Facultad de Ciencias

Trabajo Fin de Grado

Uso de técnicas de inteligencia artificial para identificar galaxias activas

Autor:
Carlos Pascual Borruel

Directores:
Dr. Francisco J. Carrera Troyano
Dr. Diego Tuccillo

Octubre 2023

Resumen

En los próximos años se publicarán los datos recopilados por las misiones Euclid y LSST, que son los proyectos de cartografiado de galaxias, en el óptico e infrarrojo cercano, más importantes de la actualidad. En este trabajo nos ha parecido interesante comprobar el potencial de estos grandes muestreos para facilitar la identificación de Núcleos Galácticos Activos (AGN) entre la enorme cantidad de galaxias observadas. Para ello, se ha decidido romper con los métodos tradicionales de clasificación de galaxias, aprovechando la potencia de las novedosas técnicas de Machine Learning en tareas de clasificación.

Concretamente se han implementado dos algoritmos de aprendizaje supervisado, Decision Tree y Random Forest, en datos simulados de los cartografiados mencionados, los cuales consisten en dos catálogos de galaxias de distinta profundidad: DEEP y WIDE. Para la preparación de estos datos, se utilizaron herramientas profesionales de manipulación de tablas astronómicas, como TOPCAT y STILTS, con las que se seleccionaron aquellas fuentes pertenecientes a las áreas comunes de ambos cartografiados y se filtraron las que no iban a ser detectadas por los observatorios reales. Además, se utilizaron distintas librerías de python para etiquetar las muestras y seleccionar únicamente las magnitudes de los filtros de cada proyecto.

Finalmente, se ha observado que las magnitudes de LSST deben ser descartadas, ya que al no incluir ruido presentan unos patrones internos que evidencian la clasificación de fuentes, de modo que se tuvo que continuar solo con las magnitudes de Euclid. Esta pérdida de información ha podido provocar un descenso importante en el rendimientos de los modelos. No obstante, se ha descubierto que agregando columnas de colores a las magnitudes de Euclid, mejoran notablemente los resultados de la clasificación. Por otra parte, se ha observado que los algoritmos de Random Forest no ofrecen mejoras significativas en el rendimiento de las métricas respecto a los modelos de Decision Tree, destacándose la profunidad máxima de los árboles como el hiperparámetros más influyente en ambos casos. Con todo, los mejores modelos de clasificación que se han desarrollado en este trabajo alcanzan una precisión de 0.57 y una sensibilidad de 0.70 para el conjunto DEEP, y una precisión de 0.75 y una sensibilidad de 0.84 para el conjunto WIDE. De esta forma, se abre un camino para mejorar la eficiencia y la precisión en la detección de AGNs.

Palabras clave: Galaxias, AGN, Machine Learning, Clasificación, Decision Tree, Random Forest.

Abstract

In the coming years, data collected by the Euclid and LSST missions, which are the most significant current galaxy mapping projects in the optical and near-infrared spectrum, will be published. This work aims to explore the potential of these extensive surveys to facilitate the identification of Active Galactic Nuclei (AGN) among the vast number of observed galaxies. To achieve this, we have decided to move away from traditional galaxy classification methods, harnessing the power of innovative Machine Learning techniques in classification tasks.

Specifically, two supervised learning algorithms, Decision Tree and Random Forest, have been implemented on simulated data from the mentioned surveys, consisting of two galaxy catalogs of different depths: DEEP and WIDE. For data preparation, professional astronomical table manipulation tools such as TOPCAT and STILTS were used. These tools helped in selecting sources belonging to the common areas of both surveys and filtering out those that would not be detected by real observatories. Furthermore, various Python libraries were utilized for labeling the samples and selecting only the magnitudes of the filters from each project.

Ultimately, it was observed that the magnitudes from LSST should be discarded, as they do not include noise and present internal patterns that bias the source classification. This required the continuation with only Euclid magnitudes. This loss of information may have led to a significant decrease in the models' performance. However, it was discovered that adding color columns to the Euclid magnitudes notably improves the classification results. On the other hand, it was observed that Random Forest algorithms do not provide significant improvements in metric performance compared to Decision Tree models, with the maximum depth of the trees being the most influential hyperparameter in both cases. Nevertheless, the best classification models developed in this work achieve a precision of 0.57 and a sensitivity of 0.70 for the DEEP set, and a precision of 0.75 and a sensitivity of 0.84 for the WIDE set. This opens a path to improving efficiency and precision in the detection of AGNs.

Key words: Galaxies, AGN, Machine Learning, Clasification, Decision Tree, Random Forest.

Contents

Apartado **Página**

1 Introducción

Las galaxias son vastas acumulaciones de gas, polvo interestelar y cientos de millones de estrellas que se mantienen como un sistema por la acción de la atracción gravitatoria. Además, las galaxias constituyen los bloques fundamentales con los que se estudian en cosmología las estructuras del Universo a gran escala. Algunas de estas galaxias hospedan en su centro unas regiones compactas que emiten una cantidad anormalmente alta de luz y otras formas de energía radiante. Estas regiones, que llegan incluso a eclipsar el brillo de toda su galaxia anfitriona, se las conoce como AGN (Núcleos galácticos activos, por sus siglas en inglés " *Active Galactic Nuclei*"). La intensa actividad de los AGN es alimentada por la acrección del material colindante (gas, estrellas y polvo interestelar) que cae hacia un agujero negro supermasivo ubicado en el centro de la galaxia. El material arrastrado se compacta formando un disco que, debido a las fricciones y colisiones de las partículas en su interior, se calienta a millones de grados liberando enormes cantidades de radiación electromagnética, lo que constituye el brillo característico de los AGN. [1]

Numerosas observaciones establecen que la masa de estos agujeros negros supermasivos, en su fase de Núcleo Galáctico Activo (AGN), es proporcional a la masa de su galaxia anfitriona y que, además, sus crecimientos están conectados y son paralelos en el tiempo [1]. Por lo tanto, para avanzar en el estudio de las galaxias y las estructuras del Universo a gran escala es muy importante saber qué galaxias albergan AGNs.

En astronomía, la detección masiva de galaxias se realiza mediante muestreos o cartografiados en los que se observa el cielo en una serie de bandas llamadas filtros (Ver sec 3.1). Actualmente, dos de los cartografiados de galaxias más importantes en el óptico y el infrarrojo cercano son los de las misiones Euclid y LSST, cuyos datos serán publicados de forma inminente en los próximos años. En este trabajo nos ha parecido interesante comprobar la capacidad que tendrán estos grandes cartografiados para ayudarnos a encontrar núcleos galácticos activos entre la enorme cantidad de galaxias detectadas. Como los datos todavía no están disponibles, para tal fin se han utilizado unos catálogos simulados de fuentes de Euclid y LSST creados a partir de la información disponible actualmente (Ver sec 4.3).

Hasta ahora se han estado utilizando diversas técnicas para el reconocimiento de núcleos galácticos activos (AGN). Una de ellas consiste en comparar la SED (o Distribución Espectral de Energía) observada para una fuente con una serie de plantillas SED asociadas a diferentes poblaciones galácticas, con o sin contribución AGN. De esta forma, si la SED de una fuente coincide con la SED de una población galáctica que incluye contribución AGN, esta se identifica como tal. En este trabajo, hemos querido romper con los métodos tradicionales de clasificación de galaxias a través de la implementación de novedosas técnicas de Machine Learning. Con ello se pretende abrir un camino para mejorar la eficiencia y la precisión en la detección de AGNs.

El Machine Learning (ML), o aprendizaje automático, es la rama de la inteligencia artificial cuya finalidad es conseguir que las máquinas adquieran la capacidad de aprender de su experiencia sin necesidad de la intervención humana. Sus algoritmos permiten a los ordenadores descubrir la forma correcta de ejecutar tareas sin estar expresamente programados para ello. Utilizando los datos que se les proporcionan, son capaces de identificar patrones que generan un conocimiento, el cual contribuye a la toma de mejores decisiones. De esta forma, el objetivo principal del ML es la predicción de resultados de interés basados en los datos de entrada.[11]

Los algoritmos de aprendizaje automático se enrriquecen y afinan a medida que se entrenan con nuevos datos. En función de si los datos utilizados durante el entrenamiento están etiquetados o no, es decir, si se conocen de antemano sus resultados, los algoritmos de ML se dividen en algoritmos de aprendizaje supervisado y no supervisado. El aprendizaje no supervisado se utiliza para descubrir patrones ocultos en los datos de entrada que puedan aportar conocimiento útil. Una de las tareas más comunes en el aprendizaje no supervisado es el *clustering*, que organiza los datos en grupos basados en su similitud. Por otro lado, en el aprendizaje supervisado cada muestra del conjunto de datos de entrenamiento

incluye la entrada y la salida deseada, y los algoritmos aprenden una regla general para predecir la salida adecuada a partir de la entrada. La bondad de esta regla general se pone a prueba con un conjunto de datos son reservados para su evaluación. Los algoritmos de aprendizaje supervisado se utilizan principalmente en tareas de clasificación y regresión. En concreto, en este trabajo se ha empleado la potencia de los algortimos de Decision Tree y Random Forest (Ver sec 3.2) para realizar la clasificación de galaxias, distinguiendo entre galaxias con núcleos galácticos activos (AGN) y galaxias sin ellos (NO-AGN).

2 Objetivos

En los próximos años se publicarán los datos recopilados por las misiones Euclid y LSST, que son los proyectos de cartografiado de galaxias, en el óptico e infrarrojo cercano, más importantes de la actualidad. En este trabajo nos ha parecido interesante comprobar el potencial de estos grandes muestreos para facilitar la identificación de Núcleos Galácticos Activos (AGN) entre la enorme cantidad de galaxias observadas. Para ello, se ha decidido aprovechar la potencia en tareas de clasificación de las novedosas técnicas de Machine Learning. De esta forma, el objetivo principal de este trabajo es evaluar algunas de estas técnicas en datos simulados de los proyectos mencionados para comprobar si son capaces de identificar eficientemente AGNs entre las galaxias.

3 Conceptos Básicos

3.1 flujos y magnitudes

El flujo refiere a la cantidad de energía de la luz proveniente de un objeto celeste que atraviesa una unidad de área en la Tierra por unidad de tiempo. Mientras que el flujo total ofrece una medida de toda la energía recibida, el flujo espectral mide la energía recibida en un cierto rango de longitudes de onda. En esencia, es una medida de la intensidad de luz que se puede captar con un instrumento y se mide en erg s^{-1} cm^{-2} (Hz^{-1}) o en janskys (una unidad común en radioastronomía que equivale a 10^{-23} erg s^{-1} $cm^{-2}(Hz^{-1})$).

Por otra parte, la luminosidad (o brillo) es la cantidad total de energía emitida por un objeto celeste en la unidad de tiempo. Es una propiedad intrínseca del astro, que no depende de la distancia a la que se encuentre el observador, y se mide en unidades de potencia. En astronomía, para medir el brillo aparente de un astro visto desde la Tierra se utilizan las magnitudes, una escala logarítmica inversa que se define en términos del flujo observado de la siguiente forma:

$$m = -2.5 \log_{10} \frac{F}{F_0} \tag{1}$$

El factor -2.5 se deriva de la definición histórica de magnitudes en astronomía, donde una diferencia de 5 magnitudes equivale a un factor de 100 en brillo En la práctica, las mediciones de magnitud se realizan para un rango de longitudes de onda definido por un filtro. De esta forma, en la ecuación 1, F es el flujo espectral que pasa a través de un filtro y F_0 es una constante que determina el flujo de referencia para el que se establece la magnitud 0 en el sistema de magnitudes asociado a ese filtro. Para evitar tener una constante distinta en cada filtro, se crearon las magnitudes AB que introducen un punto de referencia estándar para todos los filtros, lo cual permite comparar las diferencias de energía reales en cada uno de ellos. Ese punto de referencia , o punto cero, se ha establecido en 3631 jankys.

3.2 Algoritmos

3.2.1 Decision Tree

Los árboles de decisión son unos algoritmos de aprendizaje supervisado muy populares y utilizados en el mundo del machine learning. Esto se debe principalmente a que su estructura guarda mucha analogía con la de un árbol real (vuelto del revés), lo que los convierte en algoritmos intuitivos, sencillos de entender y fáciles de interpretar. Además, pueden trabajar con grandes cantidades de datos, no requieren de un gran poder computacional y, pese a su enorme sencillez, sus resultados son sorprendentemente precisos. En cambio, sus principales desventajas son que tiende con facilidad al *overfitting* (ocurre cuando un árbol es demasiado detallado o complejo y empieza a memorizar los datos de entrenamiento en lugar de generalizar a partir de ellos) y su alta varianza (pequeñas variaciones en los datos pueden dar lugar a un árbol completamente diferente). En este trabajo se explotará su faceta como clasificadores, en lugar de regresores, ya que se busca una predicción categórica (AGN, NO-AGN) en lugar de numérica (esperanza de vida de un país).

Un árbol de decisión parte de un conjunto de datos de entrada X que va dividiendo recursivamente en particiones binarias. Estas particiones se realizan de acuerdo a unas condiciones que impone sobre las características, o atributos, X_i del dataset hasta completar la clasificación. El árbol realiza las divisiones procurando que las muestras de las particiones finales (o hojas) sean de la misma clase y_i[9]. Esta idea queda ilustrada en la figura 1:

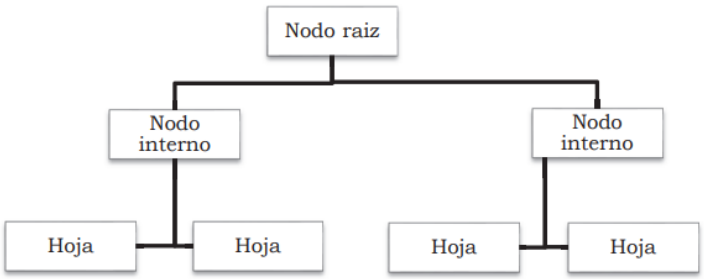


Figura 1: *Estructura de un árbol de decisión. Fuente: [11]*

Los árboles se construyen a partir de dos elementos: nodos y ramas. En cada nodo se evalúa una característica que divide al conjunto de datos en dos y así es como el árbol va creciendo. En la figura 1 se aprecian los tres tipos de nodos que componen un árbol de decisión:

-Nodo raíz: Es el nodo superior y representa a todas las muestras del conjunto de datos. En este nodo se impone la condición sobre el atributo X_i que mejor divide los datos.

-Nodo interno: Son nodos que surgen de las bifurcaciones de los nodos superiores. Al igual que los nodos raíz, tratan de imponer la condición sobre la característica X_i que mejor divide su subconjunto de datos.

-Hojas: Son los nodos ubicados en la parte inferior del árbol. Cada uno de estos nodos representa una clase predicha por el modelo. Esta predicción se determina por la clase que aparece con mayor frecuencia en las muestras de ese nodo.[10]

Así pues, el conjunto de datos de entrenamiento (datos utilizados para realizar el ajuste del modelo de clasificación) se va dividiendo desde el nodo raíz, pasando por los distintos niveles de nodos internos, hasta llegar a las hojas.

Por supuesto, la elección de los atributos que producen las mejores divisiones no es aleatoria. Para medir el grado de impureza, o de mezcla de clases que hay en un nodo, se utiliza el **índice de Gini**, G(P), o la **Entropía**, H(P), dos métricas que se definen de la siguiente manera:

$$\begin{aligned} G(P) &= 1 - \sum_{p_i \in P} p_i, \\ H(P) &= \sum_{p_i \in P} -p_i log_2 p_i \end{aligned} \tag{2}$$

donde p_i representa la proporción de muestras del nodo P pertenecientes a la clase i.

El índice de Gini mide la probabilidad de que dos muestras seleccionadas al azar pertenezcan a clases diferentes. Su valor puede variar entre 0 (nodo puro, solo contiene muestras de una clase) a 0.5 (nodo de máxima impureza, las clases de las muestras están mezcladas de manera uniforme). Por su parte, la entropía es una medida del desorden o incertidumbre de un conjunto de datos. Su rango de valores es de 0 (nodo puro) a 1 (nodo de máxima impureza). Aunque el objetivo de ambos criterios es encontrar la división que resulte en la mayor reducción de la impureza, la entropía, debido a su construcción logarítmica, es especialmente sensible a la proporción de clases. Esto quiere decir que a menudo favorece las divisiones que minimizan la presencia de casos mixtos, tendiendo a la producción de nodos puros. A diferencia de la entropía, el índice de Gini no es tan sensible a los cambios en las proporciones de las clases. Por lo tanto, es menos propenso a crear nodos pequeños altamente puros, tendiendo a crear nodos más equilibrados en términos de distribución de clases.

Por último, para controlar que el árbol no crezca de forma descontrolada y absurda, se pueden establecer algunos **criterios de parada** que ayudan a evitar el *overfitting* , como la profundidad máxima (número máximo de divisiones que puede tener el árbol) o el número mínimo de muestras por nodo (o hoja). Estos criterios de parada forman parte de los hiperparámetros del modelo, los cuales sirven para controlar el aprendizaje del modelo y tienen un gran impacto en su rendimiento. A diferencia de los parámetros del modelo, que se aprenden durante el entrenamiento, los hiperparámetros se establecen antes del aprendizaje y no varían a lo largo del proceso. [8]

3.2.2 Random Forest

Un bosque aleatorio consta de varios árboles individuales que operan como un conjunto. Cada árbol individual genera una predicción de clase, de forma que la clase más frecuente o votada se convierte en la predicción del conjunto. (ver fig 2)

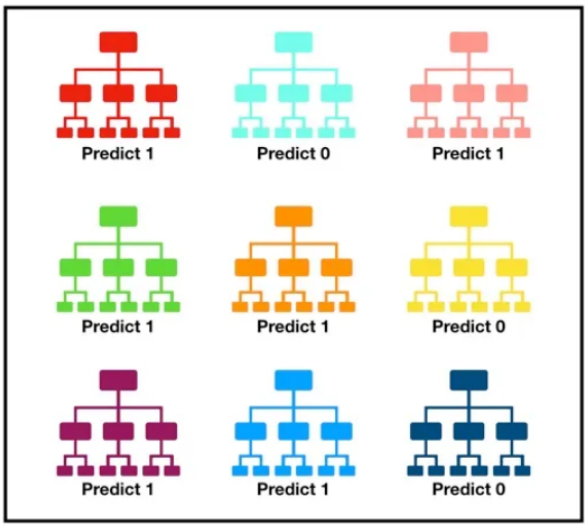


Figura 2: *Visualización de un bosque aleatorio haciendo una predicción. Seis árboles predicen la clase 1 y 3 la clase 0 y, por tanto, la clase 1 será la predicha por el bosque. Fuente: [12]*

Para beneficiarse de la predicción emitida por el conjunto de árboles, se busca que sean los más distintos entre sí, o lo que es lo mismo, que estén poco correlacionados. Si fueran idénticos o muy similares (es decir, si estuvieran muy correlacionados entre sí), sus predicciones serían esencialmente las mismas, de modo que no se obtendría ningún beneficio al combinar sus predicciones. Para solventar esto, Random Forest introduce aleatoriedad en los árboles de dos maneras. La primera se basa en la técnica de *Bagging*. Consiste en asignar a cada árbol un subconjunto aleatorio de muestras del conjunto original con reemplazo (es decir, varios árboles pueden tener la misma muestra). La segunda, por su parte, consiste en asignar a cada árbol un subconjunto aleatorio de características, aumentando la variabilidad entre los árboles del bosque.

La combinación del **Bagging** y la **selección aleatoria de características** da lugar a árboles poco correlacionados entre sí, capaces de capturar diferentes patrones o matices del conjunto total de datos. El resultado es un modelo muy potente que combina las predicciones de múltiples árboles para generar una predicción más robusta y precisa. [12]

3.3 Métricas

Las métricas se utilizan para evaluar el desempeño de los modelos de aprendizaje automático. Cada una ofrece una perspectiva distinta sobre el rendimiento del modelo en tareas de clasificación.

En primer lugar tenemos la **matriz de confusión**, una matriz 2x2 que contabiliza si las predicciones de un clasificador binario (clase positiva o negativa) se han realizado correctamente (ver fig 3).

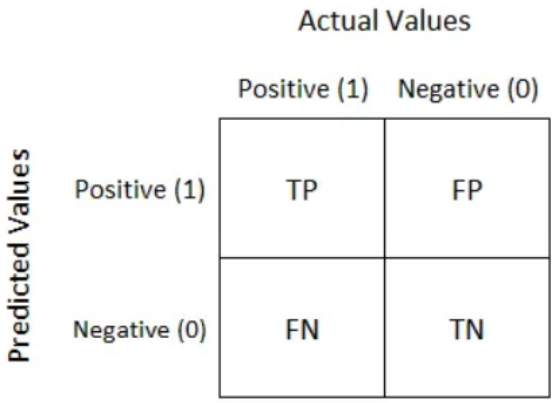


Figura 3: *Estructura de una matriz de confusión. La clase positiva se reserva normalmente para la clase que mayor interés tiene para el objetivo de la tarea (En este trabajo la clase positiva es AGN).*

.

-TP (True Positive): Ejemplos positivos clasificados correctamente.

-FP (False Positive): Ejemplos negativos clasificados incorrectamente.

-TN (True Negative): Ejemplos negativos clasificados correctamente.

-FN (False Negative): Ejemplos positivos clasificados incorrectamente.

A partir de estos 4 resultados posibles de una clasificación binaria, se definen las siguientes métricas:

Precisión: mide la proporción de predicciones positivas que fueron clasificadas correctamente. Evalúa la capacidad del modelo para evitar clasificar incorrectamente muestras negativas como positivas.

$$Pr = \frac{TP}{TP + FP}$$

Recall (o sensibilidad): mide la proporción de casos positivos reales que fueron identificados correctamente (o tasa de verdaderos positivos). Evalúa la capacidad del modelo de identificar los casos positivos.

$$Pr = \frac{TP}{TP + FN}$$

F1: Es el promedio armónico de las métricas precision y recall. Esta métrica sintetiza la capacidad del modelo para clasificar con exactitud y sin omitir casos relevantes, siendo ideal para contextos en los que ambas cualidades son importantes. Es especialmente útil cuando las clases están desbalanceadas, es decir, cuando la presencia de una clase es mucha mayor que la de otra.

$$Pr = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall}$$

FPR (False Positive Rate): Mide la proporción de casos negativos reales que fueron incorrectamente identificados como positivos.

$$FPR = \frac{FP}{FP + TN}$$

AUC (Área bajo la curva ROC): Es una medida utilizada en la evaluación de modelos de clasificación binarios para distinguir entre clases positivas y negativas, especialmente indicada para conjuntos de datos con clases balanceadas. La ROC (*Receiver Operating Characteristic*) es un gráfico que representa la sensibilidad (o tasa de verdaderos positivos) en función de la tasa de falsos positivos (FPR) a través de diferentes umbrales de clasificación. Para cada muestra el modelo predice la probabilidad que tiene de pertenecer a la clase positiva, siendo el umbral de clasificación el valor a partir del cual se decide si un caso es positivo. En la figura 4 se muestra un ejemplo de curva ROC.

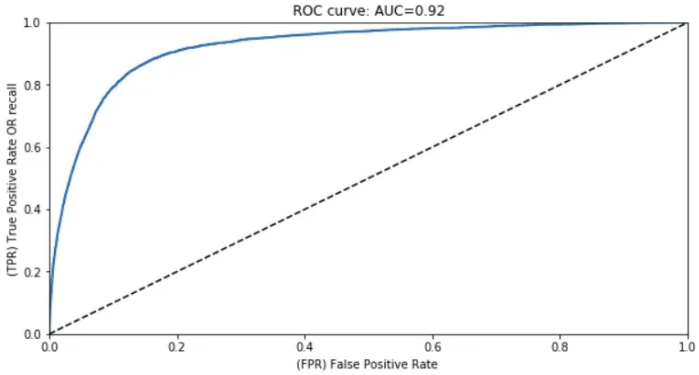


Figura 4: *Curva ROC. Esta curva permite visualizar el equilibrio entre la tasa de verdaderos positivos y la tasa de falsos positivos, Es decir, relaciona la sensibilidad (o capacidad de capturar casos positivos) del modelo con su tendencia a cometer fallos optimistas (calsificar los negativos como positivos). Fuente: [13]*

El AUC es el área bajo esta curva ROC y proporciona un número que resume la capacidad del modelo para discriminar entre clases. En las curvas ROC interesa que la curva se acerque lo máximo posible a la esquina superior izquierda (clasificador perfecto), de manera que el hecho de aumentar la sensibilidad (el recall) no haga que el modelo introduzca más falsos positivos.

Curva PR (Precision-Recall): La curva PR es un gráfico que representa la precisión frente a la sensibilidad del modelo para distintos umbrales de clasificación. Estas métricas están relacionadas entre sí de manera que si se entrena el clasificador para aumentar la precisión, el recall disminuye y viceversa. Al contrario que la curva ROC, la curva PR está especialmente indicada para datasets con clases desbalanceadas. En la figura se muestra un ejemplo de Curva PR:

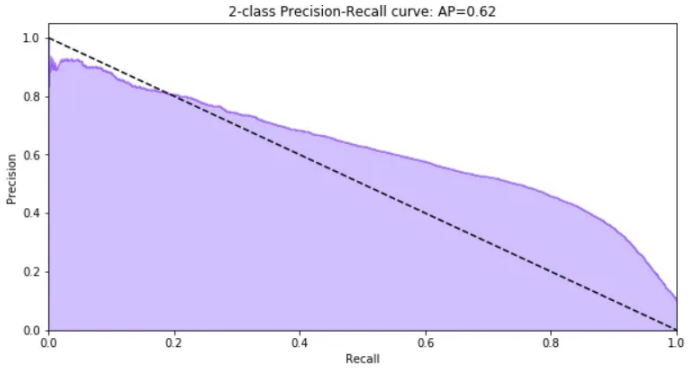


Figura 5: *Curva Precisión-Recall. Esta gráfica permite comprobar a partir de qué recall se tiene una degradación de la precisión y viceversa. Lo ideal es que la curva se acerque lo máximo posible a la esquina superior derecha (alta precisión y alto recall)). Fuente: [13]*

El AP (Average Precision) es el área bajo la curva PR y representa la media ponderada de las precisiones alcanzadas en cada umbral de decisión. Esto es particularmente útil para comprender el rendimiento general de un clasificador en términos de su capacidad para cubrir los casos positivos (sensibilidad) sin sacrificar la calidad de las predicciones positivas (precisión). [13].

4 Datos

En esta sección se describen aspectos fundamentales de los proyectos de cartografiado galáctico Euclid y LSST, incluyendo sus objetivos, el área de cobertura que abarca cada uno y la sensibilidad de sus filtros. Por último, se incluye un apartado que explica la procedencia de los catálogos simulados de galaxias y AGNs que fueron utilizados para crear los modelos de clasificación de fuentes.

4.1 Euclid

Euclid es una misión espacial de la ESA, en colaboración con la NASA, que pretende crear un cartografiado tridimensional del universo (con el tiempo como tercera dimensión) a partir del registro de miles de millones de galaxias en más de un tercio del cielo situadas hasta a 10 mil millones de años luz, lo que supone el 75% de la vida del universo. Para ello la misión cuenta con un telescopio que fue lanzado el 1 de julio de 2023 con el fin de obtener imágenes cuatro veces más nítidas que desde la tierra, y dos filtros que trabajan en el visible (Vis) y en el infrarrojo cercano (NIR). El primero captura imágenes de alta resolución en un amplio rango de frecuencias de luz visible, mientras que el segundo tiene varias funciones: como espectómetro es capaz de recopilar espectros de galaxias en el infrarrojo cercano y como fotómetro puede obtener imágenes en 3 bandas específicas (Y,J,H). La combinación de ambos filtros permite a Euclid realizar mediciones precisas de la morfología y las distancias a las que se encuentran las galaxias, esenciales para confirmar el papel de la energía oscura en la expansión acelerada del universo, así como el rol de la materia oscura en la formación y evolución de las galaxias.

Euclid consta de dos cartografiados de distinta profundidad: el WIDE y el DEEP.

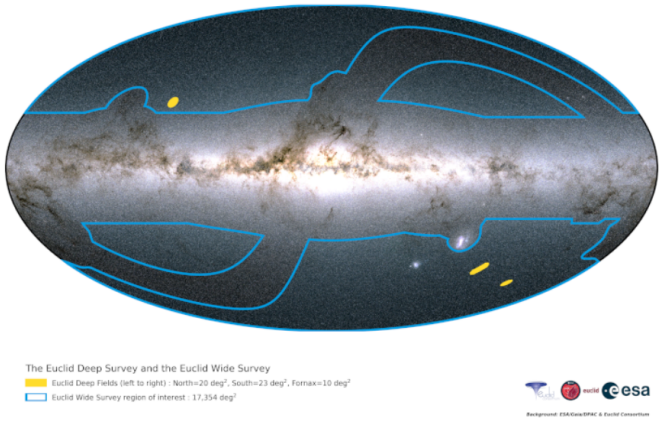
El WIDE cubre 15.000 grados cuadrados de cielo extragaláctico y tiene por objetivo principal el estudio de la evolución a gran escala del universo, crucial para avanzar en la comprensión de la naturaleza de la energía y la materia oscura.

El DEEP, un cartografiado más profundo centrado en la observación de pequeñas porciones de cielo pero con mayor sensibilidad, tiene el objetivo de proporcionar una visión más detallada y profunda de las estructuras del Universo y de servir en labores de calibración para las mediciones del cartografiado WIDE. Esta mayor profundidad le permite detectar galaxias débiles y distantes, lo que resulta de gran utilidad para estudiar la evolución temprana del universo y obtener una colección significativa de galaxias en una gran variedad de corrimientos al rojo. De las 3 regiones profundas en las que se centra el cartografiado DEEP, o *deep drilling fields*, dos de ellas se encuentran en el hemisferio sur y la otra en el hemisferio norte. La siguiente tabla recoge los datos de interés sobre estas regiones:[2]

Nombre	*Localización*	*coordenada(RA/DEC)*	*cobertura(deg^2)*
Euclid Deep Field North (EDFN)	Hemisferio Norte	(+17:58:55,+66:01:03)	20
EDFS(South)	Hemisferio Sur	(-04:04:57,-45:28:22)	23
EDFF(Fornax)	Hemisferio Sur	(-03:31:43,-28:05:18)	10

Tabla 1: *Tabla que recoge los nombres, la localización, las coordenadas y la cobertura de los diferentes Deep Drilling Fields del cartografiado Euclid DEEP.*

Las coordenadas RA(Ascesión Recta) y DEC(Declinación) son equivalentes a la longitud y latitud en la esfera terrestre y forman parte del sistema de coordenadas ecuatoriales. RA se mide a lo largo del ecuador celeste desde un punto conocido como *Punto de Aries* y va de 0 a 24 horas, avanzando hacia el este. La declinación indica la distancia de un objeto al ecuador celeste y va de +90 a -90 grados, desde el polo celeste norte al polo celeste sur.[1]



The Euclid wide and deep surveys. Credit: ESA/Euclid Consortium. Acknowledgment: Euclid Consortium Survey Group

Figura 6: *Figura tomada de la página web oficial de Euclid, mostrando los tres EDF (EDFN, EDFF y EDFS) dispuestos en diagonal desde la esquina superior izquierda hasta la inferior derecha, resaltados en amarillo. Fuente: [2]*

Por otra parte, la tabla 2 recoge los valores de los flujos límites de los filtros de Euclid DEEP y WIDE (en magnitudes AB):

4.2 LSST

El LSST (*Large Synoptic Survey Telescope*) es el proyecto astronómico más ambicioso del observatorio Vera C. Rubin, situado a 2.682 metros sobre el nivel del mar en el Cerro Pachón, Chile. El sofisticado

Filtro	*Profundidad(DEEP)*	*Profundidad(WIDE)*
VIS	26.5	26.4
Y	25.24	24.3
J	25.24	24.5
H	25.25	24.4

Tabla 2: *Valores de los flujos límites de los filtros de Euclid DEEP y WIDE.*

telescopio de LSST cuenta con la cámara digital más poderosa del mundo, de 3200 Mpx, que le permitirá cartografiar el hemisferio sur del universo con una precisión sin precedentes. Este proyecto fue diseñado para abordar objetivos como la medición del efecto de lente gravitacional débil que tiene lugar en las galaxias, el reconocimiento de objetos pequeños y tenues de nuestro sistema solar tales como los cometas y los asteroides, el mapeado de la Vía Láctea y la detección de eventos ópticos transitorios, como las novas y supernovas.

Igual que Euclid, LSST consta de dos cartografiados: el WIDE y el DEEP.

El WIDE cubre todo el hemisferio sur, un área total de 18.000 grados cuadrados. Los objetivos de este cartografiado amplio, rápido y menos profundo son los comentados anteriormente.

El DEEP cubre 40 grados cuadrados del hemisferior sur, repartidos equitativamente entre 4 pequeñas regiones. Entre el 10 y 20% del tiempo de observación se dedicará a la observación intensiva de estas regiones profundas, que son las siguientes:

Nombre	*Localización*	*coordenada(RA/DEC)*	*cobertura(deg^2)*
Elais S1	Hemisferio Sur	(00:37:48,-44:00:00)	10
XMM-LSS	Hemisferio Sur	(02:22:50,-04:45:00)	10
Extended Chandra Deep Field-Soth(ECDFS)	Hemisferio Sur	(+03:32,-28:06:00)	10
Cosmos	Hemisferio Norte	(10:00:24,+02:10:55)	10

Tabla 3: *Tabla que recoge los nombres, la localización, las coordenadas y la cobertura de los diferentes Deep Drilling Fields del cartografiado LSST DEEP.*

Los flujos límites de los filtros de LSST DEEP y WIDE (en magnitudes AB) quedan recogidos en la tabla 4

Filtro	*Profundidad (DEEP)*	*Profundidad (WIDE)*
u	28.5	26.1
g	28.7	27.4
r	28.9	27.5
i	28.4	26.8
z	28.0	26.1
y	27.0	24.9

Tabla 4: *Valores de los flujos límites de los filtros de LSST DEEP y WIDE.*

4.3 SPRITZ

SPRTIZ es un simulador que crea catálogos simulados de galaxias para proyectos de cartografiado espacial (actuales y futuros) que operan en longitudes de onda infrarrojas, entre otras (L. Bisigello et al. 2021). Las simulaciones se construyen a partir de un conjunto de funciones de luminosidad en longitudes de onda del infrarrojo cercano y medio (Herschel y banda K) y una función de masa estelar galáctica, para cubrir una amplia variedad de poblaciones de galaxias y AGNs. En ellas, se asocia una distribución

espectral de energía y propiedades físicas, tales como la masa estelar, la tasa de formación estelar y la contribución AGN, a cada galaxia simulada. Imponiendo diferentes filtros, como los de Euclid y LSST, sobre la distribución de energía espectral asociada a cada fuente, se obtiene el flujo esperado en diferentes bandas para cada galaxia simulada. Este trabajo se centrará en las magnitudes de los flujos de los filtros de LSST y Euclid para la construcción de modelos de clasificación de fuentes.

Los catálogos simulados de LSST y Euclid para las profundidades DEEP y WIDE se han descargado de la página oficial de Spritz[5]. En ellos se incluyen propiedades simuladas de una serie de fuentes en un área de 15.000 deg^2 para el catálogo simulado WIDE y 40 deg^2 para el DEEP. El área simulada coincide con el área real de Euclid (salvo en el DEEP, ya que el área simulada es algo menor), de modo que SPRTIZ ha tomado los filtros de LSST para emular las observaciones que realizaría en caso de "mirar" hacia esas fuentes.

Los catálogos simulados DEEP y WIDE de Euclid incluyen varias propiedades físicas y espectroscópicas, una serie de observaciones relacionadas con la contribución AGN de las fuentes y las magnitudes de los flujos de los filtros de Euclid (VIS, Y, J y H). Por otra parte, las simulaciones de LSST sólo incluyen las magnitudes de los filtros de LSST (ugrizy). En la tabla 5 se presenta el número de fuentes de los catálogos DEEP y WIDE..

	N fuentes
DEEP	3.720.974
WIDE	11.983.762

Tabla 5: *Número de fuentes enlos cartografiados DEEP y WIDE de LSST y Euclid .*

El número de fuentes de la tabla 5 en realidad no refleja el número total de fuentes simuladas. Esto se debe a que en los catálogos sólo figuran las 100 primeras fuentes del mismo tipo. A partir de ahí, cada una de ellas se pondera con un peso de $10^{log_{10}N-2}$, siendo log10N una de las columnas de los catálogos, donde N es el número de fuentes del mismo tipo. Para simplificar el problema, este detalle no se ha tenido en cuenta a la hora de construir los modelos de clasificación.

5 Herramientas

5.1 Topcat/Stilts

TOPCAT y STILTS son herramientas de software que se utilizan principalmente en la astronomía para la manipulación y análisis de datos tabulares, como catálogos de estrellas y galaxias. Son parte del mismo conjunto de herramientas de software, enteramente escrito en Java, y permiten realizar una gran cantidad de operaciones sobre los datos, incluyendo selección de filas, y columnas, estadísticas de columnas, creación de gráficos e histogramas, correlación de tablas, conversión de magnitudes y filtrado de datos, entre otras. Su principal diferencia radica en cómo se interactúa con ellas: mientras que TOPCAT es una aplicación interactiva, con una interfaz gráfica de usuario (GUI), STILTS es su versión en línea de comandos. [7][6]

Estas diferencias hacen que TOPCAT sea más indicado para la exploración visual de los datos y la manipulación interactiva de las tablas, mientras que STILTS es ideal para tareas que se quieren automatizar. Por ejemplo, si se quiere hacer una correlación de tablas, basta con utilizar el script de línea de comandos de STILTS de una correlación anterior y cambiar los archivos de entrada para realizar la operación. Esto permite que el usuario no tenga que ir recordando las secuencias de acciones de cada operación realizada en TOPCAT. En el Apéndice se presentan las operaciones que han sido realizadas con comandos de línea de STILTS en este trabajo, las cuales fueron ejecutadas desde la terminal de Windows.

5.2 Librerías de python

En la última década, el lenguaje de programación Python ha emergido como una herramienta clave en el campo del aprendizaje automático o Machine Learning (ML). Python, conocido por su sintaxis sencilla y fácil de interpretar, es un lenguaje de alto nivel que ofrece un rico y extenso ecosistema de librerías y herramientas especializadas, como Numpy, Pandas o Sckit-learn, que destacan por su capacidad para simplificar los procesos del aprendizaje automático, que van desde la recopilación y procesamiento de los datos hasta la construcción y evaluación de los modelos, haciendo que el ML sea mucho más accesible y eficiente. [8]

A continuación, se describen las librerías que han tenido un papel importante en el desarrollo del trabajo.

5.2.1 Numpy

NumPy es una librería muy conocida de Python que da soporte para crear vectores y matrices multi-dimensionales, junto con una extensa colección de funciones matemáticas de alto nivel. Estas funciones permiten operar de manera eficiente con grandes conjuntos de datos, haciendo de Numpy una herramienta esencial para el análisis y la construcción de modelos de aprendizaje automático.

5.2.2 Matplotlib

Matplotlib es una librería diseñada para la creación de gráficos de alta calidad, fundamentales para comprender y explorar nuestros conjuntos de datos, así como para analizar los resultados obtenidos por nuestros modelos. La mayoría de gráficas se han obtenido haciendo uso de esta librería.

5.2.3 pandas

Pandas es una librería de Python que ofrece una amplia variedad de métodos y funciones orientados al análisis y manipulación de datos tabulares. Su estructura de datos principal, el *Dataframe*, resulta muy cómoda a la hora de realizar tareas de limpieza, exploración y visualización de datos, lo que la convierte en una herramienta especialmente útil en el proceso de construcción de modelos de aprendizaje automático. A diferencia de NumPy, Pandas permite introducir un tipo distinto de dato en cada columna (int, float, str, fecha, entre otros) y es compatible con numerosos formatos de archivos. En este caso, los datos se cargaron utilizando el formato .CSV.

5.2.4 seaborn

Seaborn es una librería orientada a la visualización de datos en python . Está basada en Matplotlib, es decir, extiende sus funcionalidades y mejora la interfaz de usuario para dibujar gráficos más intuitivos y visualmente atractivos. Aunque Matplotlib también puede trabajar con DataFrames de Pandas, Seaborn hace que sea más sencillo y directo generar visualizaciones a partir de estas estructuras de datos. Seaborn se ha utilizado para crear gráficos cuya correcta interpretación era vital para entender aspectos importantes de los datos introducidos.

5.2.5 sklearn

Scikit-learn, comúnmente referida como sklearn, es una librería de python que destaca por su eficiencia y facilidad de uso en la construcción de modelos de aprendizaje automático. Constituye la librería más importante de este trabajo ya que se ha empleado en el entrenamiento y la evaluación de los algoritmos de aprendizaje supervisado implementados.

5.3 Entornos de Programación

5.3.1 Jupyter Notebook

Jupyter Notebook es un entorno interactivo de programación en python que resulta especialmente útil para desarrollar modelos de machine learning y realizar un análisis exploratorio de los datos utilizados. Su estructura se basa en celdas independientes, lo que lo hace idóneo para crear documentos que combinen código, texto explicativo, gráficos y ecuaciones. El carácter independiente de sus celdas ha resultado particularmente útil a la hora de realizar varias pruebas y comprobaciones secundarias sin afectar el código principal. Uno de los puntos negativos de jupyter notebook, es que tiende a sobrecargarse de celdas secundarias, por lo que es habitual tener que lidiar con cuadernos caóticos. Una forma de evitarlo es acostumbrarse a eliminar las celdas una vez realizada las prueba o comprobaciones

5.3.2 Google Colab

Google Colab es un entorno de programación en la nube basado en los cuadernos de Jupyter. Su principal ventaja es que permite que varios usuarios editen los cuadernos a tiempo real, lo que ha resultado fundamental para compartir mis avances con los directores de este TFG y recibir sus correciones.

6 Metodología

6.1 Preparación de los datos

Como se ha comentado anteriormente, los datos utilizados para crear todos los modelos de clasificación de fuentes fueron extraídos de la página oficial de Spritz [5]. En ella se incluyen datos simulados pertenecientes a varias misiones espaciales, tanto futuras como actuales, dedicadas al cartografiado de galaxias con el fin de expandir nuestro conocimiento acerca del Universo. Para este trabajo se tomaron los datos simulados de los proyectos LSST y EUCLID, que constan de dos cartografiados de galaxias de distinta profundidad: DEEP y WIDE. Lo primero que se hizo fue cargar estos cartografiados en TOPCAT. El formato elegido para cargar los archivos fue .fits (acrónimo en inglés de *Flexible Image Transport System*), muy utilizado en astronomía para almacenar y manipular tablas con un gran volumen de datos, como es el caso.

Debido a que los catálogos DEEP y WIDE de LSST y EUCLID presentan distintas propiedades simuladas de los mismos objetos astronómicos, se han unido sus respectivas tablas para aglutinar toda la información únicamente en dos conjuntos de datos: DEEP y WIDE. Este proceso se conoce como correlación y consiste en asociar las propiedades simuladas de un catálogo (por ejemplo, el WIDE de LSST) con las correspondientes a las mismas fuentes del otro (en este caso, el WIDE de EUCLID). La correlación de las propiedades simuladas de los dos catálogos DEEP y WIDE se ha realizado en base a un identificador asociado a cada fuente, el cual relaciona fuentes con propiedades simuladas de ambos catálogos.

Anteriormente se ha comentado que las fuentes simuladas pertencen al área de estudio de Euclid y que, para LSST, se tuvo en cuenta la sensibilidad en cada uno de sus filtros para emular las observaciones que realizaría en caso de *mirar* hacia esas fuentes. En este trabajo hemos decidido quedarnos con las fuentes que realmente estarán en las zonas comunes de ambos proyectos. Para ello, se han determinado sus áreas de solapamiento, tanto en el DEEP como en el WIDE, para luego realizar un cribado de todas las fuentes situadas fuera de esas áreas de solapamiento. El resultado de la criba se traduce en una reducción de fuentes acorde al porcentaje de sus respectivas áreas de solapamiento.

Por otra parte, debido a que los límites de detección de los filtros de LSST y Euclid se dan en mangitudes AB, se ha realizado una conversión de unidades de los flujos, de microjanskys a magnitudes

AB , aplicando un factor de conversión de 10^{-6}, para pasar de microjanskys a janskys, y utilizando la función de TOPCAT/STILTS *JasnkyToAb()*.

Por último, comentar que la correlación, el cribado de fuentes fuera del área de solapamiento y la conversión de unidades de los flujos son operaciones que han sido realizadas con comandos de STILTS (ver Apéndice). De esta forma se han generado los datos simulados que han servido de base para la construcción de los modelos.

6.2 Preparación del modelo

Para adaptar el modelo al problema físico real, se ha realizado con python un filtrado utilizando los flujos límite de los filtros (ver Datos), de tal manera que si una fuente no supera la detección mínima para alguno de los filtros, no se registrará detección en ese filtro. El valor de aquellos flujos que no superan el límite de detección se ha sustituido por un *NaN* o valor nulo. Para simplificar el problema solo se han tomado aquellas fuentes que superan el límite de detección en todos los filtros. De esta forma, se obtienen unas mediciones que reproducen fielmente las observaciones reales que podrían tomar Euclid y LSST.

Ahora ya estamos en disposición de realizar la preparación del modelo. Para ello, se abre un cuaderno de Jupyter, que es el entorno en el que se desarrolla todo el proceso de machine learning: la preparación, el entramiento y la evaluación del modelo.

En primer lugar se importan las librerías de Python (ver Herramientas) y se cargan los datos en un *Dataframe* de Pandas. A continuación, se crea el vector 'y' que contiene las etiquetas del dataset, es decir, la categoría a la que pertenece cada fuente (AGN/NO-AGN). Como se dijo anteriormente, en el dataset existen una serie de variables relacionadas con la contribución AGN, las cuales evidencian la clasificación de las fuentes. Se trata de un etiquetado binario, de modo que a las fuentes con detección en las columnas AGN se les ha asignado el número 1 (AGN), dejando el 0 (NO-AGN) para el resto de fuentes.

Seguidamente, se realiza una selección de las características o variables que van a intervenir en los modelos de clasificación. En este caso, se han seleccionado las magnitudes de los filtros de Euclid y LSST para crear el primer modelo, mientras que para el segundo, únicamente se tomaron las magnitudes de Euclid. El tercer modelo incluye, además de las magnitudes de los filtros de Euclid, unas características adicionales, obtenidas a partir de la resta de las magnitudes de dos filtros, los llamados *índices de color*.

Por último, se ha dividido el dataset en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba o validación. La razón es que no se pueden utilizar los mismos datos para construir y evaluar el modelo. Esto se debe a que el modelo memorizaría todos los datos y siempre sería capaz de predecir la etiqueta correcta. Sin embargo, esto no nos indica que el modelo vaya a generalizar bien a nuevos datos. Por ello, se reserva el conjunto de validación para evaluar la capacidad predictiva del modelo con fuentes que no ha visto durante el entrenamiento [8]. Para la división de los datos se ha utilizado la función *train test split* de la librería sklearn. Esta función permite fijar el porcentaje de los datos que será destinado al conjunto de prueba. En este caso, ese porcentaje se fijó en el 20 %. Además, se ha utilizado el atributo *stratify* para asegurar que la distribución de las etiquetas en los conjuntos de entrenamiento y validación sea la misma que en el conjunto original. Esto resulta especialmente útil en los datasets con clases desbalanceadas, como es el caso.

6.3 Entrenamiento y Evaluación

Para la creación de los modelos de clasificación de fuentes se seleccionaron los algoritmos de aprendizaje supervisado mencionados anteriormente: Decision Tree y Random Forest. Estos dos algoritmos se implementaron en cada una de las 3 variantes de los conjuntos de datos DEEP y WIDE: 1) incluyendo las magnitudes de LSST y Euclid, 2) incluyendo sólo las magnitudes de Euclid, 3) incluyendo las magnitudes

Euclid y columnas de colores (o índices de color).

Todos los modelos de clasificación fueron evaluados mediante la métrica de rendimiento f1, ya que al ser un promedio de la precisión y la sensibilidad ofrece una idea global del rendimiento de los modelos, y el AUC de la curva ROC, debido a que resume la capacidad del modelo para discriminar entre clases. Además, para el tercer conjunto de datos también se añadió la Curva PR, y el AP que resume sus resultados, para realizar una comprobación más específica de la relación entre la precisión y la sensibilidad de los modelos.

Por último, para obtener los mejores rendimientos posibles en cada modelo se dibujaron gráficas que aglutinan curvas ROC para distintas profundidades máximas, en el caso de los modelos de Decision Tree, y para distinto número de árboles, en el caso de los de Random Forest. La razón es que son los hiperparámetros que más influyen en los rendimientos de sus respectivos modelos. Combinando las curvas ROC (que también contienen el valor de la métrica F1) con gráficos que miden el overfitting [1] asociado a cada Curva ROC, se obtuvieron los valores de los hiperparámetros que daban lugar a los mejores rendimientos de cada modelo.

7 Resultados

7.1 Preparación de los datos y los modelos

En primer lugar, se han determinado las áreas de solapamiento. En el caso del WIDE, Euclid cubre un área de 15.000 grados cuadrados repartidos entre los dos hemisferios de observación definidos por el plano de la eclíptica (plano en el que orbita la Tierra entorno al Sol), mientras que LSST cubre todo el hemisferio Sur terrestre (es decir, el hemisferio sur respecto al plano ecuatorial de la Tierra), unos 18.000 grados cuadrados de cielo. Como es bien sabido, el plano ecuatorial de la Tierra no es coplanar con el plano de la eclíptica, sino que está inclinado con respecto a él un ángulo aproximado de 23 °, lo que se conoce como oblicuidad de la eclíptica (Ver fig 7).

Figura 7: *Oblicuidad de la eclípitca. El plano ecuatorial de la Tierra está inclinado unos 23 ° respecto al plano de la eclíptica. Fuente: [14]*

En la figura 8 se representa un croquis sencillo para visualizar el de solapamiento de los dos cartografiados WIDE:

[1] El overfitting se midió tomando el valor absoluto de la resta de las métricas f1 obtenidas para el conjunto de entrenamiento y validación.

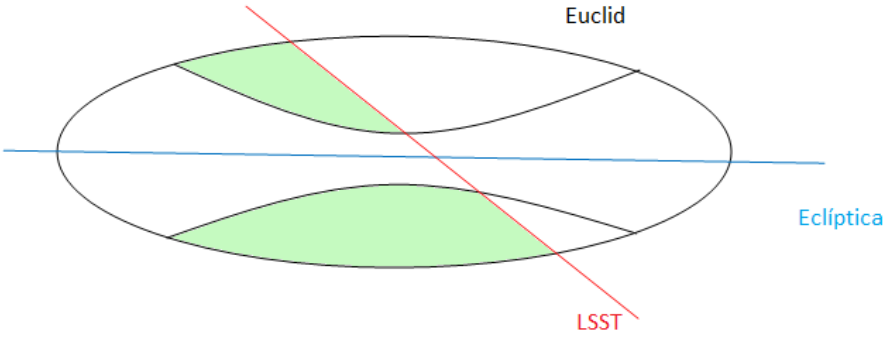


Figura 8: *Croquis del área de solapamiento de los cartografiados LSST y Euclid para el caso WIDE. Euclid observa el área contenida en los lóbulos (líneas negras) perpendiculares al plano de la eclípitica (línea azul), mientras que LSST observa todo el área situada a la izquierda de la línea roja, que representa el ecuador terrestre. Las regiones coloreadas en verde se corresponden con las áreas de solapamiento de ambos. Fuente: [Elaboración propia]*

De esta forma, se ha determinado que el área de solapamiento en el WIDE es del 50 %. Cómo se dijo en la sección de Datos, las fuentes simuladas pertenecen al área de Euclid, de modo que si el solapamiento con LSST es la mitad de ese área, ambos cartografiados tendrán en común la mitad de las fuentes.

En el caso del DEEP, para Euclid se han simulado 40 grados cuadrados de los 53 que suma entre todas sus regiones de campo profundo, o *Deep Drilling Fields*. Para simplificar el problema, se ha asumido que esos 13 $\deg^2$ que quedan excluidos en la simulación, corren a cuenta de los Deep Drilling Fields: EDFN (20 $\deg^2$) y EDFS (23 $\deg^2$) (Ver tabla 1). El motivo es que el solapamiento se produce entre las regiones profundas EDFF (Euclid) y ECDFS (LSST) (ver tablas 1 y 3), de 10 grados cuadrados cada una, de modo que el solapamiento en el DEEP es del 25 %.

En la figura 9 se muestra un croquis sencillo que ilustra el solapamiento de los dos cartografiados DEEP:

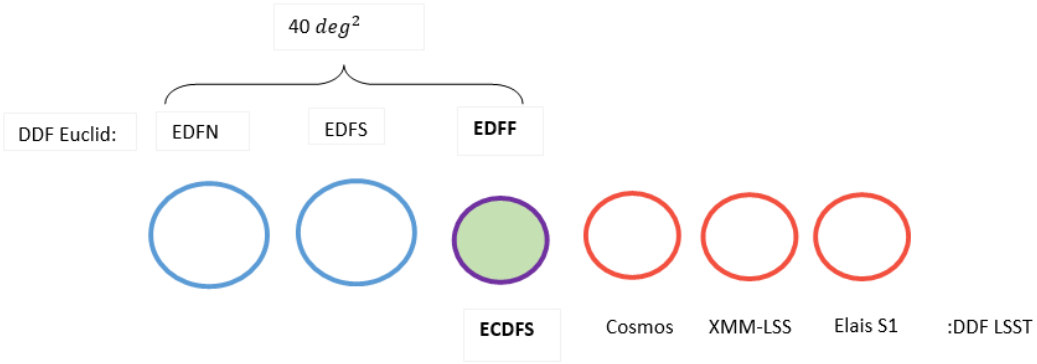


Figura 9: *Croquis del área de solapamiento de los cartografiados LSST y Euclid para el caso DEEP. Los Deep Drilling Fields (DDF) de Euclid vienen representados por círculos en azul, mientras que los correspondientes de LSST aparecen en rojo. Por otra parte, el solapamiento entre las regiones EDFF (Euclid) y ECDFS (LSST) se representa con un círculo verde enmarcado en morado. Fuente: [Elaboración propia]*

El cribado de fuentes fuera del área de solapamiento se ha realizado con comandos de Stilts. Estos comandos, en primer lugar, asignan un valor aleatorio entre 0 y 1 a cada fuente y luego, seleccionan aquellas fuentes con valores comprendidos entre 0 y la fracción correspondiente de área de solapamiento (DEEP: 0.25 y WIDE: 0.50).

Por otra parte, el filtrado de flujos límite se llevó a cabo con python (Ver Apéndice) utilizando los límites de detección de cada uno de los filtros que figuran en las tablas: 1 (Euclid WIDE), 3 (Euclid DEEP), 4 (LSST WIDE) y 6 (LSST DEEP).

Con todo, en la tabla 6 se presenta cuál ha sido la disminución de fuentes tras cada uno de los procesos de filtrado.

	N total	*filtrado por solapamiento*	*filtrado flujos límite*	*% AGN*
DEEP	3.720.974	930.107	530.134	17%
WIDE	11.983.762	5.992.762	3.787.503	55%

Tabla 6: *Número de fuentes resultantes en los catálogos DEEP y WIDE tras el cribado de fuentes fuera del área de solapamiento y el filtrado de flujos límite. La última columna recoge el porcentaje de AGNs presente en cada catálogo.*

Respecto a los porcentajes de AGNs, nos ha llamado la atención la gran diferencia que hay entre ambos catálogos, ya que se esperaban obtener porcentajes similares. Como ya se dijo anteriormente, en las simulaciones de SPRITZ solo figuran las 100 primeras fuentes del mismo tipo y , a partir de ahí, cada una de ellas se pondera con un peso de $10^{log_{10}N-2}$. Esto indica que en los conjuntos de datos utilizados existen tipos de fuentes que están infrarrepresentados, de modo que se hallaron los porcentajes de los tipos de fuentes con un valor de log10N superior a 2 (es decir, con más de 100 fuentes). Para ello, se crearon subconjuntos en TOPCAT imponiendo la condición $log10N > 2$ y los porcentajes obtenidos se ilustran en la tabla 7:

En vista de estos porcentajes tan altos, se procedió a realizar una desagregación de fuentes para obtener

	% log10N>2
DEEP	51 %
WIDE	82 %

Tabla 7: *Porcentajes de tipos de fuentes infrarrepresentados.*

el porcentaje real de AGNs y así poder comprobar si son parecidos. Para la desagregación de fuentes simplemente se asignó un peso de 1 a todos las fuentes con menos de 100 muestras del mismo tipo, mientras que aquellas que contaban con más de 100 muestras , se le asignó un peso de $10^{log_{10}N-2}$, siendo N el número de muestras de cada tipo. En el Apéndice aparece el código de python que se utilizó para llevar a cabo este proceso. En la tabla se exponen los resultados del proceso de desagregación:

	N total	*AGNs*	*% AGN*
DEEP	1.426.378	109.018	7.6%
WIDE	546.335.679	43.989.011	8.1%

Tabla 8: *Tabla que recoge el número total de fuentes , el número de AGNS y el porcentaje de AGNs tras la desagregación de fuentes en los cartografiados DEEP y WIDE.*

En la tabla 8 se observa que los porcentajes reales de AGNs son muy similares y razonables.

7.2 Entrenamiento y evaluación de los modelos

7.2.1 Modelos con filtros de LSST y EUCLID

En las figuras 10 y 11 se presentan los resultados obtenidos para los modelos de Decision Tree que han sido entrenados con las magnitudes de los filtros de LSST y Euclid. En ellas se representan la curva ROC

[2] , el AUC y la métrica F1 para clasificadores de Árbol de Decisión con distintas profundidades.

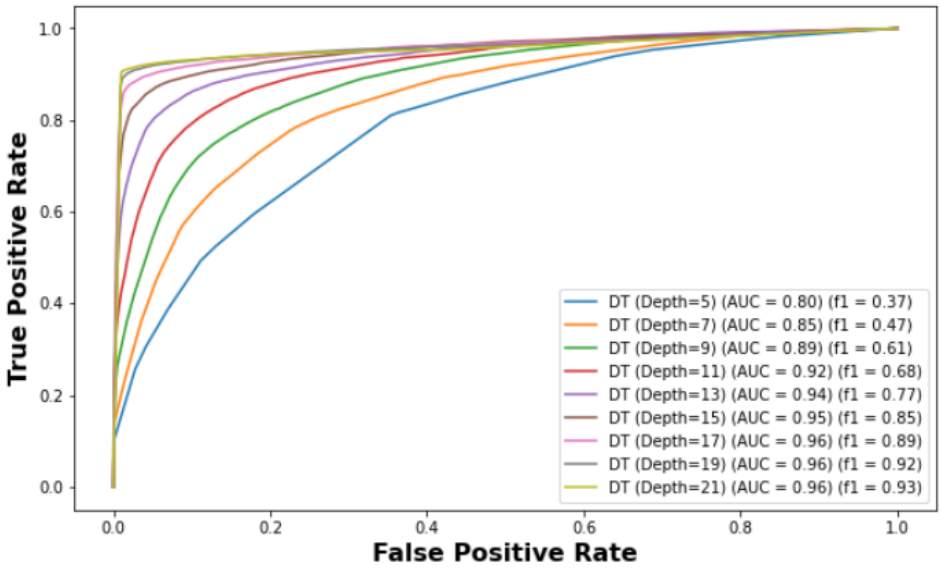


Figura 10: *Curvas ROC para clasificadores de árbol de decisión con distintas profundidades para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, se observa como los rendimientos de las métricas mejoran notablemente con el aumento de la profundidad del árbol. Si bien el AUC se estanca, o deja de aumentar, a partir de una profundidad de 17, la métrica f1 sigue creciendo aunque ya muy lentamente. Se puede ver como rápidamente se alcanzan unas métricas con rendimientos excelentes, rozando los de un clasificador perfecto.

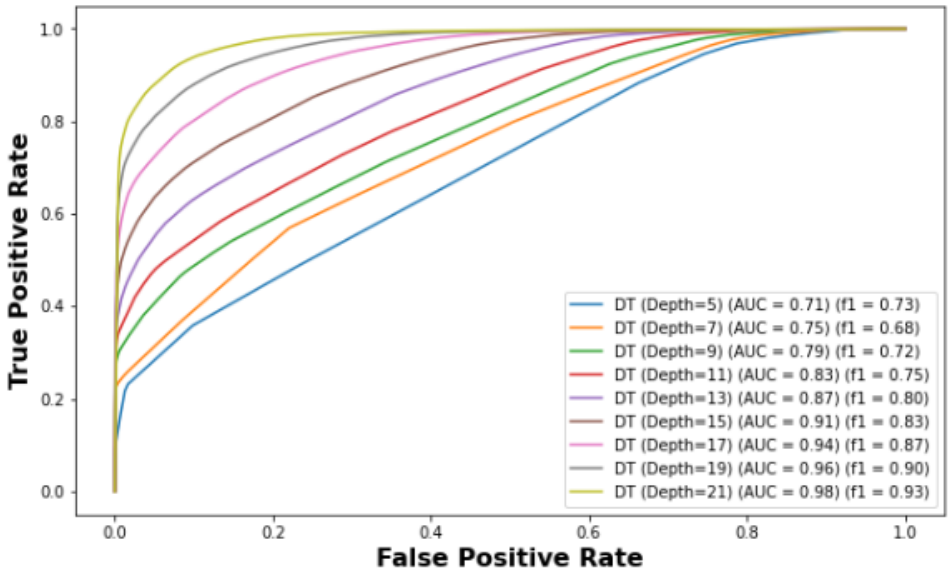


Figura 11: *Curvas ROC para clasificadores de árbol de decisión con distintas profundidades para el conjunto WIDE. Además, también se incluyen los valores de las métricas AUC y F1.*

En el WIDE ocurre esencialmente lo mismo. Los rendimientos de las métricas mejoran significativamente con el aumento de la profundidad del árbol, de forma que se vuelve a alcanzar rápidamente

[2]Las Curvas ROC se representan para umbrales de decisión que van de 0 a 1, donde 1 significa que el clasificador está 100% convencido de que la fuente es un AGN (clase positiva).

un clasificador que roza la perfección. A diferencia del DEEP, los valores de las métricas ya son muy buenos a bajas profundidades, es decir, el árbol de decisión parece que necesita muy pocas divisiones de los datos para clasificar las fuentes razonablemente bien. Además, salvo ese pequeño bajón de f1 al pasar de 5 a 7 de profundidad, ninguna de las métricas se estancan en ningún momento y todo apunta a que si probaramos con profundidades mayores, todavía se obtendrían mejores rendimientos.

Por otra parte, en las figuras 12 y 13 se presentan los resultados correspondientes a los modelos de Random Forest. En ellas se representan la curva ROC, el AUC y la métrica F1 para clasificadores Random Forest con distinto número de árboles.

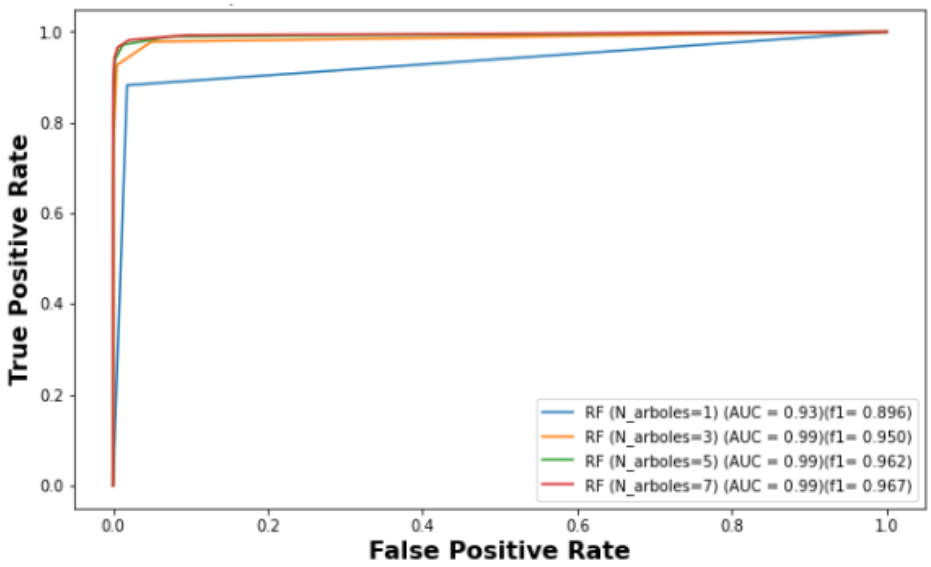


Figura 12: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, se observa como ya solo con un árbol se alcanzan rendimientos muy buenos y que, simplemente aumentando a tres árboles, la métrica AUC ya alcanza un rendimiento casi del 100%, mientras que para f1 se obtiene un rendimiento excelente. A partir de aquí, el AUC se estanca y la métrica f1 aumenta muy lentamente. Solo hay que ver la angulosidad de las curvas ROC (todas pegadas a la esquina superior izquierda) para advertir que con pocos árboles se obtienen clasificadores prácticamente perfectos.

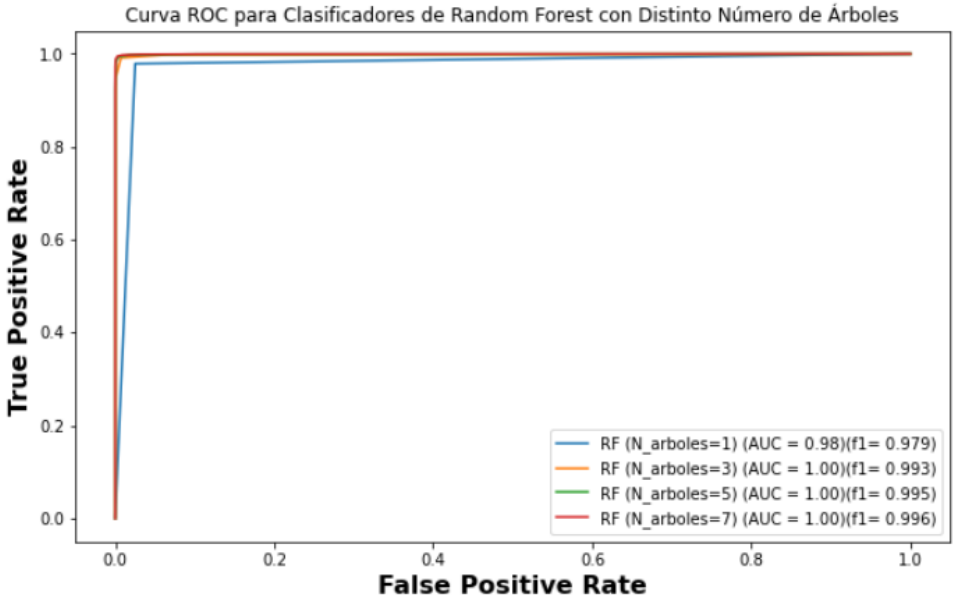


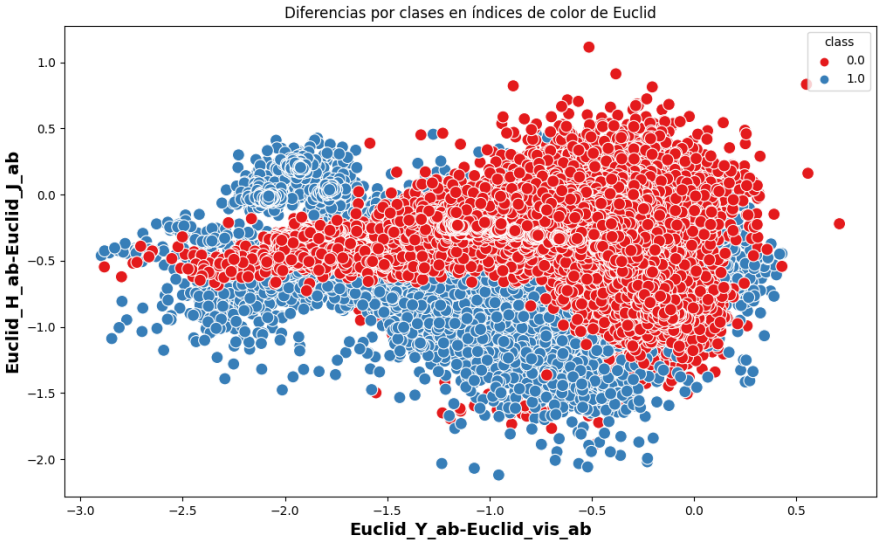
Figura 13: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el WIDE se repite el paralelismo visto en los modelos de Decision Tree. Los rendimientos de las métricas son notablemente mejores que en el DEEP para un solo árbol y simplemente con añadir dos árboles más al bosque, ya se alcanzan, está vez sí, rendimiento perfectos de las métricas.

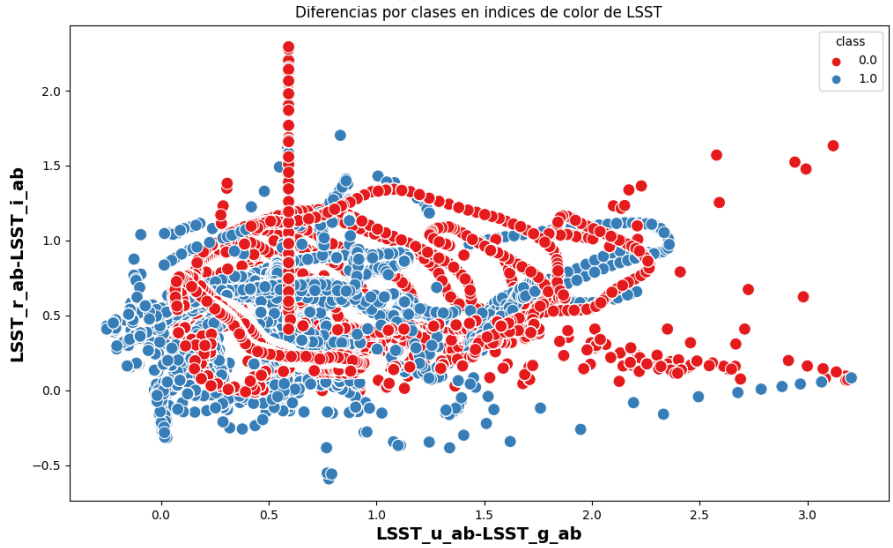
Los resultados obtenidos para todos los modelos son tan extraordinarios que se empezó a sospechar que había algún patrón interno en los datos que debía estar evidenciando la clasificación de las fuentes. Para comprobar si esta sospecha estaba bien fundada, decidimos representar pares de índices de color (resta de magnitudes de distintos filtros) tanto para los filtros de Euclid como los de LSST. De este modo, podíamos averiguar si los patrones internos se hallaban en las magnitudes de uno u otro. El hallazgo que hicimos se ilustra en la figura 14.

De esta forma, nuestras sospechas fueron confirmadas. En el caso de Euclid (ver fig 14a) se observa que los datos ocupan regiones extensas con bordes difusos, mientras que en LSST, en vez de encontrar las manchas homogéneas que se aprecian en Euclid, se observan líneas en forma de cordones, azules y rojos, múltiples espacios en blanco entre ellas e incluso una recta roja vertical. Claramente, todos estos patrones son los que estaban evidenciando la clasificación de las fuentes.

Finalmente, revisando el artículo en el que están basadas las simulaciones SPRTIZ (L. Bisigello, et al.(2021)), descubrimos que no se habían introducido los errores en las magnitudes de los filtros de LSST, siendo este el motivo por el que aparecen patrones internos que evidencian la clasificación de las fuentes (ver fig 14b). Ante esta disyuntiva decidimos descartar las magnitudes de LSST y seguir adelante únicamente con las magnitudes de Euclid.



(a) *EUCLID*



(b) *LSST*

Figura 14: *Representación gráfica de pares de índices de color de Euclid (a) y LSST(b). Se han utilizado dos colores para diferenciar las clases de cada fuente: AGN (azul) y NO-AGN (rojo).*

7.2.2 Modelos con filtros de EUCLID

Los resultados obtenidos para los modelos de Decision Tree que han sido entrenados únicamente con las magnitudes de los filtros de Euclid se presentan en las figuras 15 y 18. En ellas se representan la curva ROC, el AUC y la métrica F1 para clasificadores de Árbol de Decisión con distintas profundidades

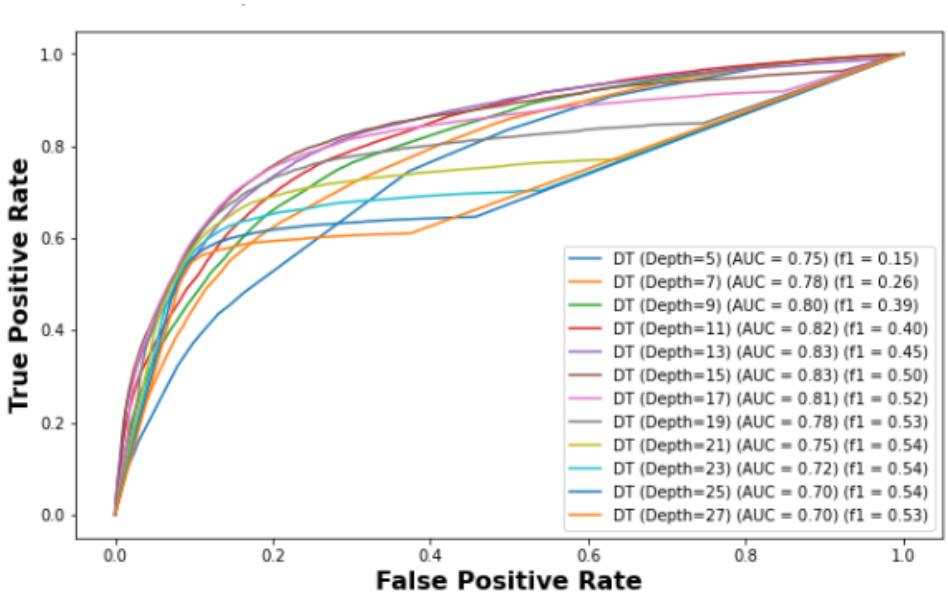


Figura 15: *Curvas ROC para clasificadores de Árbol de Decisión con distintas profundidades para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, se observa que a profundidades bajas la diferencia entre el AUC y f1 es abismal, siendo los rendimientos de AUC muy superiores a los de f1. Sin embargo, esta distancia se va acortando a medida que aumenta la profundidad del árbol. Además, resulta llamativo el bajón progresivo que da el AUC, así como el estancamiento de f1, a partir de una profundidad de 15. . El quiebre que experimentan las métricas a partir de dicha profundidad sugiere hacer una comprobación de overfitting:

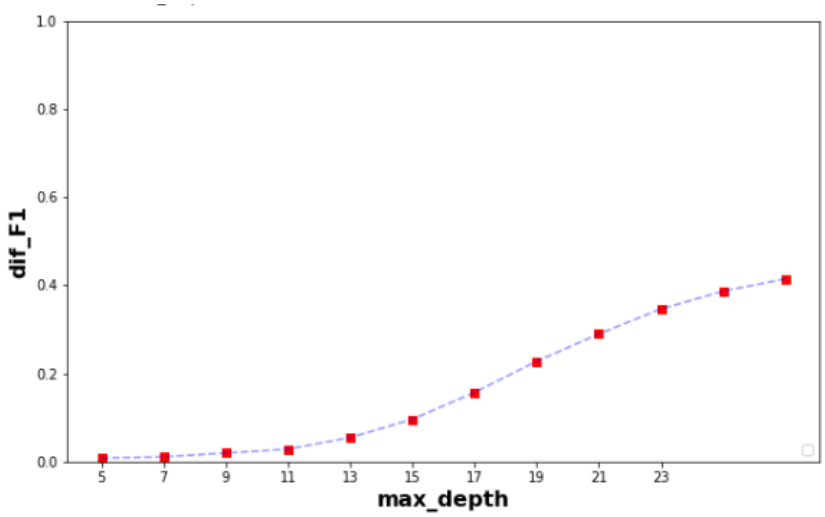


Figura 16: *Comprobación de overfitting para clasificadores de árbol de decision con distintas profundiades para el conjunto DEEP. En el eje Y se presenta la diferencia entre las métricas f1 obtenidas para los conjuntos de entrenamiento y validación.*

El gráfico muestra un claro inicio de sobreajuste en los modelos a partir de una profundidad de 15 [3]. De esta forma, los mejores rendimientos que se pueden obtener de las métricas, sin incurrir en overfitting, se dan para un árbol con una profundidad máxima de 15 (AUC= 0.83, f1=0.50)

Por otra parte, debido al bajo rendimiento de la métrica f1, se ha decidido comprobar la relación

[3]Se considera que el modelo entra en régimen de overfitting a partir de una diferencia entre las métricas f1 (train y test) de 0.2.

entre la precisión y la sensibilidad del modelo a través de la representación de la curva PR (especialmente indicada para conjuntos desbalanceados, como es el caso). De esta forma, se pretende averiguar cuál de las métricas está causando el bajo rendimiento de f1.

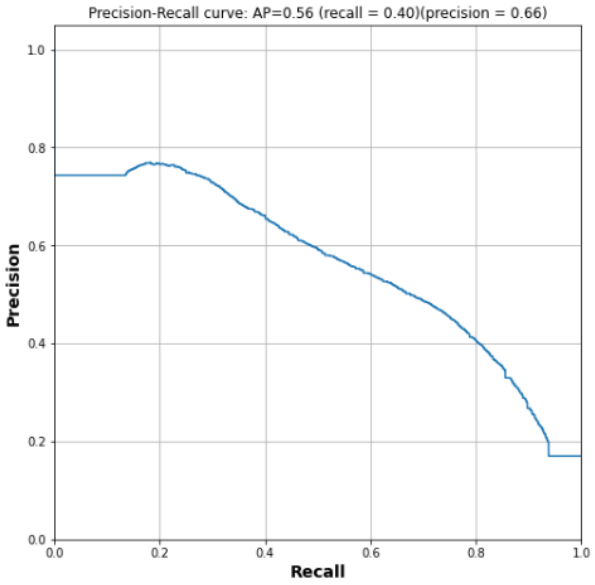


Figura 17: *Curva PR de clasificador de árbol de decisión con una profundidad máxima de 15. Además, también se incluye el Average Precision(AP), la precision y la sensibilidad (recall) del modelo.*

La figura 17 pone de manifiesto que la baja sensibilidad del modelo es lo que está provocando el bajo rendimiento de f1. Este resultado muestra que el modelo, pese a ser moderadamente preciso a la hora de clasificar una fuente como AGN, detecta muy pocos de los AGNs. Esto hace que el número de falsos negativos sea muy alto y, por tanto, que la sensibilidad del modelo sea muy baja.

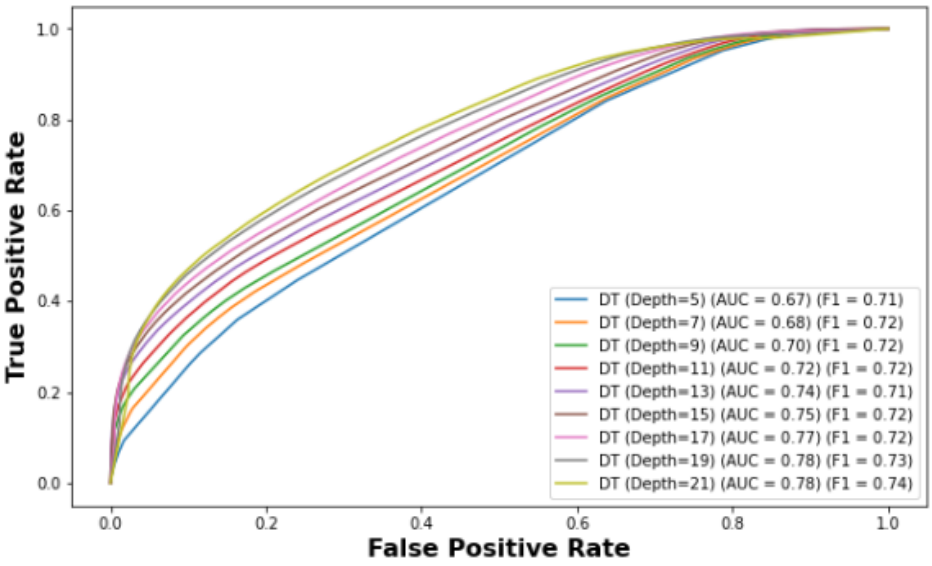


Figura 18: *Curvas ROC para clasificadores de Árbol de Decisión con distintas profundidades para el conjunto WIDE. Además, también se incluyen los valores de las métricas AUC y f1.*

En el WIDE, lo primero que llama la atención es que la métrica f1 es muy superior que en el DEEP. No solo eso, sino que además el rendimiento de f1 es bastante bueno a bajas profundidades, siendo incluso mejor que el AUC, que hasta ahora había estado siempre por delante. Sin embargo, mientras que el AUC

mejora progresivamente con el aumento de la profundidad, la métrica f1 práticamente se queda estancada desde una profundidad de 5. Esto podría llevar a pensar que se está produciendo sobreajuste, no obstante no es el caso. [4] . Con todo, se tiene que los mejores rendimientos de las métricas se obtienen para un árbol con una profundidad máxima de 21 (AUC= 0.78, f1=0.74).

Por otra parte, en las figuras 19 y 20 se presentan los resultados correspondientes a los modelos de Random Forest. En ellas se representan la curva ROC, el AUC y la métrica F1 para clasificadores Random Forest con distinto número de árboles.

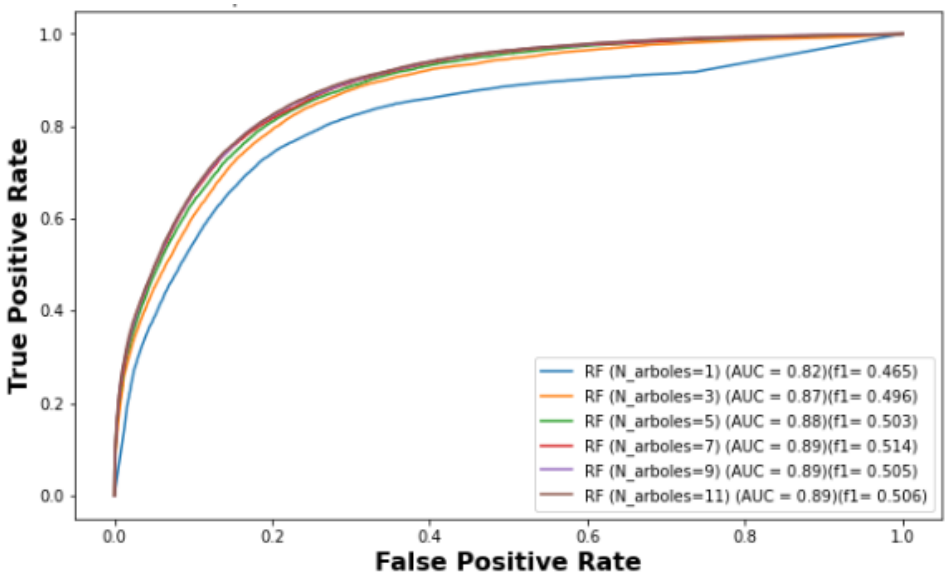


Figura 19: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, se observa que la diferencia entre el AUC y f1 vuelve a ser muy grande y que , a partir de 7 árboles, el AUC se estanca y f1 empeora. Por tanto, los mejores rendimientos se obtienen para un bosque de 7 árboles (AUC= 0.89, f1= 0.514). Mencionar también que todos estos modelos de Random Forest fueron entrenados para una profundidad máxima de 15, ya que, como se vió en el modelo de Decision Tree, a partir de 15 los árboles entran en régimen de overfitting.

[4] A partir de ahora solo se muestran las comprobaciones en los casos que se produce overfitting. Para el resto, solo mencionaré que no se produce el sobreajuste.

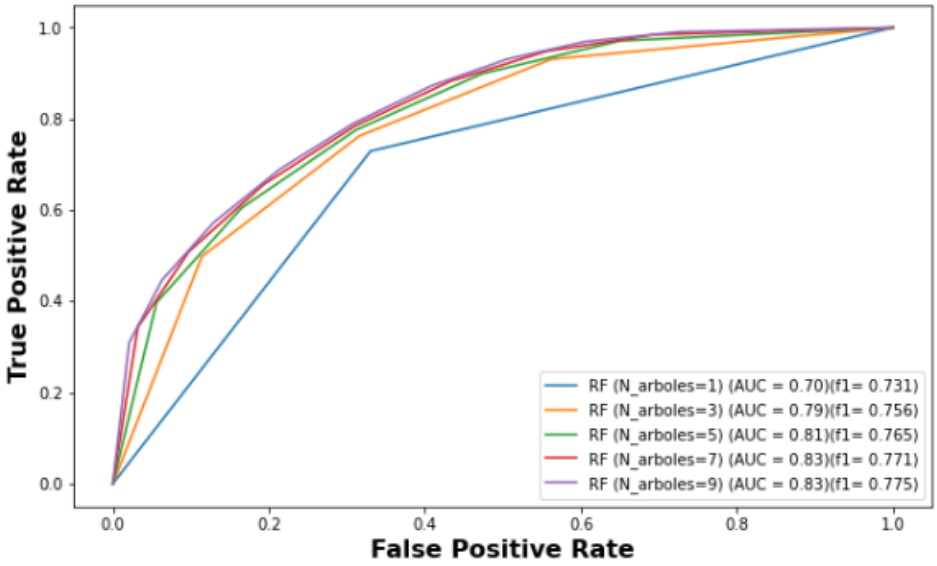


Figura 20: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto WIDE. Además, también se incluyen los valores de las métricas AUC y f1.*

Para el WIDE, sucede algo parecido que en el modelo de Decision Tree, las métricas AUC y f1 están muy parejas y ambas mejoran al aumentar el número de árboles. El mejor clasificador se obtiene en este caso para un bosque de 9 árboles (AUC=0.83, f1=0.775).

En primer lugar se concluye que los resultados obtenidos para los modelos entrenados con magnitudes de Euclid son razonables, demostrando así que el error no incluido en las magnitudes de LSST era el motivo por el que se obtenían clasificadores perfectos. Además, se observa que los rendimientos de las métricas obtenidos para el WIDE son más parejos, siendo el AUC mayor en los modelos del DEEP y f1 muy superior en los del WIDE. Es posible que el desbalance de clases presente en el conjunto DEEP (ver tabla 6) esté motivando los bajos rendimientos de f1. El problema es que al contar con menos ejemplos de AGNs, la precisión y la sensibilidad de los modelos se ven resentidas, las cuales influyen directamente en la métrica f1. En general, se puede afirmar que los modelos de machine learning obtenidos para el conjunto WIDE son bastante buenos, mucho mejores que los del DEEP.

Otra de las razones por las que el conjunto DEEP puede estar dando lugar a peores resultados es que cuenta con una cantidad mucho menor de fuentes que el WIDE. De modo que, para tratar de aportar más información a los modelos DEEP, y obtener así mejores resultados, se ha probado a introducir columnas de índices de color. [5]

7.2.3 Modelos con filtros de EUCLID y columnas de colores

Las columnas de colores añadidas fueron las siguientes: Vis-Y, Y-J y J-H.

Los resultados obtenidos para los modelos de Decision Tree entrenados con las magnitudes de Euclid y las columnas de colores se presentan en las figuras 15 y 18. En ellas se representan la curva ROC , el AUC y la métrica F1 para clasificadores de Árbol de Decisión con distintas profundidades

[5] También se llaman columnas de colores

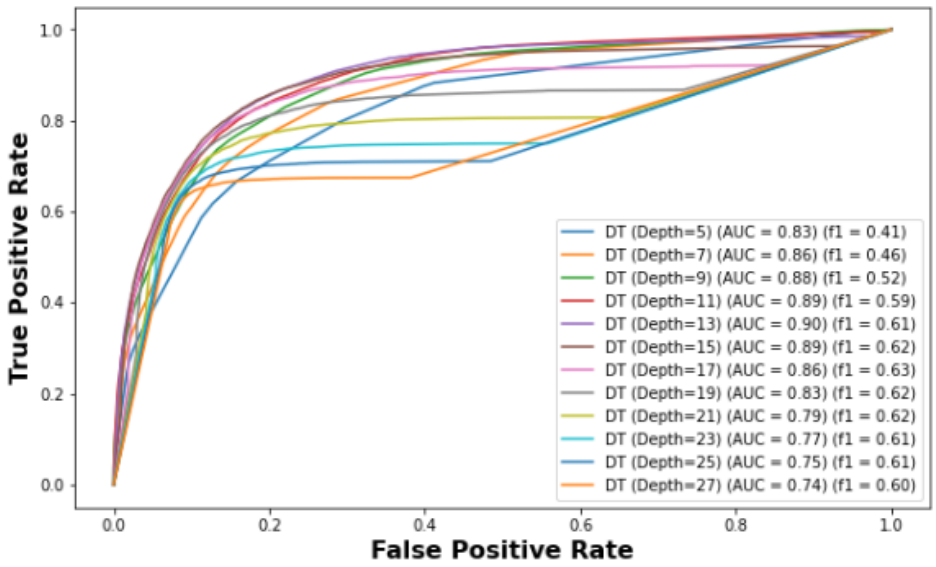


Figura 21: *Curvas ROC para clasificadores de Árbol de Decisión para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, se vuelve a comprobar la gran diferencia entre las métricas AUC y f1. Por otra parte, se observa que a partir de una profundidad de 13, la métrica f1 se estanca y el AUC comienza a disminuir, por lo que se ha realizado una comprobación de overfitting para confirmar si se está produciendo sobreajuste a partir de esa profundidad.

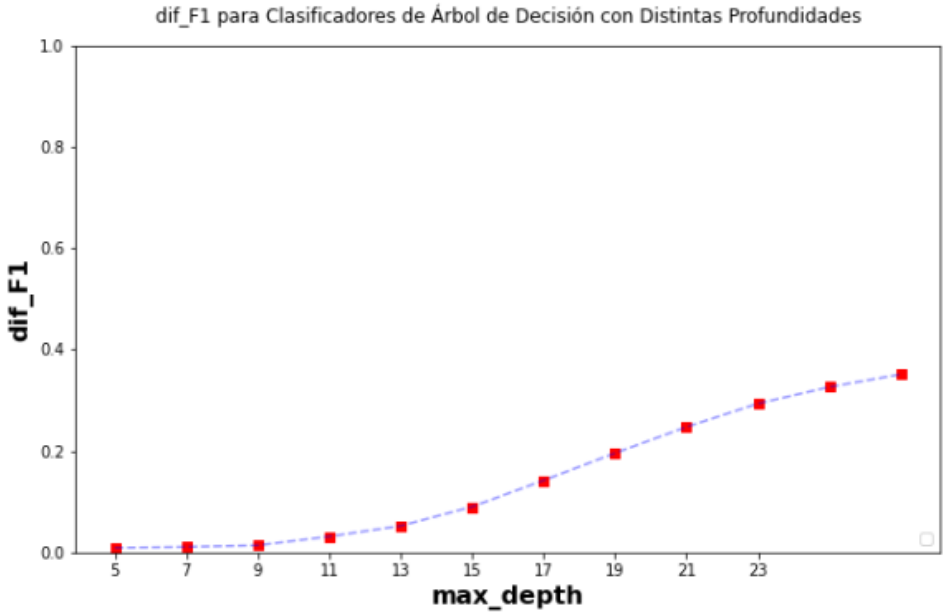


Figura 22: *Comprobación de overfitting para clasificadores de árbol de decision con distintas profundiades para el conjunto DEEP.*

El gráfico muestra que efectivamente el sobreajuste comienza a aumentar a partir de una profundidad de 13, entrando ya en el régimen de overfitting a una profundidad de 17. Por tanto, de nuevo los mejores rendimientos de las métricas, sin incurrir en overfitting, se vuelven a obtener para un árbol con una profundidad máxima de 15 (AUC=0.89, f1=0.62).

Se comprueba que la métrica f1 ha aumentado notablemente al añadir las columnas de color. Para indagar en cómo ha evolucionado la relación entre la precision y la sensibilidad del modelo elegido (Árbol

de Decisión de profundidad 15), se representa su curva PR:

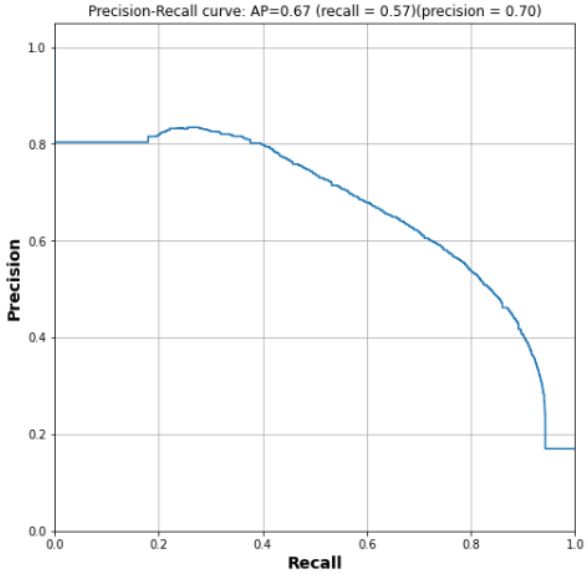


Figura 23: *Curva PR de clasificador de árbol de decisión con una profundidad máxima de 15. Además, también se incluye el Average Precision(AP), la precision y la sensibilidad (recall) del modelo.*

Se aprecia que la adición de columnas de color ha tenido un impacto muy positivo en la sensibilidad del modelo, pasando de 0.40 a 0.57, y que este es el motivo por el que la métrica f1 ha experimentado un incremento significativo, pasando de 0.50 a 0.62.

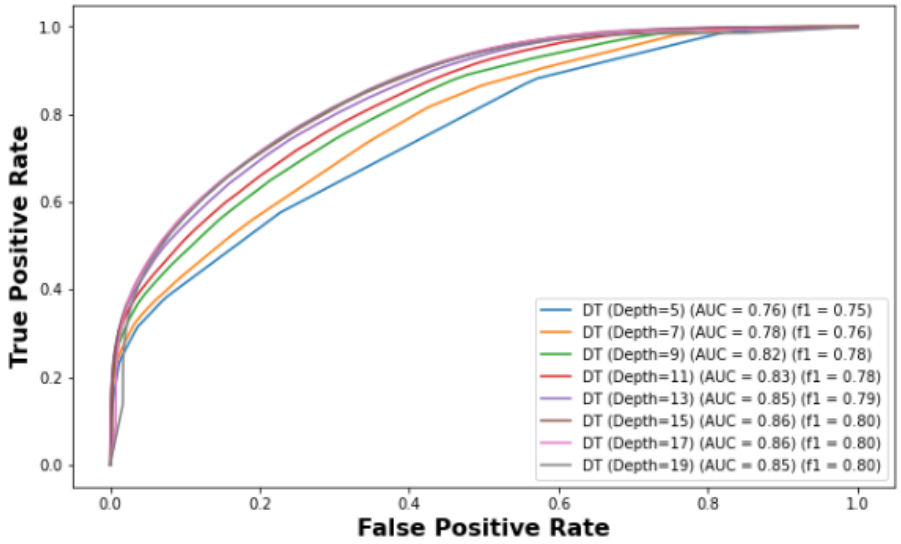


Figura 24: *Curvas ROC para clasificadores de Árbol de Decisión con distintas profundidades para el conjunto WIDE. Además, también se incluyen los valores de las métricas AUC y f1.*

En el WIDE, se vuelve a observar que los rendimientos de las métricas son mucho más parejos en el DEEP y que ambas mejoran con el aumento de la profundidad hasta estancarse a una profundidad de 15. De modo que los mejores resultados se obtienen para un árbol de decisión con una profundidad máxima de 15 (AUC=0.86, f1=0.80).

Nuevamente se observa que la adición de columnas de color se traduce en un aumento considerable del rendimiento de las métricas, pasando de un AUC de 0.78 a 0.85 y de un f1 de 0.74 a 0.80.

Por otra parte, en las figuras 25 y 26 se presentan los resultados correspondientes a los modelos de Random Forest. En ellas se representan la curva ROC, el AUC y la métrica F1 para clasificadores Random Forest con distinto número de árboles.

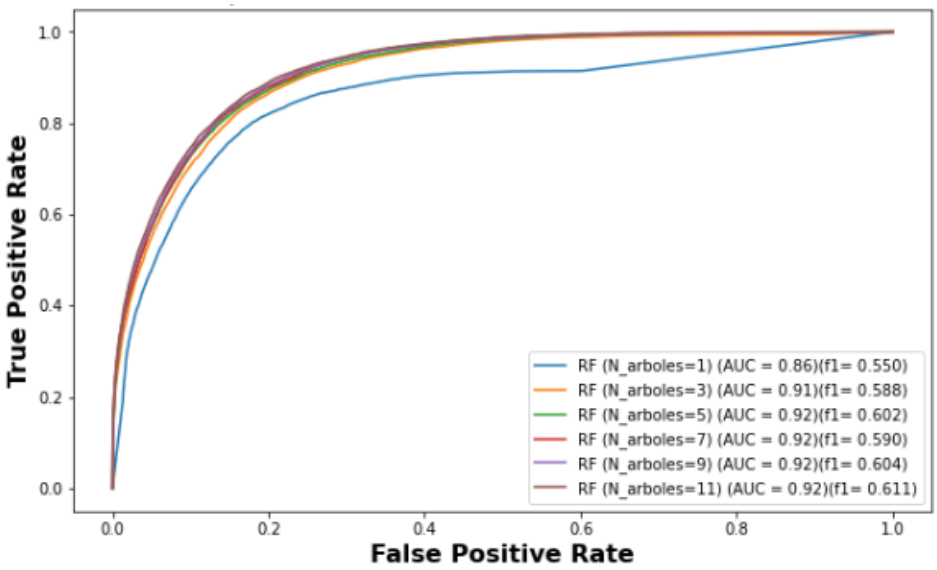


Figura 25: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto DEEP. Además, también se incluyen los valores de las métricas AUC y f1.*

En el caso del DEEP, en primer lugar mencionar que todos los árboles fueron entrenados con una profundidad máxima de 15 para evitar el overfitting. En el gráfico se observa que f1 mejora en general respecto a los modelos de Random Forest sin columnas de colores, alcanzando su mayor rendimiento con un bosque de 11 árboles (AUC=0.92, f1=0.611). Cabe destacar que aunque el AUC se estanca desde los 5 árboles, no se produce overfitting en ningún caso.

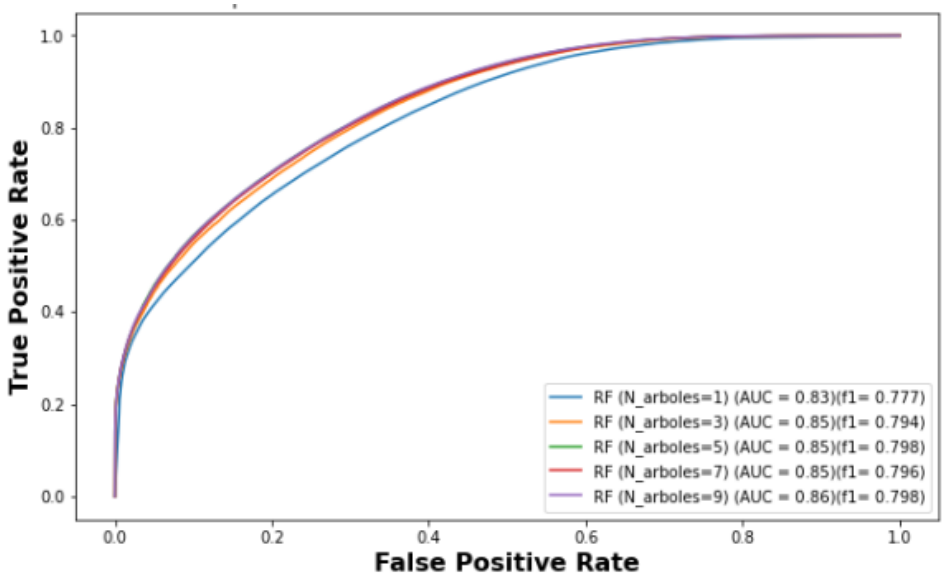


Figura 26: *Curvas ROC para clasificadores Random Forest con distinto número de árboles para el conjunto WIDE. Además, también se incluyen los valores de las métricas AUC y f1.*

En el WIDE, los mejores rendimientos de las métricas (aunque en todos los casos son muy similares) se obtienen para un bosque de 9 árboles (AUC=0.86, f1=0.798).

En la tabla 9 se presenta una comparativa de los resultados obtenidos para los modelos entrenados

únicamente con las magnitudes de Euclid y aquellos que incorporan además las columnas de color.

	DT	RF
DEEP	AUC=0.83, f1=0.50 (recall=0.40,precison=0.66)	AUC=0.89, f1=0.514
DEEP COLORES	AUC=0.89 f1=0.62(recall=0.57, precision=0.70)	AUC=0.92, f1=0.611
WIDE	AUC=0.78, f1=0.74	AUC=0.83, f1=0.775
WIDE COLORES	AUC=0.85, f1=0.80	AUC=0.86, f1=0.798

Tabla 9: *Comparativa de resultados obtenidos para modelos entrenados con las mangnitudes de Euclid y para aquellas que además incluyen las columnas de colores.*

En primer lugar, se observa que la adición de columnas de colores ha tenido un impacto positivo en los rendimientos de todos los modelos, especialmente en el modelo de Decision Tree del conjunto DEEP, donde la sensibilidad ha pasado de 0.40 a 0.57. Por otra parte, se ha visto que la profundidad de los árboles es el hiperparámetro más determinante de los modelos de Decision Tree y Random Forest, siendo el número de árboles de los bosques muy poco relevante, ya que el aumento de estos a penas se traduce en un incremento de los rendimientos de las métricas, especialmente de f1. Además, se observa que para los modelos del WIDE los rendimientos de la métricas son mucho más parejos que los del DEEP. De esta forma, se tiene que los mejores rendimientos para la clasificación de fuentes se han obtenido con modelos de Decision Tree entrenados con las magnitudes de Euclid y las columnas de colores. En general, se puede afirmar que el clasificador obtenido para el conjunto WIDE es bastante bueno, alcanzando una precisión de 0.75 y una sensibilidad de 0.84, mientras que el del DEEP es un clasificador de fuentes menos eficiente, con una precisión de 0.57 y una sensibilidad de 0.70.

Por último, cabe destacar que la selección de las columnas de color adicionales se basó en su cantidad y no en su identidad específica. Para llegar a esta conclusión, se fueron añadiendo una tras otra, comprobando en cada paso los resultados de las métricas , hasta llegar a seis. Este proceso se repitió tres veces para ordenamientos distintos, obteniendo en todos los casos los mismos resultados: El incrementó más significativo de las métricas se daba al pasar de 2 a 3 columnas de colores y, a partir de ahí, no se obtenía una mejoría en sus rendimientos.

8 Conclusiones

En este trabajo nos hemos propuesto como objetivo comprobar si con técnicas de ML se pueden identificar eficientemente galaxias y AGNs utilizando datos simulados de dos proyectos de cartografiado muy importantes, Euclid y LSST, que próximamente harán públicas sus observaciones.

Para ello, en primer lugar he realizado una investigación general sobre conceptos básicos de cosmología y astronomía, incluyendo un estudio más exhaustivo de galaxias y AGNs, así como de técnicas tradicionales de detección de AGNs y técnicas de detección másiva de galaxias mediante cartografiados. Tras ello, he descargado datos simulados de dos de estos grandes cartografiados y he utilizado TOPCAT, un software de visualización y manipulación de tablas astronómicas, para visualizarlos y familiarizarme con ellos. Además, he usado STILTS, que es su versión en línea de comandos, para seleccionar las fuentes que pertenecen a las áreas comunes de ambos cartografiados y para filtrar las que no van a ser detectadas en los cartografiados reales. Para introducir estos comandos también he tenido que aprender el funcionamiento básico de la Terminal de Windows. A continuación, he usado Jupyter Notebook, un entorno de programación de Pyhton, para preparar las muestras e implementar los modelos de ML. Para ello, he utilizado librerías conocidas de python (Numpy y Pandas) para cargar los datos simulados, etiquetarlos y hacer una selección conveniente de sus variables. Por último, he usado la librería sckit-learn de python para entrenar y evaluar los modelos obtenidos de clasificación de galaxias, y he extraído las siguientes conclusiones:

En primer lugar, no hemos podido utilizar las magnitudes de LSST porque al no incluir ruido pre-

sentan unos patrones internos que evidencian la clasificación de las fuentes, de modo que nos hemos quedado solamente con las magnitudes de los filtros de Euclid. Además, hemos descubierto que introduciendo columnas de colores se obtienen mejores resultados en la clasificación. Por otra parte, hemos verificado que los algoritmos de Random Forest no ofrecen mejoras significativas en el desempeño de las métricas respecto a los modelos de Decision Tree, destacándose la profundidad máxima de los árboles como el hiperparámetro más influyente en ambos casos. Con todo, los mejores modelos de clasificación que hemos desarrollado alcanzan una precisión de 0.57 y una sensibilidad de 0.70 para el conjunto DEEP, y una precisión de 0.75 y una sensibilidad de 0.84 para el conjunto WIDE.

9 Bibliografía

[1] Karttunen, H.(2017) *Fundamental Astronomy* , Sixth edition, Springer, Berlin.

[2] Página oficial de Euclid (20/9/2023) [En linea]. Disponible: https://www.cosmos.esa.int/web/euclid/euclid-survey

[3] Página oficial de LSST (25/9/2023) [En linea]. Disponible: https://www.lsst.org/

[4] L. Bisigello, C. Gruppioni, A.Feltre, et al. (2021) *Simulating the infrared sky with a SPRITZ* Astronomy and astrophysics (Berlin), vol. 651, pp.A52

[5] Página oficial de SPRITZ (26/9/2023) [En linea]. Disponible: http://spritz.oas.inaf.it/

[6] Página oficial de TOPCAT (1/10/2023) [En linea]. Disponible: http://adsabs.harvard.edu/abs/2005ASPC..347...29T

[7] Página oficial de STILTS (1/10/2023) [En linea]. Disponible: https://www.star.bris.ac.uk/~mbt/stilts/

[8] C.Müller & Sarah Guido (2017) *Introduction to Machine learning with python* ,cap. 1-2

[9] Hastie,Tibshirani y Friedman (2001) *The elements of statistical learning* , Second edition

[10] Thorn, J. (2021) *Decision Trees Explained.* [En linea]. Disponible: https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6

[11] Alejandro Nieto Jeux (2021 *Algoritmos de aprendizaje automático. Un estudio de su difusión y utilización.* TFG, Grado de Ingeniería Informática, UPM

[12] Yiu, T. (2019) *Understanding of Random Forest.* [En linea]. Disponible: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[13] Ramiréz, J. (2018) *Curvas PR y ROC.* [En linea]. Disponible: https://medium.com/bluekiri/curvas-pr-y-roc-1489fbd9a527

[14] *Oblicuidad del Sol (6/11/2023)* [En linea]. Disponible: https://es.wikipedia.org/wiki/Oblicuidad_de_la_ecl%C3%ADptica

A Apéndice

A.1 Comandos de Stilts

Aquí se incluyen los datos tal como aparecen en el archivo de texto.

```
#COMANDOS DE LÍNEA DE STILTS UTILIZADOS EN LA PREPARACIÓN DE LOS DATOS:

#1-Correlación en ID de los cartografiados LSST y Euclid: deep y wide.

#DEEP(n_fuentes=3.720.974):
java -jar c:\stilts.jar tmatch2 in1=d:\FÍSICA\cat\LSST_deep.fits in2=d:\FÍSICA\cat\Euclid_deep.fits matcher=exact values1="ID" values2="ID"
 out=d:\FÍSICA\cat\match_deep.fits

#WIDE (n_fuentes = 11.983.762):
java -jar c:\stilts.jar tmatch2 in1=d:\FÍSICA\cat\LSST_wide.fits in2=d:\FÍSICA\cat\Euclid_wide.fits matcher=exact values1="ID" values2="ID"
 out=d:\FÍSICA\cat\match_wide.fits

#(Se utiliza la función tmacht2 para correlacionar los catálogos in1 e in2 de acuerdo al valor exacto de la columna "ID" y el catálogo
#resultante se guarda en el fichero especificado por out)


#2-Cribado de las fuentes situadas fuera del área de solapamiento

#(nota: Es importante "escapar" con "^" los carácteres especiales porque si no aparece un error al introducir el comando de línea.
#ej: cmd="select "random^<0.25"" )

#DEEP:
java -jar c:\stilts.jar tpipe in="d:\match_deep.fits" cmd ="addcol random\"random(index)"" cmd="select \"random^<0.25""
 out="d:\FÍSICA\cat\deep_025.fits"

#El resultado de este comando da lugar a 930.107 fuentes.

#WIDE:
java -jar c:\stilts.jar tpipe in=d:\FÍSICA\cat\match_wide.fits cmd ="addcol random \"random(index)"" cmd = "select \"random^<0.50""
 out = d:\FÍSICA\cat\wide_050.fits

#El resultado de este comando da lugar a 5.992.134 fuentes.

#(Se utiliza la función tpipe para manipular las tablas. En primer lugar se crea una nueva columna que asigna a cada fuente
#un número aleatorio entre 0 y 1 y luego se seleccionan las fuentes cuyo valor es inferior a 0.25 (deep) o 0.50 (wide)
# Los catálogos resultantes del cribado de fuentes fuera del área de solapamiento se guardan en el fichero especificado por out).


#3-Conversión de flujos de microjanskys a magnitudes AB

#DEEP Y WIDE:

java -jar c:\stilts.jar tpipe in=d:\match_deep.fits cmd="addcol Euclid_vis_ab \"janskyToAb(Euclid_vis*1.e-6)\""
cmd="addcol Euclid_Y_ab \"janskyToAb(Euclid_Y*1.e-6)\"" cmd="addcol Euclid_J_ab \"janskyToAb(Euclid_J*1.e-6)\""
cmd="addcol Euclid_H_ab \"janskyToAb(Euclid_H*1.e-6)\" "cmd="addcol LSST_u_ab \"janskyToAb(LSST_u*1.e-6)\""
cmd="addcol LSST_g_ab \"janskyToAb(LSST_g*1.e-6)\"" cmd="addcol LSST_r_ab \"janskyToAb(LSST_r*1.e-6)\""
cmd="addcol LSST_i_ab \"janskyToAb(LSST_i*1.e-6)\"" cmd="addcol LSST_z_ab \"janskyToAb(LSST_z*1.e-6)\""
cmd="addcol LSST_y_ab \"janskyToAb(LSST_y*1.e-6)\""  out=d:\flujos_ab.fits

#comando es el mismo para el wide y el deep.

#(Se añaden nuevas columnas, una por cada filtro de LSST y Euclid, con los valores de los flujos de los filtros en magnitudes AB.
#Debido a que los flujos vienen en microjankys, primero se pasan a janskys multiplicando por 1e-6 y luego la función janskysToAb()
#realiza la conversión a magnitudes AB.
```

A.2 Filtrado de Flujos límite con python

Aquí se muestra el código de python con el que se ha realizado el filtrado de flujos límite. En concreto, se presenta el filtrado realizado para el cartografiado DEEP.

```
import numpy as np
import pandas as pd

#carga el cartografiado DEEP con los flujos de los filtros ya en mag AB.
df = pd.read_csv("deep_ab.csv")

#Filtrado de flujos limites de LSST y Euclid para el caso DEEP.
#Sustituye los valores por encima del flujo limite por NaNs.
flujos_lim = [26.5,25.24,25.24,25.24,28.5,28.7,28.9,28.4,28.0,27.0]
for i in range (47,df.shape[1]):
    col = df.columns[i]
    df[col] = np.where(df[col] > flujos_lim[i-47], np.nan, df[col])

#Eliminamos todas aquellas filas que tengan alg n NaN, es decir,
#nos quedamos solo con aquellas fuentes que tengan registro
#en todos los filtros tras el proceso de filtrado
columnas_rango = df.columns[47:df.shape[1]]
df= df.dropna(subset=columnas_rango)
#El dataset resultante se guarda en formato .csv
df.to_csv("deep_sin_nan.csv")
```

A.3 Desagreagación de fuentes con python

Aquí se muestra el código de python con el que se llevó a cabo la desagregación de fuentes de cartografiado DEEP.

```
#DESAGREGACI N DE FUENTES DEL CARTOGRAFIADO DEEP.

#Importar librer as
import numpy as np
import pandas as pd

#Cargamos el dataframe, en este caso el cartografiado deep con los filtros de
    ↪ Euclid y LSST
df_deep = pd.read_csv("deep_sin_nan.csv")

#log10N: 27, Euclid_vis_ab: 48, LIR_AGN = 19

selected_columns = df_deep.iloc[:,[48,49,50,51,27,19]]
new_deep = selected_columns.copy()

#Cambiamos el nombre del dataset por comodidad.
X = new_deep
#Sustituimos por 1 las entradas d nde haya detecci n AGN.
X.loc[X["LIR_AGN"]>0,["LIR_AGN"]]=1

#Desagregaci n de fuentes a partir de la asignaci n de pesos:  log10N<2: 1 ,
    ↪ log10N>2: 10**(log10N - 2)
X.loc[X["log10N"]<2,"log10N"] = 1
X.loc[X["log10N"]>=2,"log10N"] = 10**(X["log10N"]-2)
#n mero total de fuentes
num_total = X["log10N"].sum()
```

```
#n mero total de AGNs
num_agn = X["log10N"][X["LIR_AGN"]==1].sum()
#n mero total de NO_AGNs
num_no_agn = X["log10N"][X["LIR_AGN"]==0].sum()

#fracci n y porcentaje de AGNs
frac_agn = num_agn/num_total
frac_no_agn = num_no_agn/num_total

print("N mero de AGNs: ", "{:.0f}".format(num_agn))
print("N mero de No-AGNs: ","{:.0f}".format(num_no_agn))
print("N mero total de fuentes: ", "{:.0f}".format(num_total))
print("Fracci n de AGNs :", "{:.3f}".format(frac_agn))
```

A.4 Código implementado para la creación de los modelos

Aquí se incluye el código de python que ha sido implementado para construir todos los modelos de clasificación. En particular, se muestra el código correspondiente a los modelos del conjunto DEEP entrenados con las magnitudes de Euclid y las columnas de colores.

DEEP_COLORES

November 13, 2023

[1]:
```
#Importar librerías
import numpy as np
import pandas as pd

#Cargamos el dataframe, en este caso el cartografiado deep con los filtros de␣
 ↪Euclid y LSST
df_deep = pd.read_csv("deep_sin_nan.csv")
```

[2]:
```
#Creamos el vector de etiquetas y (contiene las categorías reales(AGN/NO_AGN)␣
 ↪de todas las fuentes del dataset X).
X = df_deep
a = df_deep["LIR_AGN"]

y = np.zeros(len(a))

for i in range(len(a)):
    if a[i] >0:
        y[i] =1
```

[3]:
```
#Seleccionamos las características del dataset que corresponden únicamente a␣
 ↪los flujos de los filtros de EUCLID

X_flujos_eu = X[X.columns[48:52]]
```

[17]:
```
#Añadimos las columnas de colores: V-Y,Y-J,J-H

X_flujos_eu['V_Y']= X_flujos_eu.iloc[:, 0] - X_flujos_eu.iloc[:, 1]
X_flujos_eu['Y_J']= X_flujos_eu.iloc[:, 1] - X_flujos_eu.iloc[:, 2]
X_flujos_eu['J_H']= X_flujos_eu.iloc[:, 2] - X_flujos_eu.iloc[:, 3]
```

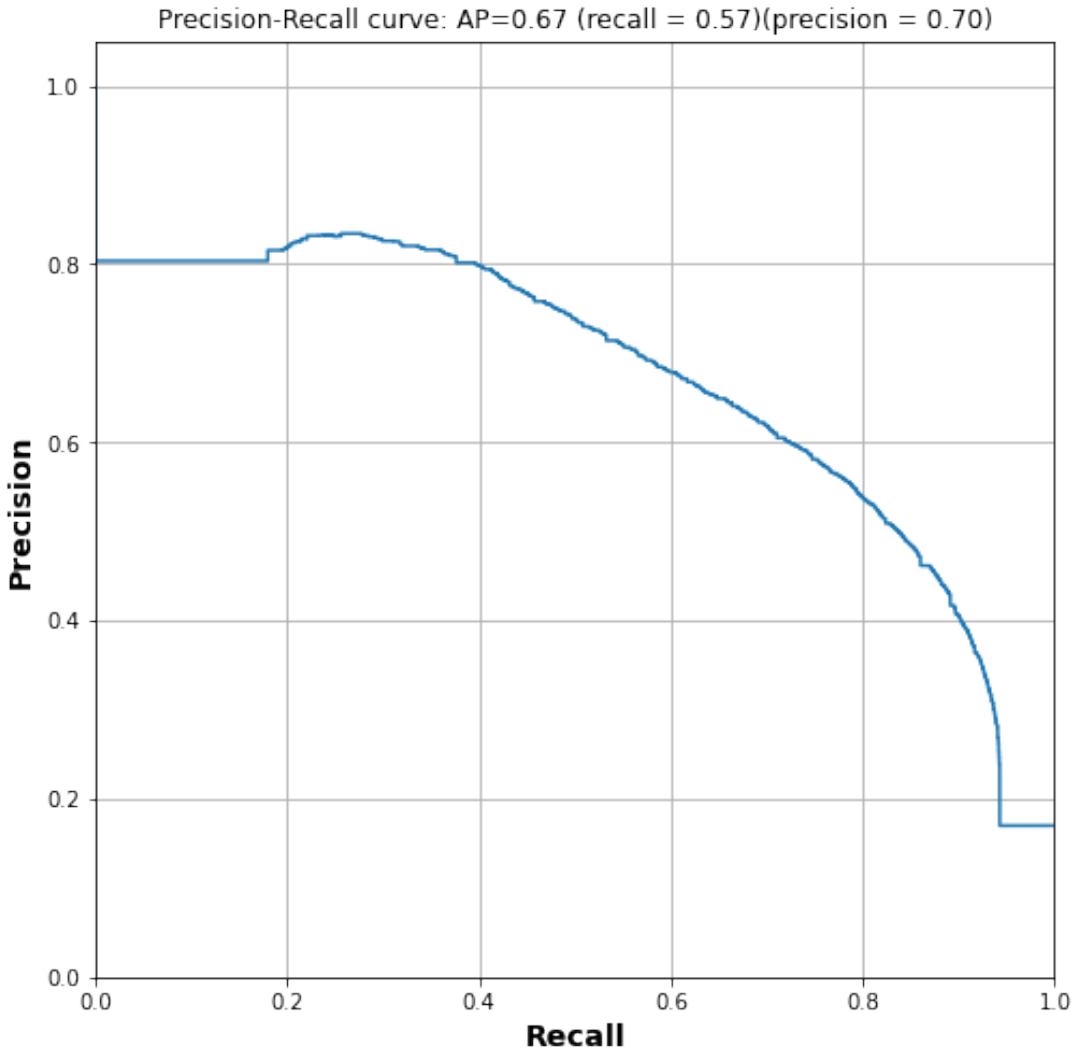
```
<ipython-input-17-8a96665b2177>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_flujos_eu['V_Y']= X_flujos_eu.iloc[:, 0] - X_flujos_eu.iloc[:, 1]
<ipython-input-17-8a96665b2177>:4: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_flujos_eu['Y_J']= X_flujos_eu.iloc[:, 1] - X_flujos_eu.iloc[:, 2]
<ipython-input-17-8a96665b2177>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_flujos_eu['J_H']= X_flujos_eu.iloc[:, 2] - X_flujos_eu.iloc[:, 3]
```

[5]:
```
#5-Dividimos el dataset en cojuntos de entrenamiento y validación
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X_flujos_eu,y, test_size=0.
 ↪2,stratify=y,random_state=42)

#En este caso el 20% de los datos se destinará al conjunto de validación.
#El atributo stratify=y asegura que la distribución de las etiquetas en los␣
 ↪conjuntos de entrenamiento y
#validación sea la misma que en el conjunto original, lo que resulta␣
 ↪especialmente útil en los datasets
# con clases desbalanceadas (o desequilibradas), como es el caso.
```

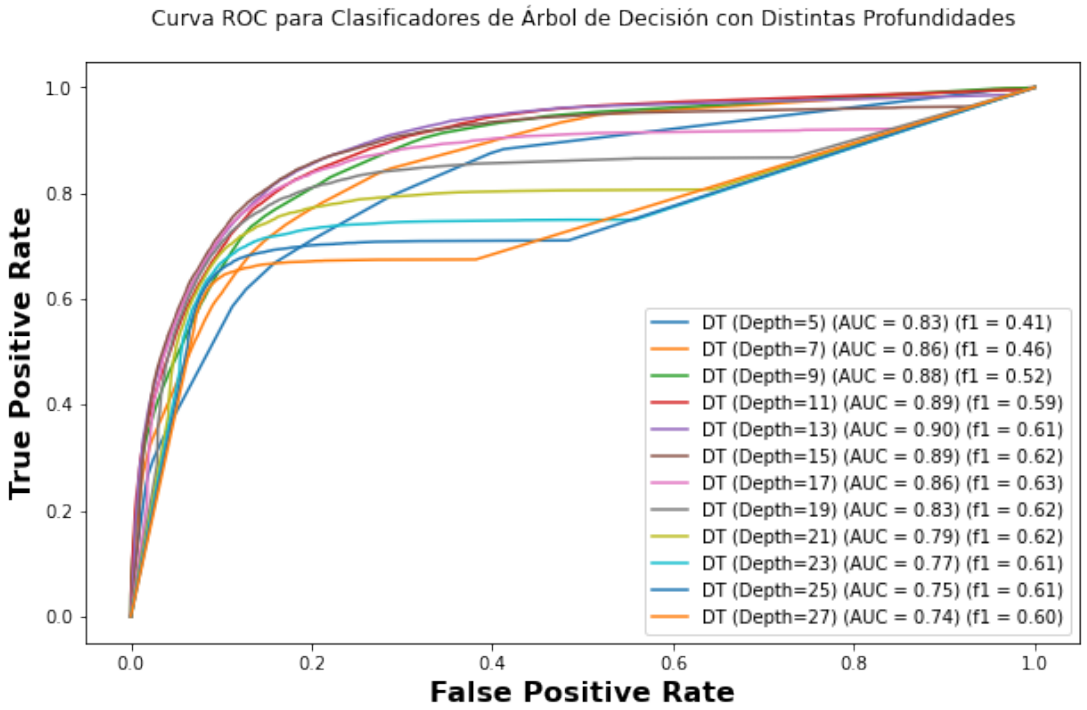
[16]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve,␣
 ↪average_precision_score,recall_score
from sklearn.metrics import precision_score
from sklearn import tree

clf = tree.
 ↪DecisionTreeClassifier(criterion="entropy",max_depth=15,random_state=42)
clf = clf.fit(X_train,y_train)

# Obtener las probabilidades de la clase positiva
y_proba = clf.predict_proba(X_test)[:, 1]
y_pred = clf.predict(X_test)

# Calcular la precisión y la exhaustividad para varios umbrales
precision, recall, _ = precision_recall_curve(y_test, y_proba)

# Calcular el score de precisión promedio
average_precision = average_precision_score(y_test, y_proba)
```

```
rec = recall_score(y_test,y_pred)
pr = precision_score(y_test,y_pred)

# Graficar la curva PR
plt.figure(figsize=(8, 8))
plt.step(recall, precision, where='post')

plt.xlabel('Recall',fontsize=14,fontweight="bold")
plt.ylabel('Precision',fontsize=14,fontweight="bold")
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title(f'Precision-Recall curve: AP={average_precision:0.2f} (recall = {rec:.
  ↪2f})(precision = {pr:.2f})',fontsize=12)
plt.grid(True)
plt.show()
```



[119]:
```
# CURVA ROC DE CLASIFICADORES DT CON DISTINTAS PROFUNDIDADES
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.metrics import roc_curve, auc,f1_score



# Profundidades que deseas probar
depths = list(range(5,29,2))

# Crear un plot para las curvas ROC
plt.figure(figsize=(10, 6))

for depth in depths:
    # Crear el clasificador de árbol con la profundidad dada
    clf = tree.DecisionTreeClassifier(max_depth=depth, random_state=42)
    clf.fit(X_train, y_train)

    # Obtener las probabilidades predichas para la clase positiva
    y_pred_proba = clf.predict_proba(X_test)[:, 1]
    y_pred = clf.predict(X_test)

    # Calcular la curva ROC y el AUC
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)
    #Calcular f1_score
    f1 = f1_score(y_pred,y_test)

    # Agregar la curva ROC al plot
    plt.plot(fpr, tpr, label=f'DT (Depth={depth}) (AUC = {roc_auc:.2f}) (f1 =␣
 ↪{f1:.2f})')

# Configurar los ejes y la leyenda
plt.xlabel('False Positive Rate',fontsize=16,fontweight="bold")
plt.ylabel('True Positive Rate',fontsize=16,fontweight="bold")
plt.title('Curva ROC para Clasificadores de Árbol de Decisión con Distintas␣
 ↪Profundidades',pad=20)
plt.legend(loc='lower right')

# Mostrar el plot
plt.show()
```

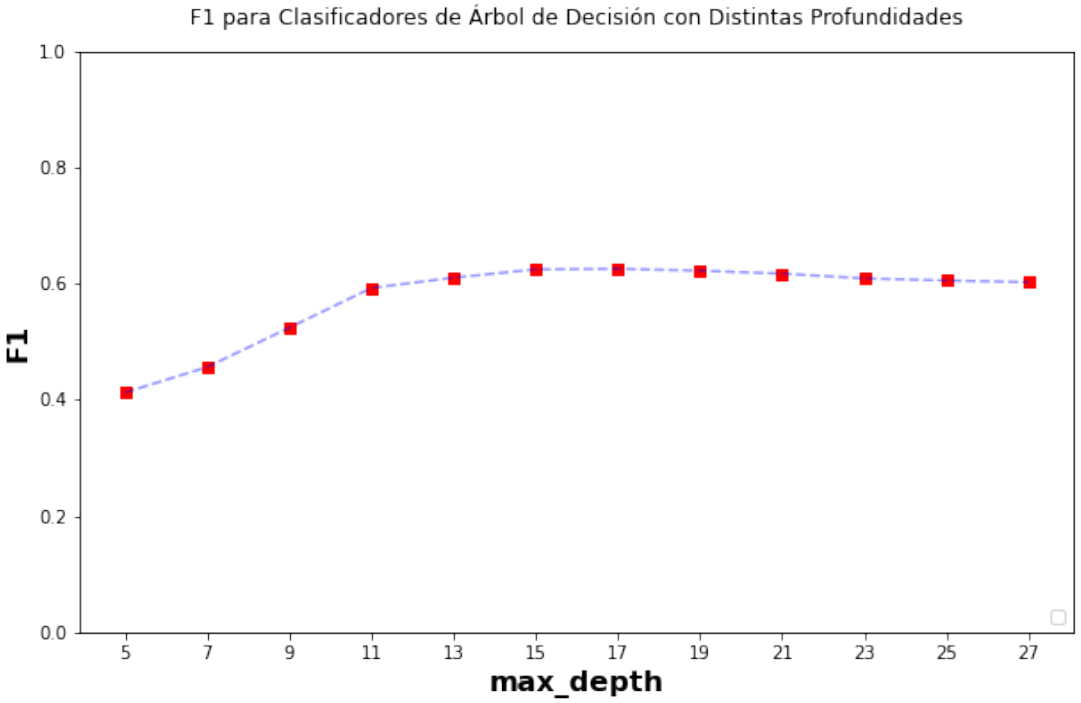
[119]:
```
<matplotlib.legend.Legend at 0x145b127dfd0>
```



[118]:
```python
# F1 VS MAX_DEPTH PARA CLASIFICADORES DT CON DISTINTAS PROFUNDIDADES
from sklearn.metrics import f1_score

# Profundidades que deseas probar
depths = list(range(5,29,2))

# Crear un plot para las curvas ROC
plt.figure(figsize=(10, 6))

f1_test = []
f1_train = []
for depth in depths:
    # Crear el clasificador de árbol con la profundidad dada
    clf = tree.DecisionTreeClassifier(max_depth=depth, random_state=42)
    clf.fit(X_train, y_train)
    y_pred_test= clf.predict(X_test)
    y_pred_train = clf.predict(X_train)

    f1_test.append(f1_score(y_pred_test,y_test))
    f1_train.append(f1_score(y_pred_train,y_train))

# Configurar los ejes y la leyenda
plt.plot(depths, f1_test,color="blue",linestyle="--",alpha=0.4)
```

```
plt.scatter(depths,f1_test,color="red",marker="s")
plt.ylim(0,1)
plt.xlabel('max_depth', fontsize=16, fontweight='bold')
plt.ylabel('F1', fontsize=16, fontweight='bold')

plt.xticks(np.arange(5,29,2))
plt.title('F1 para Clasificadores de Árbol de Decisión con Distintas␣
 ↪Profundidades',fontsize=12,pad=15)
plt.legend(loc='lower right')

plt.show()
```

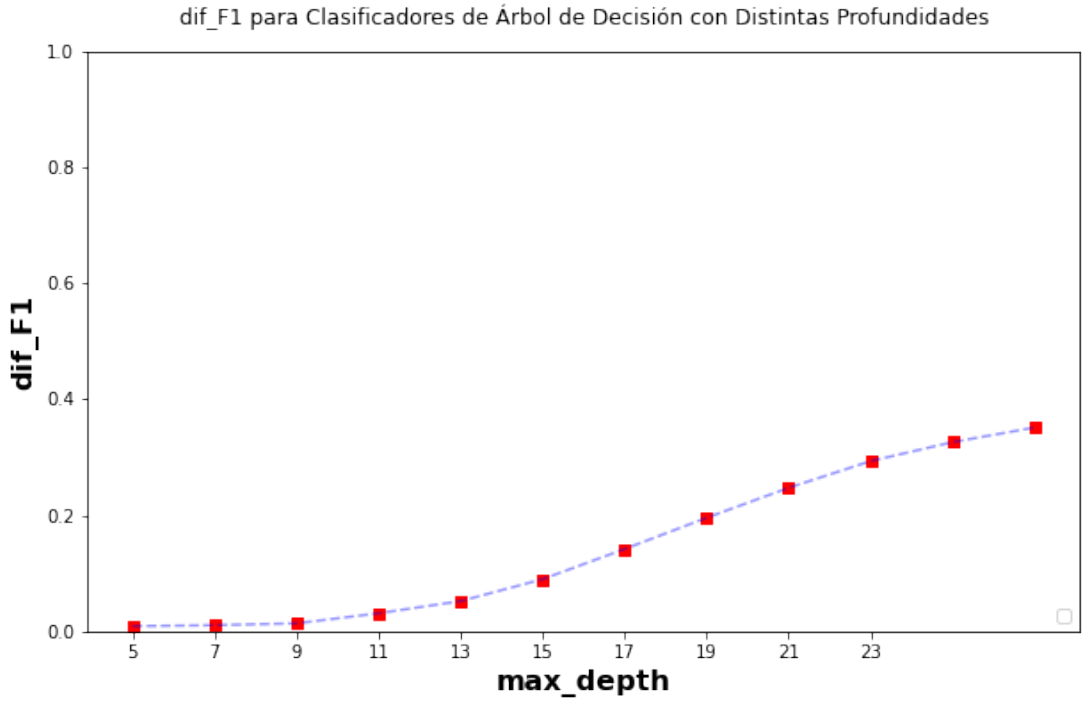
```
No handles with labels found to put in legend.
```



[120]:
```
#test de overfitting

dif_f1 = []
for i in range(len(f1_test)):
    dif_f1.append(np.abs(f1_test[i]-f1_train[i]))


plt.figure(figsize=(10, 6))

plt.plot(depths,dif_f1,color="blue",linestyle="--",alpha=0.4)
```

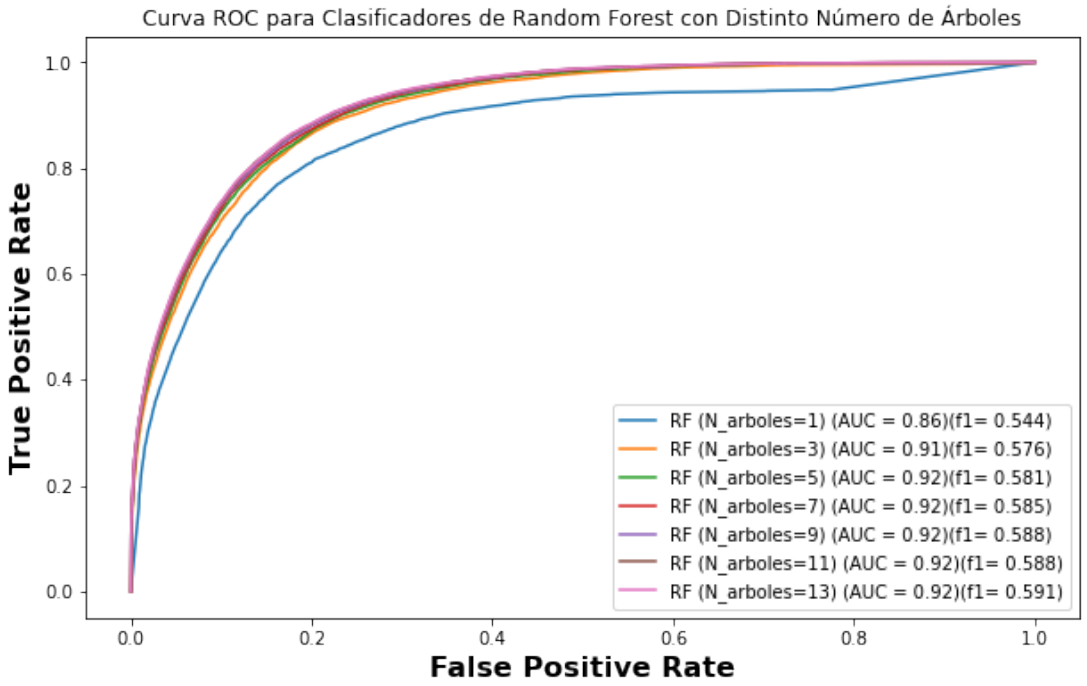
```
plt.scatter(depths,dif_f1,color="red",marker="s")
plt.ylim(0,1)
plt.xlabel('max_depth', fontsize=16, fontweight='bold')
plt.ylabel('dif_F1', fontsize=16, fontweight='bold')

plt.xticks(np.arange(5,24,2))
plt.title('dif_F1 para Clasificadores de Árbol de Decisión con Distintas␣
 ↪Profundidades',fontsize=12,pad=15)
plt.legend(loc='lower right')

plt.show()

#Se demuestra que no hay overfitting
```

```
No handles with labels found to put in legend.
```



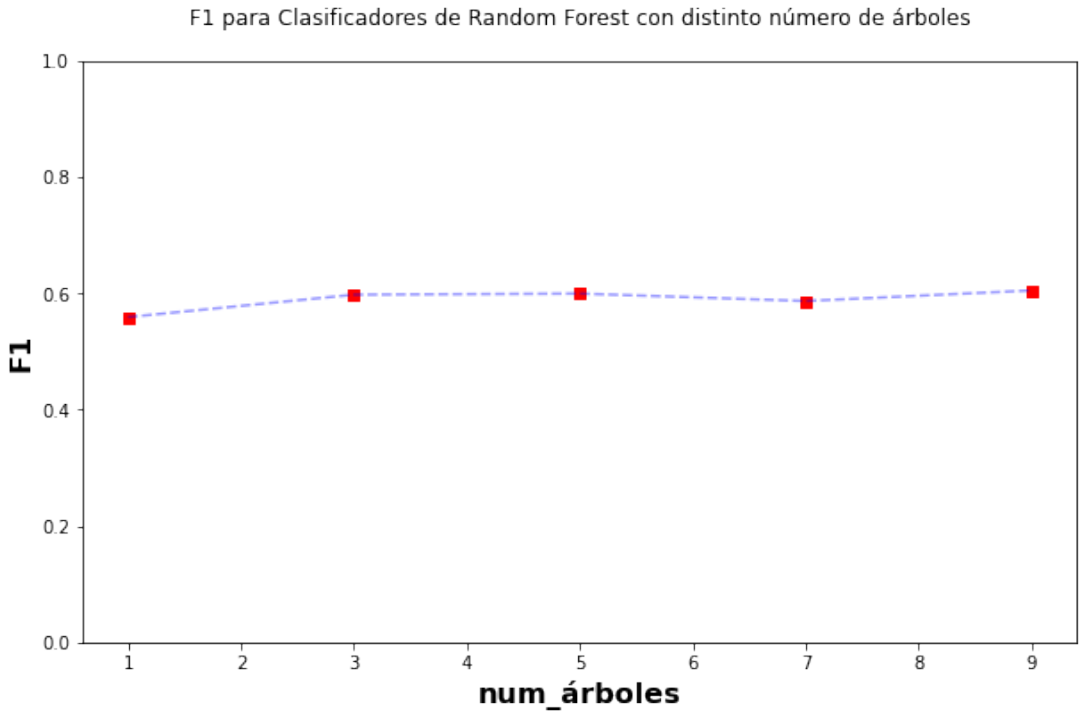
[80]:
```
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.metrics import roc_curve, auc,f1_score
from sklearn.ensemble import RandomForestClassifier



# Profundidades que deseas probar
```

```
n_arboles = list(range(1,15,2))

# Crear un plot para las curvas ROC
plt.figure(figsize=(10, 6))


for arbol in n_arboles:
    # Crear el clasificador de árbol con la profundidad dada
    clf_rf = RandomForestClassifier(criterion="entropy",n_estimators =␣
 ↪arbol,max_depth=15,random_state=42)
    clf_rf.fit(X_train, y_train)

    # Obtener las probabilidades predichas para la clase positiva
    y_pred_proba = clf_rf.predict_proba(X_test)[:, 1]
    y_pred= clf_rf.predict(X_test)
    # Calcular la curva ROC y el AUC
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)
    f1 = f1_score(y_pred,y_test)

    # Agregar la curva ROC al plot
    plt.plot(fpr, tpr, label= f'RF (N_arboles={arbol}) (AUC = {roc_auc:.
 ↪2f})(f1= {f1:.3f})')

# Configurar los ejes y la leyenda
plt.xlabel('False Positive Rate',fontsize=16, fontweight='bold')
plt.ylabel('True Positive Rate',fontsize=16, fontweight='bold')
plt.title('Curva ROC para Clasificadores de Random Forest con Distinto Número␣
 ↪de Árboles')
plt.legend(loc='lower right')

# Mostrar el plot
plt.show()
```



[106]:
```
#GRÁFICAS DE F1 PARA RF CON DISTINTO NÚMERO DE ÁRBOLES

from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier


# Profundidades que deseas probar
n_arboles = list(range(1,11,2))

# Crear un plot para las curvas ROC
plt.figure(figsize=(10, 6))

f1_train_rf = []
f1_test_rf = []
for arbol in n_arboles:
    clf_rf = RandomForestClassifier(criterion="entropy",n_estimators =␣
 ↪arbol,max_depth=15)
    clf_rf.fit(X_train, y_train)
    y_pred_test = clf_rf.predict(X_test)
    y_pred_train = clf_rf.predict(X_train)

    f1_test_rf.append(f1_score(y_pred_test,y_test))
    f1_train_rf.append(f1_score(y_pred_train,y_train))
```

```
# Configurar los ejes y la leyenda
plt.plot(n_arboles, f1_test_rf,color="blue",linestyle="--",alpha=0.4)
plt.scatter(n_arboles,f1_test_rf,color="red",marker="s")
plt.ylim(0,1)
plt.xlabel('num_árboles', fontsize=16, fontweight='bold')
plt.ylabel('F1', fontsize=16, fontweight='bold')
plt.title('F1 para Clasificadores de Random Forest con distinto número de␣
 ↪árboles',pad=20)


# Mostrar el plot
plt.show()
```



[107]:
```
#Comprobación de overfitting
dif_f1 = []
for i in range(len(f1_test_rf)):
    dif_f1.append(np.abs(f1_test_rf[i]-f1_train_rf[i]))


plt.figure(figsize=(10, 6))
```

```
plt.plot(n_arboles,dif_f1,color="blue",linestyle="--",alpha=0.4)
plt.scatter(n_arboles,dif_f1,color="red",marker="s")
plt.ylim(0,1)
plt.xlabel('n_arboles', fontsize=16, fontweight='bold')
plt.ylabel('dif_F1', fontsize=16, fontweight='bold')

plt.xticks(np.arange(1,11,2))
plt.title('dif_F1 para Clasificadores de Random Forest con distinto número de␣
 ↪árboles',fontsize=12,pad=15)
plt.legend(loc='lower right')

plt.show()

#Se demuestra que no hay overfitting
```

```
No handles with labels found to put in legend.
```