (2019). Common methodological choices in nonparametric and parametric analyses of firms' performance. In The Palgrave Handbook of Economic Performance Analysis (pp. 419-484). Palgrave Macmillan, Cham.

Parmeter, C. F., Sun, K., Henderson, D. J., & Kumbhakar, S. C. (2014). Regression and inference under smoothness restrictions. Journal of Productivity Analysis, 41, 111-129.

Parmeter, C. F., & Racine, J. S. (2013). Smooth constrained frontier analysis. In Recent advances and future directions in causality, prediction, and specification analysis (pp. 463-488). Springer, New York, NY.

Pastor, J. T., Ruiz, J. L., & Sirvent, I. (2002). A statistical test for nested radial DEA models. Operations Research, 50(4), 728-735.

Pearl, J. (1984). Intelligent search strategies for computer problem solving. Addision Wesley.

Perelman, S., & Santín, D. (2009). How to generate regularly behaved production data? A Monte Carlo experimentation on DEA scale efficiency measurement. European Journal of Operational Research, 199(1), 303-310.

R Core Team (2021). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Raab, R. L., & Lichty, R. W. (2002). Identifying subareas that comprise a greater metropolitan area: The criterion of county relative efficiency. Journal of Regional Science, 42(3), 579-594.

Rebai, S., Yahia, F. B., & Essid, H. (2019). A graphically based machine learning approach to predict secondary schools' performance in Tunisia. Socio-Economic Planning Sciences, 100724.

Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. IEEE transactions on pattern analysis and machine intelligence, 28(10), 1619-1630.

Ruggiero, J. (2005). Impact assessment of input omission on DEA. International Journal of Information Technology & Decision Making, 4(03), 359-368.

Shah, A. A., Wu, D. D., Korotkov, V., & Jabeen, G. (2019). Do commercial banks benefited from the belt and road initiative? A three-stage DEA-tobit-NN analysis. IEEE Access, 7, 37936-37949.

Shen, W. F., Zhang, D. Q., Liu, W. B., & Yang, G. L. (2016). Increasing discrimination of DEA evaluation by utilizing distances to anti-efficient frontiers. Computers & Operations Research, 75, 163-173.

Shephard, R. W. (1953). Cost and production functions. Princeton University Press.

Simar, L., & Wilson, P. W. (1998). Sensitivity analysis of efficiency scores: How to bootstrap in non-parametric frontier models. Management science, 44(1), 49-61.

Simar, L., & Wilson, P. W. (2000a). A general methodology for bootstrapping in non-parametric frontier models. Journal of applied statistics, 27(6), 779-802.

Simar, L., & Wilson, P. W. (2000b). Statistical inference in non-parametric frontier models: The state of the art. Journal of productivity analysis, 13(1), 49-78.

Social Progress Index (2020). Social Progress Index 2018. Recovered from https://www.socialprogress.org/.

StataCorp (2021). Stata Statistical Software: Release 14. StataCorp LP, College Station. URL http://www.stata.com/.

Stojanova, D., Ceci, M., Appice, A., & Džeroski, S. (2012). Network regression with predictive clustering trees. Data Mining and Knowledge Discovery, 25(2), 378-413.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. SIAM journal on computing, 1(2), 146-160.

Tavakoli, I. M., & Mostafaee, A. (2019). Free disposal hull efficiency scores of units with network structures. European Journal of Operational Research, 277(3), 1027-1036.

The MathWorks Inc (2015). MATLAB – The Language of Technical Computing, Version R 2021. Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Tsionas, M. G. (2022). Efficiency estimation using probabilistic regression trees with an application to Chilean manufacturing industries. International Journal of Production Economics, 108492.

Tortosa-Ausina, E., Armero, C., Conesa, D., & Grifell-Tatjé, E. (2012). Bootstrapping profit change: An application to Spanish banks. Computers & Operations Research, 39(8), 1857-1871.

Ueda, T., & Hoshiai, Y. (1997). Application of principal component analysis for parsimonious summarization of DEA inputs and/or outputs. Journal of the Operations Research Society of Japan, 40(4), 466-478.

Vapnik, V. (2000). The nature of statistical learning theory. Springer science & business media.

Valero-Carreras, D., Aparicio, J., & Guerrero, N. M. (2021). Support vector frontiers: A new approach for estimating production functions through support vector machines. Omega, 104, 102490.

Valero-Carreras, D., Aparicio, J., & Guerrero, N. M. (2022). Multi-output Support Vector Frontiers. Computers & Operations Research, 143, 105765.

Yang, C. W., & Chen, P. S. (2020). Applying Data Envelopment Analysis to Evaluate the Operation Performance of Taiwan's TFT-LCD Industry After Post-Global Financial Crisis: A Longitudinal Study. IEEE Access, 8, 145171-145181.

Zhu, J. (2019). DEA under big data: data enabled analytics and network data envelopment analysis. Annals of Operations Research, 1-23.

Zhu, Q., Wu, J., & Song, M. (2018). Efficiency evaluation based on data envelopment analysis in the big data context. Computers & Operations Research, 98, 291-300.

CHAPTER 7: CONCLUSIONS

In this thesis, a bridge between production frontier analysis and machine learning has been established for the first time. Until now, these two fields have grown in parallel with few points of contact. However, they clearly present certain connections, and the new trend in operations research linked to big data, data science, and machine learning encourages efficiency analytics researchers to tap into the field of data analytics (see, for example, Khezrimotlagh et al., 2019). In our case, this has meant the introduction of a new method for estimating production functions using trees, called Efficiency Analysis Trees (EAT). The new technique is clearly inspired by the famous CART (Classification and Regression Trees) of Breiman et al. (1984). In EAT, minimization of the mean square error is chosen as the criterion for recursively generating binary partitions of the data (the learning sample) until no further meaningful partitioning is possible or a stopping rule is satisfied. The graphical result of this process is a tree that starts at the root node, develops through intermediate nodes, and ends at the terminal nodes (leaves). Unlike CART, EAT is capable of estimating production functions, as it has been defined to capture peak trends rather than mean trends, as well as ensuring that free disposability is satisfied.

Through the new approach, the space of the input variables is partitioned into a sequence of binary partitions at terminal nodes. At each terminal node, the expected response value (the output) is constant. Therefore, graphically, the predictor looks like a step function and has similarities and differences with respect to FDH when building a deep tree. In particular, it is shown that, in the case of the production of an input and an output, FDH and EAT generate the same estimate of the efficiency frontier. However, while FDH suffers from an overfitting problem, the EAT technique can be improved by pruning and

cross-validation (Breiman et al., 1984), solving the initial problems. This connection between the free disposability assumption and efficiency analysis trees further contributes to the development of the data science foundation for FDH, opening new paths to adapt and integrate machine learning techniques to the world of efficiency analysis.

Historically speaking, before the introduction of CART by Breiman et al. (1984), there were already other tree generation techniques for regression, such as AID (Automatic Interaction Detection) by Morgan and Sonquist (1963). Thus, the use of the regression trees of Breiman et al. (1984) was by no means original, except on one crucial point: avoiding the overfitting problem through an objective data-driven process. As Breiman et al. (1984, p. 216) claim: "*The main difference between AID and CART lies in the process of pruning and estimation, that is, in the process of 'growing an honest tree."* In the same way, we wish to highlight a certain parallelism between AID and CART and FDH and EAT, the latter being defined for the estimation of production functions. And, in a sense, EAT could be interpreted as a "pruned" FDH or an Out-Of-Sample predictor of the FDH type, overcoming its problem of overfitting the data if it is to estimate the true theoretical frontier. Decision trees could be an interesting way of looking at data in production frontier analysis. EAT should not be used to the exclusion of other frontier methods. However, we believe that EAT adds a new flexible and interesting non-parametric tool to the data analyst's toolbox in the realm of estimating production functions.

In Chapter 4 Section 4.1., the performance of the new approach was investigated using the simulations. The results indicated that EAT (pruned) outperforms FDH with respect to several traditional error measures such as mean square error (MSE), bias, and the absolute value of bias. Regarding the MSE, it is observed that the improvements ranged between 13% and 70% in the simulations. Furthermore, it was observed that the larger the sample size, the greater the reduction in MSE.

The new technique offers some additional advantages but also has some disadvantages compared to FDH. Regarding the additional advantages, it is worth mentioning that the EAT methodology allows the graphical representation of the estimated production function through trees, even in the case of a high number of inputs. This could be an

interesting feature from a data visualization point of view. Moreover, EAT also allows determining a ranking of the inputs with respect to the importance of those variables in the prediction of the output variable (the response variable). These two properties contrast positively with the standard FDH technique. Regarding the weaknesses of EAT, a clear limitation is related to the computational intensity necessary for its application in practice.

On the other hand, Random Forest is an effective machine learning technique for predicting one or a set of response variables from a set of predictors. Accurate estimators are obtained when randomness is included in the learning algorithm, as far as the data and the predictors are concerned, (Breiman, 2001). Compared to CART (Classification and Regression Trees), where a single tree is built, Random Forest aggregates hundreds of random trees across sets, reducing the generalization error by decreasing the variance term, keeping the bias approximately constant. In particular, Random Forest overcomes certain drawbacks of individual decision trees: (1) individual trees do not typically have a high level of prediction accuracy, and (2) trees can be very poorly robust, that is, a small change in the data can cause a large change in the final structure of the fitted tree. Random Forest proves that combining random trees actually achieves better performance than a single non-random tree (James et al., 2013).

In this thesis, in Chapter 4 Section 4.2., the Random Forest technique was adapted to the field of production frontier estimation, adjusting the standard Random Forest algorithm, based on CART, to deal with EAT. For this, certain new rules were introduced to select the hyperparameter associated with the number of input variables to be considered in each division, based on the well-known general rules of the Data Envelopment Analysis literature. The analyzes showed that this Random Forest adaptation to efficiency measurement outperforms both FDH and EAT with respect to bias and mean square error. These facts give validity to the new technique (RF+EAT) for estimating the technical efficiency of a set of Decision Making Units (DMUs) in a mult-input/multi-output setting. In addition, a ranking of the inputs was provided with respect to the importance of these variables in predicting the outputs, which was based on the Out-Of-Sample estimate of the generalization error of the approach. Furthermore, it was shown, through several

numerical instances taken from the literature, that RF+EAT is a possible solution to the curse of dimensionality that occurs in standard FDH.

From the computational point of view raised in Chapter 4 Section 4.3., the idea behind the heuristic approach applied in the standard EAT algorithm and the potential of backtracking combine to increase the quality of the estimation of production functions based on EAT (Chapter 4 Section 4.1.) in a feasible time. To do this, two parameters were established as a stopping condition to change the development strategy of the decision tree. The heuristic growth parameter is determined by *numStopH* and the backtracking parameter by *numStopB*. The simulation experiments carried out show that backtracking obtains greater precision than heuristics, although with a very high computational cost. On the other hand, in the approaches in which both techniques are combined, it can be observed that in the first of them (based first on heuristics and then on backtracking) results similar to those of the heuristic algorithm are obtained; in the second one (based first on backtracking and then on heuristics), the results are closer to those obtained in backtracking, although in a more reasonable time. In this case, it was observed that the improvement percentage with respect to backtracking accuracy approaches 94% when *numStopB* takes the smallest value, and when it increases, the improvement percentage decreases from 0.7% to 0%. According to the computational results, the combination of the heuristic and backtracking algorithm, in particular, the one in which the growth of the tree starts with the heuristic and ends with backtracking, has achieved an accuracy similar to that of backtracking and within a reasonable computational time. In addition, we suggest how to tune some key parameters related to the performance of the algorithm, depending on the size of the problem and the computation time available.

Finally, the *eat* package (Chapter 4 Section 4.4. of this thesis) allows the estimation of production frontiers in microeconomics and engineering using EAT and RF+EAT trees on the R platform. In the *eat* package, several functions were implemented to determine the best model, through a pruning process based on cross-validation, plot the results, calculate a ranking of importance of the inputs and compare the efficiency scores estimated by EAT with respect to the results achieved by standard approaches, i.e. FDH and DEA. By this we mean the input- and output-oriented radial models, the input- and

output-oriented Russell measures, the directional distance function, and the weighted additive model. Throughout this chapter of the thesis, it was also shown how to organize the data, use the available functions and interpret the results. In particular, to illustrate the different functions implemented in the package, they have all been applied on a common empirical example so that the results can be easily compared. In this way, it was shown that the eat package is a valid autonomous R package for the measurement of technical efficiency through decision trees. Lastly, since the code is available free of charge in an open source repository, users will benefit from community collaboration and review. Users can review and modify the code to suit their own needs and extend it with new definitions.

It concludes by mentioning several interesting lines for future research. The first is the possibility of extending the RF+EAT technique to deal with alternative technical efficiency measures, such as the directional distance function (Chamber et al., 1998). Another line of research, from a more computational perspective, would be the development of the hyperparameters that control the EAT and RF+EAT algorithms. Although in standard EAT and Random Forest, most of the usual default hyperparameters work well in practice, different settings could be tested in the context of production frontier estimation (size of nodes, number of trees, number of sampled inputs). A third obvious line of research that should be followed is the application of the new approach to real databases in different empirical contexts, thus verifying the validity of the technique in practice. Finally, the new technique could also be applied to measure efficiency in other contexts, such as economic inefficiency (see for example, Tortosa-Ausina, et. al, 2012, Pastor et al., 2022) or the change in productivity over time (see Balk et al., 2020). Regarding the future lines that arise from a computational point of view, we go on to mention several of interest. The first is the possibility of extending the EAT technique in such a way that splits produce more than two child nodes in each iteration of the heuristic tree growth algorithm. In another line of research, different adjustments could be tested in the context of estimating the production frontier, for example, by playing with the size of the nodes or the number of input and output samples. A third line of research is the possibility of extending the EAT technique by trying alternative technical measures of efficiency to determine the accuracy of the decision tree.

Finally, indicate that some progress has already been made with respect to the natural extension of EAT so that it can be used to estimate convex production possibility sets, thus competing against DEA (see, in this regard, Aparicio et al., 2021).

CAPÍTULO 7: CONCLUSIONES

En esta tesis, se ha establecido por primera vez un puente entre el análisis de las fronteras de producción y el aprendizaje automático. Hasta ahora, estos dos campos han crecido en paralelo con pocos puntos de contacto. Sin embargo, presentan claramente ciertas conexiones y la nueva tendencia en la investigación operativa vinculada a grandes volúmenes de datos, la ciencia de datos y el aprendizaje automático anima a los investigadores del análisis de la eficiencia a aprovechar el campo de la analítica de datos (véase, por ejemplo, Khezrimotlagh et al., 2019). En nuestro caso, esto ha significado la introducción de un nuevo método para estimar las funciones de producción mediante el uso de árboles, llamado Árboles de Análisis de Eficiencia (*Efficiency Analysis Trees*, en inglés) (EAT). La nueva técnica está claramente inspirada en el famoso CART (*Classification and Regression Trees*) de Breiman et al. (1984). En EAT, se elige la minimización del error cuadrático medio como criterio para generar recursivamente particiones binarias de los datos (la muestra de aprendizaje) hasta que no sea posible ninguna otra división significativa o se satisfaga una regla de parada. El resultado gráfico de este proceso es un árbol que comienza en el nodo raíz, se desarrolla a través de los nodos intermedios y termina en los nodos terminales (hojas). A diferencia de CART, EAT es capaz de estimar las funciones de producción, ya que se ha definido para captar las tendencias máximas en lugar de las tendencias medias y para garantizar la satisfacción de la libre disponibilidad.

A través del nuevo enfoque, el espacio de las variables de entrada se divide en una secuencia de divisiones binarias en nodos terminales. En cada nodo terminal, el valor de

respuesta previsto (el output) es constante. Por lo tanto, gráficamente, el predictor parece una función escalonada y presenta similitudes y diferencias con respecto a FDH cuando se construye un árbol profundo. En particular, se muestra que, en el caso de la producción de un input y un output, FDH y EAT generan la misma estimación de la frontera de eficiencia. Sin embargo, mientras que FDH sufre de un problema de sobreajuste, la técnica EAT puede mejorarse mediante la poda y la validación cruzada (Breiman et al., 1984), resolviendo los problemas iniciales. Esta conexión entre el supuesto de libre disponibilidad y los árboles de análisis de eficiencia contribuye aún más al desarrollo de la base de la ciencia de datos para FDH, abriendo nuevos caminos para adaptar e integrar las técnicas de aprendizaje automático al mundo del análisis de eficiencia.

Históricamente hablando, antes de la introducción de CART por Breiman et al. (1984), ya existían otras técnicas de generación de árboles para la regresión, como AID (Automatic Interaction Detection) de Morgan y Sonquist (1963). Así pues, la utilización de los árboles de regresión de Breiman et al. (1984) no fue en absoluto original, excepto en un punto crucial: evitar el problema del sobreajuste mediante un proceso objetivo basado en datos. Como afirman Breiman et al. (1984, pág. 216): "*La principal diferencia entre AID y CART radica en el proceso de poda y estimación, es decir, en el proceso de 'hacer crecer un árbol honesto'*". De la misma manera, se quiere destacar un cierto paralelismo entre AID y CART y FDH y EAT, este último definido para la estimación de las funciones de producción. Y, en cierto sentido, EAT podría interpretarse como un FDH "podado" o un predictor fuera de la muestra (*Out-Of-Sample*, en inglés) del tipo FDH, superando su problema de sobreajuste de los datos si se trata de estimar la verdadera frontera teórica. Los árboles de decisión podrían ser una forma interesante de ver los datos en el análisis de las fronteras de producción. EAT no debería utilizarse con exclusión de otros métodos de frontera. Sin embargo, creemos que EAT añade una nueva herramienta no paramétrica flexible e interesante a la caja de herramientas del analista de datos en el ámbito de la estimación de funciones de producción.

En el Capítulo 4 Sección 4.1., se investigó el rendimiento del nuevo enfoque mediante las simulaciones. Los resultados indicaron que EAT (podado) supera a FDH con respecto a varias medidas de error tradicionales como el error cuadrático medio (MSE), el sesgo y

el sesgo basado en el valor absoluto. Con respecto al MSE, se observa que las mejoras determinadas oscilaron entre el 13% y el 70% en las simulaciones. Además, se observó que cuanto mayor era el tamaño de la muestra, mayor era la reducción del MSE.

La nueva técnica presenta tanto ventajas como inconvenientes adicionales en comparación con FDH. En cuanto a las ventajas adicionales, cabe mencionar que la metodología EAT permite la representación gráfica de la función de producción estimada a través de árboles, incluso en el caso de que se trate de un número elevado de inputs. Esta podría ser una característica interesante desde el punto de vista de la visualización de los datos. Además, EAT también permite determinar una clasificación de los inputs con respecto a la importancia de esas variables en la predicción de la variable de salida (la variable respuesta). Estas dos propiedades contrastan positivamente con la técnica estándar de FDH. En cuanto a las debilidades de EAT, una clara limitación se encuentra relacionada con la intensidad de cómputo necesario para su aplicación en la práctica.

Por otra parte, Random Forest es una técnica efectiva de aprendizaje automático para predecir una o un conjunto de variables respuesta a partir de un conjunto de predictores. Incluyendo aleatoriedad en el algoritmo de aprendizaje, en lo que respecta a los datos y los predictores, se obtienen estimadores precisos (Breiman, 2001). En comparación con CART (*Classification and Regression Trees*), donde se construye un árbol individual, Random Forest agrega cientos de árboles aleatorios a través de conjuntos reduciendo el error de generalización al disminuir el término de varianza, manteniendo aproximadamente constante el sesgo. En particular, Random Forest supera ciertos inconvenientes de los árboles de decisión individuales: (1) los árboles individuales no suelen presentar un alto nivel de exactitud de predicción y (2) los árboles pueden ser muy poco robustos, es decir, un pequeño cambio en los datos puede causar un gran cambio en la estructura final del árbol ajustado. Random Forest demuestra que la combinación de árboles aleatorios realmente logra un mejor rendimiento que un solo árbol no aleatorio (James et al., 2013).

En esta tesis, en el Capítulo 4 Sección 4.2., se adaptó la técnica de Random Forest al campo de estimación de las fronteras de producción, ajustando el algoritmo estándar de

Random Forest, basado en CART, para tratar con EAT. Para ello, se introdujeron ciertas reglas nuevas para seleccionar el hiperparámetro asociado con el número de variables de entrada a considerar en cada división, basadas en las conocidas reglas generales de la literatura del Análisis Envolvente de Datos. Los análisis demostraron que esta adaptación del Random Forest a la medición de la eficiencia supera tanto a FDH como a EAT con respecto al sesgo y al error cuadrático medio. Estos hechos dan validez a la nueva técnica (RF+EAT) para estimar la eficiencia técnica de un conjunto de Unidades de Toma de Decisión (DMUs) en un entorno de múltiples inputs y outputs. Además, se proporcionó una clasificación de los inputs con respecto a la importancia de estas variables para predecir los outputs, que se basó en la estimación del *Out-Of-Sample* (fuera de la muestra) del error de generalización del enfoque. Además, se mostró, a través de varias instancias numéricas tomadas de la literatura, que RF+EAT es una posible solución de la maldición de la dimensionalidad que se presenta en el FDH estándar.

Desde el punto de vista computacional planteado en el Capítulo 4 Sección 4.3., la idea que hay detrás del enfoque heurístico aplicado en el algoritmo EAT estándar y la potencialidad del *backtracking* se combinan para incrementar la calidad de la estimación de las funciones de producción basadas en EAT (Capítulo 4 Sección 4.1.) en un tiempo factible. Para ello, se establecieron dos parámetros como condición de parada para cambiar la estrategia de desarrollo del árbol de decisiones. El parámetro de crecimiento heurístico está determinado por *numStopH* y el parámetro de *backtracking* por *numStopB*. Los experimentos de simulación realizados, demuestran que el *backtracking* obtiene una mayor precisión que el de la heurística, aunque con un coste computacional muy elevado. Por otra parte, en los enfoques en los que se unen ambas técnicas, se puede observar que en la primera de ellas (basada primero en la heurística y luego en el *backtracking*) se obtienen resultados similares a los del algoritmo heurístico; en la segunda de ellas (basada primero en el *backtracking* y luego en la heurística), los resultados se acercan más a los obtenidos en el *backtracking*, aunque en un tiempo más razonable. En este caso, se observó que el porcentaje de mejora con respecto a la precisión del *backtracking* se acerca al 94% cuando el *numStopB* toma el valor más pequeño, y cuando éste aumenta, el

porcentaje de mejora disminuye del 0.7% al 0%. De acuerdo con los resultados computacionales, la combinación del algoritmo heurístico y de *backtracking*, en particular, aquel en el que el crecimiento del árbol comienza con la heurística y termina con el *backtracking*, ha logrado una precisión similar a la del *backtracking* y dentro de un tiempo computacional razonable. Además, sugerimos cómo ajustar algunos parámetros clave relacionados con el rendimiento del algoritmo, en función del tamaño del problema y el tiempo de cálculo disponible.

Finalmente, el paquete *eat* (Capítulo 4 Sección 4.4. de esta tesis) permite la estimación de fronteras de producción en microeconomía e ingeniería mediante los árboles EAT y los RF+EAT en la plataforma de R. En el paquete *eat* se implementó varias funciones para determinar el mejor modelo, mediante un proceso de poda basado en la validación cruzada, graficar los resultados, calcular un ranking de importancia de los inputs y comparar las puntuaciones de eficiencia estimadas por EAT con respecto a los enfoques estándar, es decir, FDH y DEA. Con esto nos referimos a los modelos radiales orientados al input y al output, a las medidas de Russell orientadas al input y al output, a la función de distancia direccional y al modelo aditivo ponderado. A lo largo de este capítulo de la tesis, también se mostró cómo organizar los datos, utilizar las funciones disponibles e interpretar los resultados. En particular, para ilustrar las diferentes funciones implementadas en el paquete, se aplicaron todas sobre un ejemplo empírico común para que los resultados puedan ser fácilmente comparados. De este modo, se mostró que el paquete *eat* es un paquete R autónomo válido para la medición de la eficiencia técnica mediante los árboles de decisión. Por último, dado que el código está disponible gratuitamente en un repositorio de código abierto, los usuarios se beneficiarán de la colaboración y la revisión de la comunidad. Los usuarios pueden revisar y modificar el código para adaptarlo a sus propias necesidades y ampliarlo con nuevas definiciones.

Se termina mencionando varias líneas que plantean interesantes vías para seguir investigando. La primera es la posibilidad de ampliar la técnica RF+EAT para tratar medidas técnicas de eficiencia alternativas como, por ejemplo, la función distancia direccional (Chamber et al., 1998). Otra línea de investigación, desde una perspectiva más computacional, sería la puesta a punto de los hiperparámetros que controlan los algoritmos EAT y RF+EAT. Aunque en EAT y Random Forest estándar, la mayoría de

los hiperparámetros habituales por defecto funcionan bien en la práctica, se podría comprobar diferentes ajustes en el contexto de la estimación de las fronteras de producción (tamaño de los nodos, número de árboles, número de inputs muestreadas). Una tercera línea de investigación evidente que debe seguirse es la aplicación del nuevo enfoque a bases de datos reales en diferentes contextos empíricos, comprobando así la validez de la técnica en la práctica. Por último, la nueva técnica podría aplicarse también para medir la eficiencia en otros contextos, como la ineficiencia económica (véase, por ejemplo, Tortosa-Ausina, et. al, 2012, Pastor et al., 2022) o el cambio de la productividad a lo largo del tiempo (véase Balk et al., 2020). Respecto a las líneas futuras que se plantean desde un punto de vista computacional, se mencionan varias interesantes a continuación. La primera es la posibilidad de extender la técnica EAT de tal manera que las divisiones produzcan más de dos nodos hijos en cada iteración del algoritmo de crecimiento del árbol heurístico. En otra línea de investigación, se podría comprobar diferentes ajustes en el contexto de la estimación de la frontera de producción, por ejemplo, jugando con el tamaño de los nodos o el número de muestras de input y output. Una tercera línea de investigación es la posibilidad de ampliar la técnica EAT para tratar medidas técnicas de eficiencia alternativas para determinar la precisión del árbol de decisión.

Por último, indicar que ya se han realizado ciertos avances con respecto a la extensión natural de EAT para que pueda ser utilizado con el objetivo de estimar conjuntos de posibilidades de producción convexos, compitiendo así frente a DEA (véase, a este respecto, Aparicio et al., 2021).

AGRADECIMIENTOS

Esta tesis ha sido posible gracias a la ayuda FPU17/05365 financiada por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033 y por el "Fondo Social Europeo Invierte en tu futuro".

Esta publicación es parte del proyecto de I+D+i PID2019-105952GB-I00, financiado por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033/.

Además, esta publicación también es parte del proyecto de I+D+i MTM2016-79765-P, financiado por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033/ y por "FEDER Una manera de hacer Europa".