TFG/Física/2019-20

APLICACIÓN DE MODELOS DE MACHINE LEARNING A SIMULACIONES COSMOLÓGICAS: APRENDIZAJE A PARTIR DE SIMULACIONES HIDRODINÁMICAS DE CÚMULOS DE GALAXIAS

Abstract

Los cúmulos de galaxias son las formaciones más masivas del universo. Estos cúmulos están formados en un 85% por materia oscura y en un 15% por materia bariónica. Para tratar con la parte de materia oscura, se pueden realizar simulaciones de N cuerpos mientras que para la parte bariónica se utilizan las simulaciones hidrodinámicas. En este trabajo se ha tratado de entrenar dos algoritmos de machine learning, de random forests y redes neuronales, utilizando bases de datos de simulaciones de N cuerpos de materia oscura junto con su contraparte de simulación hidrodinámica, para que una vez entrenado pueda predecir las propiedades bariónicas de simulaciones de solo materia oscura sin tener que realizar la simulación hidrodinámica correspondiente. Se ha visto que en efecto es posible entrenar los algoritmos para que realicen predicciones con una gran precisión. También se ha estudiado qué variables son las más relevantes para realizar la predicción. Finalmente se han aplicado los algoritmos entrenados a bases de datos obtenidos de simulaciones nuevas y se ha comprobado que los algoritmos son capaces de predecir de manera razonable las variables bariónicas. Estas predicciones también se han realizado sobre simulaciones iguales calculadas con distinta resolución, observando que aunque la resolución de una simulación sea ocho veces menor, no se pierde poder predictivo siendo las predicciones muy similares para las dos resoluciones.

IÑIGO MAIZA RAZKIN

TUTOR: GUSTAVO YEPES

1. INTRODUCCIÓN

Los cúmulos de galaxias son los objetos más masivos del universo, y constituyen un escenario ideal para testear modelos cosmológicos y teorías sobre evolución de galaxias. Estos cúmulos están formados principalmente por materia oscura (DM, por su abreviación en inglés). Esta DM supondría hasta un 85% de la masa de los cúmulos mientras que el componente bariónico, formado por el gas intracúmulo (ICM por su abreviación en inglés) y las estrellas y galaxias, compondría el 15% total de la masa. Teniendo en cuenta que hasta día de hoy no se ha observado la materia oscura, debido a que esta solo interactúa mediante su contribución gravitatoria, simulaciones de N cuerpos pueden describir las componentes de DM de los cúmulos. Para la componente bariónica, debido a la complejidad de la física involucrada, ha de tratarse con simulaciones hidrodinámicas numéricas.

Las simulaciones hidrodinámicas son fundamentales para estudiar la formación y evolución de los cúmulos de galaxias (Borgani & Kravtsov 2011 [1]). Estas simulaciones son computacionalmente muy exigentes, de manera que se suele, imitando a las observaciones, crear catálogos de cúmulos de galaxias resimuladas extraídas de las simulaciones de N cuerpos.

Describir la física del gas en simulaciones de cúmulos de galaxias requiere un gran trabajo computacional y un equilibrio en las resolución numérica y estadística y el volumen cosmológico de la simulación. En esta situación, es posible pensar que el machine learning (ML) puede ofrecer una alternativa para obtener información acerca de algunas de las propiedades principales como la temperatura o masa de estrellas. El ML es una subrama de las ciencias de la computación que proporciona una plataforma para aprender relaciones complejas en grandes conjuntos de datos multidimensionales. El ML se ha aplicado en Astronomía obteniendo grandes resultados s (Ball et al. 2006 [2] ; Fiorentin et al. 2007 [3]; Banerji et al. 2010 [4] ; Ball & Brunner 2010 [5] ; Gerdes et al. 2010 [6] ; Kind & Brunner 2013 [7] ; Ivezi´c et al. 2014 [8] ;Ness et al. 2015 [9] ; Kim 2015 [10]; Dieleman et al. 2015 [11]).

El ML también se ha empezado a aplicar en problemas de formación de galaxias y de cúmulos de galaxias. Xu et al. 2011 [12] predijeron el número de galaxias en un DM halo para crear catálogos de galaxias. Para ello usaron los algoritmos de kNN (k-Nearest-Neighbours, en inglés) y de SVM (Simple Vector Machines, en inglés). Las aplicaciones de ML a simulaciones cosmológicas son más recientes. Kamdar, Turk & Brunner, 2016 [13], usaron los “Illustris Simulations” (IL) para explorar la posibilidad de utilizar ML para poblar simulaciones hidrodinámicas. Basado en este trabajo, Federico Sembolini (2018) [14], realizó su tesis de Máster sobre el uso de ML para predecir

propiedades en simulaciones hidrodinámicas haciendo uso de la base de datos MUSIC (www.cosmosin.org).

1. 1. OBJETIVOS Y MOTIVACIÓN

El principal objetivo de este estudio será aplicar técnicas de ML a simulaciones hidrodinámicas de cúmulos de galaxias: usaremos propiedades de cúmulos de galaxias extraídos de simulaciones cosmológicas de solo DM como atributos (o features, variables de entrenamiento) de nuestro set de datos, y por otro lado varias propiedades bariónicas de los mismos objetos resimulados con hidrodinámica y con una mayor resolución como objetivos (o targets, variables a predecir).

Para realizar nuestro estudio necesitábamos unas muestras de simulaciones hidrodinámicas y se selecciono la base de datos MUSIC (Sembolini et.al 2013 [15]). El dataset MUSIC constituye uno de los catálogos más grandes de simulaciones hidrodinámicas de cúmulos de galaxias. Consiste en dos sets de objetos resimulados extraídos de dos simulaciones de N-cuerpos: el MareNostrum Universe y el MultiDark (Prada et al. 2012 [16]). Para la primera parte de este trabajo, se han extraído de la muestra MUSIC-2, que fue creado resimulando todos los objetos más masivos de la simulación MultiDark. Por tanto, la motivación principal para este trabajo es que al entrenar algunos algoritmos con el dataset mencionado, que supone solamente un 0.42% del volumen simulado, el algoritmo aprenda y pueda predecir para todo el volumen algunos targets, ahorrando así una simulación hidrodinámica que supondría un coste computacional enorme.

Una vez se tiene el set de datos preparado, se pretende realizar un estudio previo para determinar la relevancia de cada atributo de cara a la predicción haciendo uso del algoritmo “Random Forests” (RF) que introduciremos más adelante.

El objetivo después será entrenar mediante los algoritmos de RF y redes neuronales (NN, por su abreviatura en inglés) nuestro set de datos y comprobar si los algoritmos son capaces de realizar predicciones precisas.

Una vez se tiene un algoritmo entrenado, se tratarán de realizar predicciones sobre otras simulaciones de solo DM para ver si es capaz de predecir las variables bariónicas de un set de datos nuevo. El set de datos que se utilizará para esta predicción se obtendrá de UNIT- Project (Chuang et al. 2018 [17]). Debido a que las simulaciones de N cuerpos de UNIT están realizadas para distintas resoluciones, también se intentará realizar las predicciones para otras resoluciones de los mismos sets de datos, para ver si nuestro algoritmo entrenado tiene la misma capacidad predictiva para datos de menor resolución.

2. MÉTODOS

Los dos algoritmos de ML aplicados en nuestro estudio han sido los mencionados "Random Forest" (RF) y "Neural Networks" (NN). Estos algoritmos han sido extraídos de la librería de "scikit-learn" (Pedregosa et al. (2011) [18]) y se han utilizado para el entrenamiento del set de datos en un entrenamiento con 5 K-folds. A la hora de realizar el entrenamiento de un algoritmo con un set de datos en ML, estos datos se dividen en dos partes: la parte de entrenamiento y la parte de test. La técnica de K-folds, consiste en dividir el set de entrenamiento en las dos partes mencionadas de manera aleatoria varias veces, para después obtener el rendimiento de nuestro algoritmo como la media de los distintos K-folds. Esta idea se aprecia de manera clara en la *figura 1*.

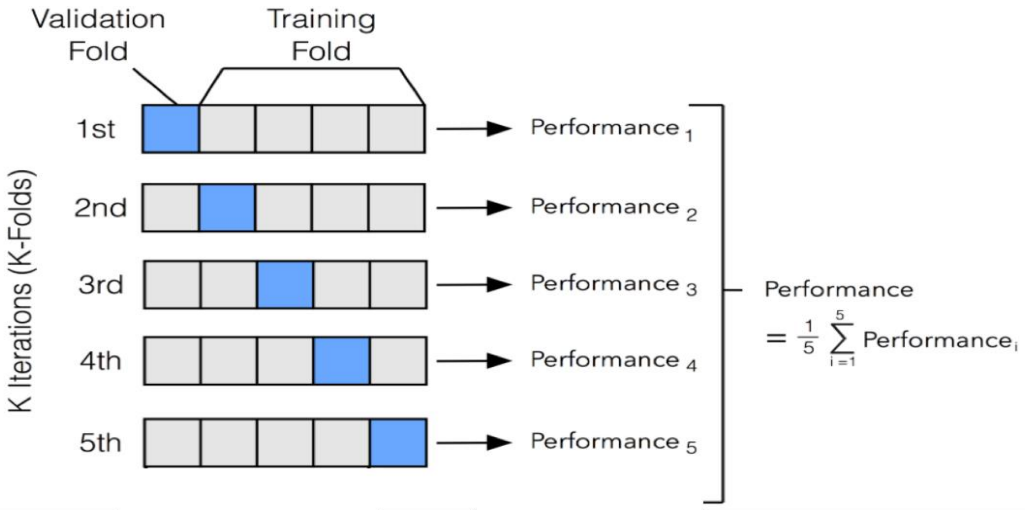

***Figura 1**: Funcionamiento de la técnica de K-Folds.*

2. 1. RANDOM FOREST

Random forest (RF) es una colección de árboles de decisión donde cada árbol da una clasificación y el resultado se compone de la media de los resultados de todos los árboles. RF es el algoritmo más usado en el ML ya que intriduce una gran aleatoriedad en los árboles de entrenamiento, evitando el sobreentrenamiento u *overfitting*. Los parámetros óptimos del algoritmo para nuestro caso (número de árboles, profundidad …) se obtendrán haciendo un "Grid Search" (obtenido de librería scikit-learn) en nuestro set de entrenamiento.

Debido a que en el estudio de la importancia de cada atributo en el entrenamiento (feature importance) haremos uso de RF, tendremos que trabajar con dos tipos distintos de algoritmos en este caso. En primer lugar, el RF monoobjetivo, es decir, que está diseñado y entrenado para predecir un único

target. Esto se debe a que al trabajar con distintos targets, los atributos que más influyan en la predicción no tienen porque ser los mismos para todos los targets.

El RF multiobjetivo, que entrena para todos los targets a la vez, se utilizará en el resto del estudio y a la hora de realizar predicciones sobre nuevos datasets (UNIT).

2. 2. NEURAL NETWORKS

Neural Networks (NN) o redes neuronales, son un algoritmo de ML basado en el funcionamiento de las neuronas cerebrales. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. Es decir, una red neuronal es un grupo interconectado de nodos. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida.

Al igual que en RF haremos un "Grid Search" de nuestro dataset para obtener los parámetros óptimos (número de neuronas, funciones de activación…) de cara al entrenamiento.

En este caso el algoritmo funcionará en multiobjetivo, realizando la predicción de todos los targets en una sola vez.

2. 3. SCORES

Las puntuaciones o scores de nuestros algoritmos, serán los marcadores estadísticos que nos indicarán la calidad de las predicciones. Debido a que se realizan 5 K-Folds en el entrenamiento, el resultado final que se presentará será la media de los 5 scores. En este estudio se han utilizado dos scores diferentes:

● *Coeficiente de determinación* o R^2: el coeficiente de determinación será el principal score utilizado en el estudio, que se define como:

$$R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}} \quad (1)$$

Donde SS_{tot} es la suma total de cuadrados (proporcional a la varianza de los datos) y SS_{res} la suma de cuadrados de los residuos. En el caso perfecto en el que el modelo predice exactamente los valores, tendríamos un caso en el que $SS_{res} = 0$ por lo que $R^2 = 1$.

● *Error cuadrático medio* o *MSE:* el error cuadrático medio será el segundo score utilizado, que se define como:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y}_i - Y_i\right)^2 \quad (2)$$

Donde $\hat{Y}$ es un vector con las predicciones y Y un vector con los valores reales. Por lo tanto, un modelo perfecto que realice una predicción exacta presentará un MSE = 0.

2. 4. SELECCIÓN DE ATRIBUTOS Y TARGETS

Tal y como habíamos introducido en el apartado anterior, en el entrenamiento haremos uso de las simulaciones de solo materia oscura Multidark (cuyas propiedades se obtuvieron usando el buscador de halos ROCKSTAR), que es una simulación que contiene $5.6 \cdot 10^{11}$ partículas en un cubo de $1Gpc^3$.

Partiendo de la base de datos online (www.cosmosim.org), se han elegido 32 variables que pasarán a ser los atributos, es decir, las variables predictivas en nuestro estudio. Una descripción detallada de estos atributos se encuentra adjuntada al final de este informe en el ***Anexo 1***.

En cuanto a los targets o variables a predecir, hemos elegido cinco variables bariónicas importantes que definiremos a continuación:

● M_{gas} , la masa total del gas, que corresponde a la masa del ICM encerrada en el radio del cúmulo; calculado contando las partículas de gas que se encontraban dentro del radio del cúmulo.

● M_{star} , la masa total de estrellas, que se ha formado en cada galaxia del cúmulo. Al igual que M_{gas} se calculó contando las partículas de estrellas dentro del radio del cúmulo.

● T_{gas} , la temperatura del gas promediada en masa, es decir, la temperatura de las partículas de gas promediado a las masas de estas partículas. Se define como:

$$T_{gas} = \frac{\sum m_i T_i}{\sum m_i} \quad (3)$$

Donde m_i es la masa de la partícula de gas í-esima cuya temperatura es T_i.

● Y_{sz}, el Y integrado, que proporciona una estimación de la componente térmica del efecto Sunyaev Zel'dovich (SZ) en el cluster (Sunyaev & Zel'dovich, 1970 [19]). Se define como:

$$Y_{sz} \propto \int n_e T_e dA \tag{4}$$

Donde n_e es la densidad electrónica y T_e la temperatura electrónica. El efecto SZ surge de la interacción Compton inversa de los fotones del fondo cósmico de microondas con los electrones libres de los cúmulos, y puede considerarse como una medición de la presión electrónica del cúmulo.

● Y_{sx}, que podríamos considerar como la contraparte de rayos X de Y_{sz} (Biffi et al. 2014 [20]), se define como:

$$Y_{sx} = M_{gas} \cdot T_{gas} \tag{5}$$

Es preciso destacar que todas están cantidades han sido calculadas a R_{500}, que corresponde al radio al cual la densidad del cúmulo corresponde a 500 veces la densidad crítica del universo (en la época considerada).

2. 5. FEATURE IMPORTANCE

Para entender nuestro set de datos con el que entrenaremos el modelo, utilizaremos tres maneras distintas para obtener el feature importance, es decir, la importancia que tiene cada atributo a la hora de realizar las predicciones.

Para comprender las técnicas utilizadas en la obtención del feature importance, es importante antes entender como se divide el set de datos en el entrenamiento. Se presenta un diagrama en la *figura 2* donde puede verse esta división con claridad.

Los datos de entrenamiento están formados por los atributos (variables predictivas) y los targets (variables a predecir). Además, dentro de cada K-Fold, se dividen de manera aleatoria los datos de cada atributo y del target (recordemos que para esta parte trabajamos en monoobjetivo) entre la parte "train" y la parte "test". Los datos de los atributos los nombraremos con una X, de manera que los datos se dividirán entre X_{test} y X_{train} ; y los del target con una Y, dividiéndose entre Y_{train} y Y_{test}.

En el entrenamiento, el algoritmo (RF o NN), haciendo uso de X_{train} verá cuales son los parámetros óptimos para predecir los targets Y_{train}, y finalmente comprobará lo bueno que ha sido el entrenamiento tratando de predecir los valores de Y_{test} partiendo de X_{test}.

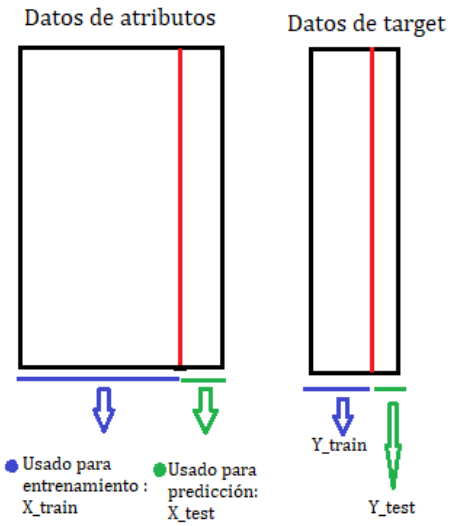

***Figura 2**: Distribución de los datos de entrenamiento.*

Conociendo la estructura que presentan los sets de datos de entrenamiento, podemos introducir los distintos métodos por los que se obtendrá el feature importance:

● ***Método 1** o método de "Permutation importance":* Se realiza el entrenamiento con X_{train} y Y_{train} , y se modifican de manera aleatoria las posiciones de los datos en X_{test} para cada atributo (manteniendo el resto igual), comprobando en cada caso como afecta ese cambio a la hora de predecir Y_{test}. Estas permutaciones en los datos se realizan diez veces (comprobando que tal se predicen los valores Y_{test} tras la permutación cada vez) y el error se almacena como la desviación estándar de los diez scores.

Si un atributo es muy relevante para realizar la predicción correctamente, al permutar aleatoriamente las posiciones de ese atributo en X_{test} es de esperar que las predicciones sean malas, mientras que si permutamos los valores de un atributo insignificante para predecir ese target, debido a que el resto de atributos no se han modificado la predicción debería ser buena.

Este método se implementa haciendo uso de la función "PermutationImportance" obtenido de la librería ELI5.

● ***Método 2:*** Debido a que para el análisis del feature importance se hará uso del algoritmo de RF, podemos aprovechar que al aplicar el algoritmo obtenido de la librería scikit-learn en el entrenamiento, aparte del algoritmo entrenado nos devuelve la variable "feature_importances_" junto con su error "feature_importances_std_". Estas variables son vectores con la información de la fracción relativa de veces que cada atributo ha sido utilizado en un árbol para realizar las predicciones.

Si un atributo es muy importante para la predicción, es de esperar que aparezca con gran frecuencia en los árboles de decisión que conforman esa predicción.

● ***Método 3*** *o método de re-entrenamiento:* En este método, a diferencia del método 1, se mantienen X_{test} y Y_{test} sin modificar y serán los datos de entrenamiento los que permutaremos. En concreto, se permutarán aleatoriamente los valores de un atributo de X_{train} y se entrenará el algoritmo (sin modificar Y_{train}). Finalmente se comprobará que tal predice X_{test} los Y_{test} con ese algoritmo.

La idea es que si se entrena un algoritmo en el que se ha modificado un atributo importante, el algoritmo debería perder poder predictivo y el score en la parte de test debería ser malo. Si se permutan los valores de un atributo que no es relevante en cambio, el algoritmo seguirá prediciendo la parte de test con un score alto.

En este método, definimos el error como la desviación estándar de los scores en los 5 K-Folds.

Cabe destacar que este método es extremadamente costoso computacionalmente, ya que deben realizarse 5 entrenamientos (uno para cada K-fold) para las 32 variables, obteniendo un total de 160 entrenamientos.

3. RESULTADOS Y DISCUSIÓN

3. 1. FEATURE IMPORTANCE

En primer lugar, se ha tratado de obtener el feature importance de nuestro set de datos, que recordemos, se define como la importancia que tiene cada atributo a la hora de realizar las predicciones.

Obteniendo el feature importance siguiendo los tres métodos presentados en el apartado anterior para M_{gas} llegamos a la figura 3. El score elegido para representar el error relativo a sido el R^2 y se ha obtenido comparándolo con un entrenamiento en el que no se han modificado ni permutado ningún valor (set de datos original), es decir:

$$\epsilon_{rel} = \frac{\left(R^2_{original} - R^2_{método1|3}\right) \cdot 100\%}{R^2_{original}} \tag{6}$$

De manera que los errores relativos se presentarán en tanto por ciento.

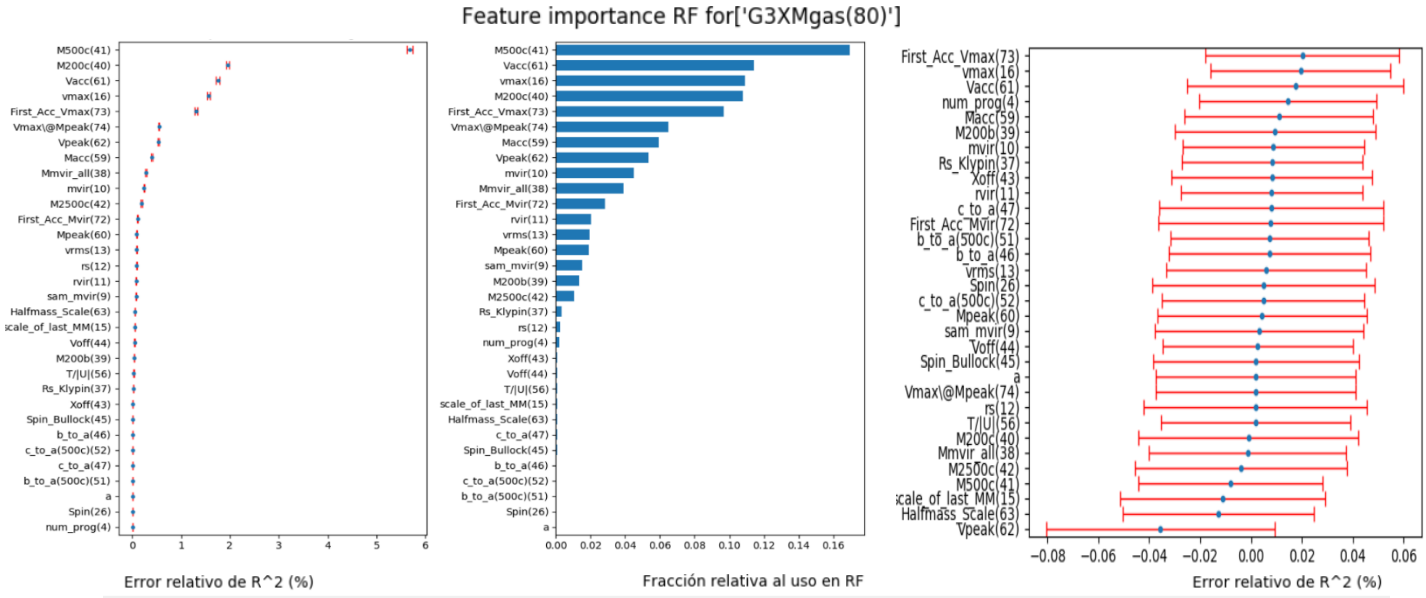

***Figura 3**: Feature importance de M_{gas} obtenido para todo el set de datos para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

Antes de realizar el análisis para el resto de targets vemos que hay varias cosas de la figura 3 que nos llaman la atención. Para empezar, vemos que los resultados obtenidos mediante el método 3 no son concluyentes ya que dice que todos los atributos tienen una importancia muy similar, con un error enorme, y variando del R^2 original prácticamente nada. Los otros dos análisis si que parecen proporcionarnos resultados mas coherentes, aunque vemos como hay varios atributos parecidos que tienen la misma importancia: M500-M200, Vacc-Macc, Mpeak-Vpeak … (recordar que la descripción de los atributos se encuentra en el *Anexo 1*).

Para entender estos resultados tenemos que tener en cuenta la naturaleza de nuestro set de datos, que está compuesto por variables físicas (de cúmulos de galaxias) que estarán muy correlacionadas. Es decir, existe una ecuación física que relacionará por ejemplo Mpeak con Vpeak, de manera que la información que contienen esos dos atributos puede ser la misma. Podríamos eliminar (o permutar sus valores como en el método 3) Mpeak de nuestro set de datos y el entrenamiento y predicciones seguirían siendo igual de buenas, ya que Vpeak contiene la misma información. Esto puede explicar porqué los resultados obtenidos por el método 3 son tan malos, o porque muchas de las variables que podrían estar correlacionadas presentan la misma importancia en los otros métodos.

Partiendo de este estudio preliminar, realizamos un estudio del set de datos, en el que podamos ver cuál es la correlación entre los datos. En primer lugar, realizamos la matriz de correlación de los datos de entrenamiento acompañado de un dendrograma para poder visualizar las relaciones entre los atributos de manera más clara. Los resultados de este análisis de se presentan en la figura 4:

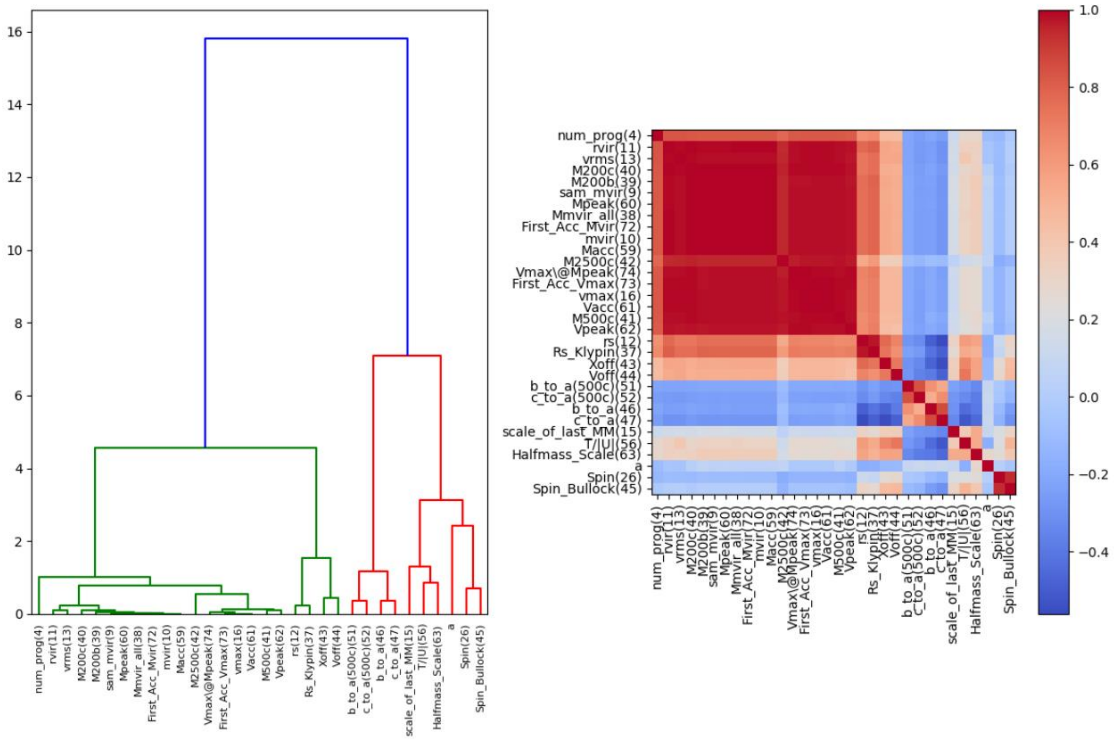

***Figura 4**: **Izq** :Dendrograma del set de entrenamiento , **Der**: matriz de correlación de Pearson del dataset de entrenamiento.*

En la matriz de correlación donde se presenta la correlación de Pearson entre todos los atributos vemos como tal y como esperábamos muchos atributos están altamente correlacionados. Como se ha comentado, los atributos que están correlacionados aportan la misma información al algoritmo y a la posterior predicción, de manera que se trató de simplificar el problema, reduciendo nuestro set de entrenamiento eliminando algunos atributos. Para ello, podemos o bien cortar horizontalmente el dendrograma en un punto, o tomar un atributo de cada cluster en la matriz de correlación.

En nuestro caso, de los 32 atributos pasaremos a tener solamente 9: “M500”, “Vpeak”, “num_prog”, “Spin”, “T/U”, “scale_of_last_MM”, “a”, “b_to_a” y “rs” (descripción de atributos en Anexo 1). Esta reducción de dimensionalidad también será muy eficiente desde el punto de vista computacional, ya que entrenar un algoritmo con 9 atributos será más del doble de rápido que hacerlo con los 32 iniciales.

Para comprobar que los datos ya no están tan correlacionados podemos graficar la matriz de correlación del nuevo set de datos con 9 atributos, *Anexo 2* figura 1. En esta figura comprobamos como efectivamente los datos ya no están apenas correlacionados, así que trataremos con este nuevo set de datos reducido de ahora en adelante.

Una vez tenemos definido nuestro set de entrenamiento con la dimensionalidad reducida, podemos pasar a calcular el featureimportance de los targets nuevamente. Los resultados para M_{gas} se muestran

en la figura 5. Los resultados para los otros cuatro targets se encuentran en el *Anexo 2* figuras 2 a 5.

Para M_{gas}:

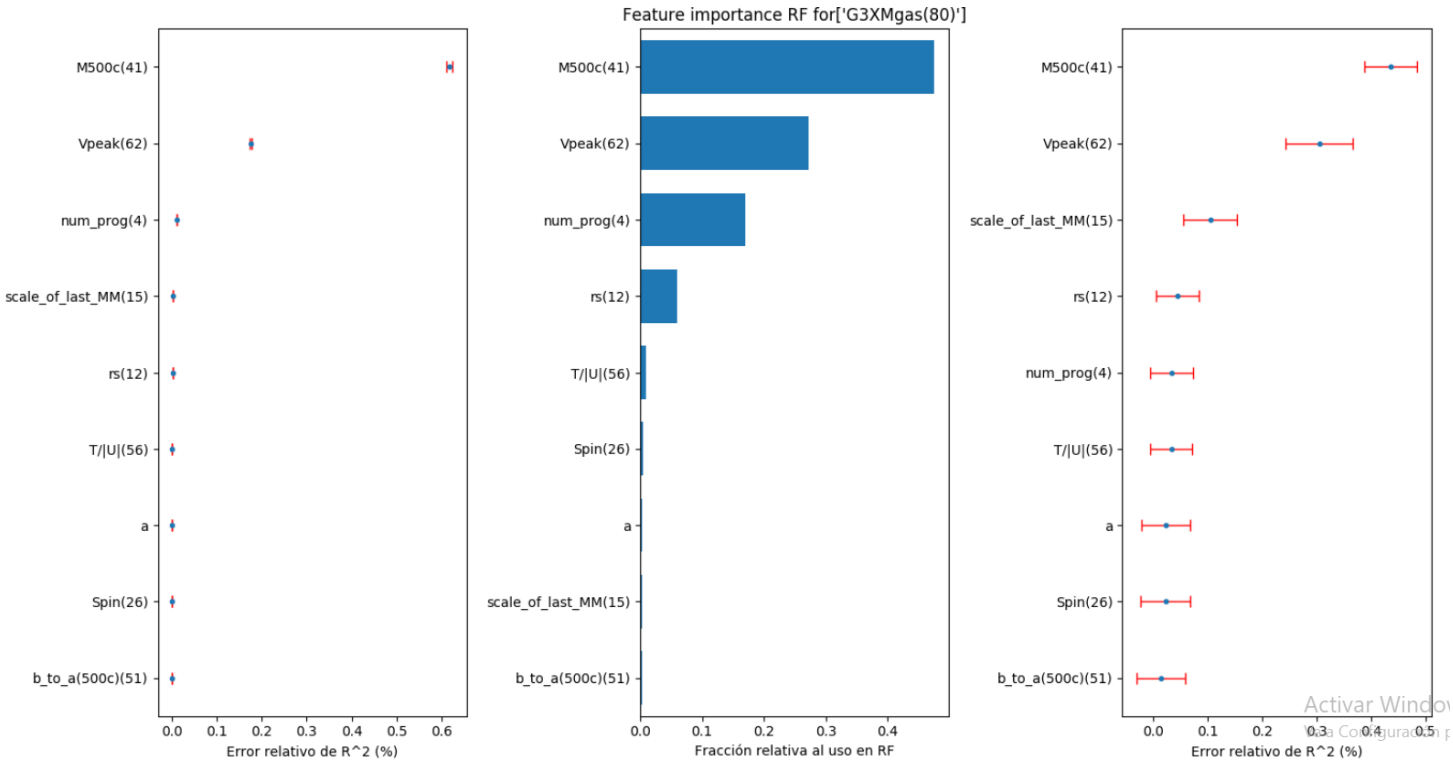

***Figura 5**: Feature importance de M_{gas} obtenido para el set de datos reducido para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

Esta vez, a diferencia del caso de la figura 3, debido a la simplificación realizada en el modelo vemos como los resultados de los tres métodos son razonables y coinciden en gran medida entre si.

El resultado que obtenemos para M_{gas} es muy parecido al obtenido para el resto de targets (*Anexo 2* figuras 2 a 5). Vemos como el atributo M500 es el más relevante a la hora de realizar las predicciones, seguidos de Vpeak que sería el segundo atributo más importante en nuestro modelo. Como excepción mencionar el caso del método 3 para T_{gas}, donde se ha obtenido que Vpeak sería el atributo más importante seguido de M500. Como tercer y cuarto atributos más importantes, en la mayoría de los casos encontramos a num_prog y scale_of_last_MM.

El error de los valores en el método 1 es menor que los del método 3, debido a que en el primero se realizan 10 permutaciones cada 5 K-Folds, mientras que en el método 3 solo una por K-Fold tal y como hemos comentado en el apartado anterior.

En definitiva, vemos que nuestro modelo estará altamente determinado por los valores de M500 y Vpeak, siendo M500 el atributo que jugará el papel más importante a la hora de realizar las predicciones.

Visto que aún y habiendo reducido mucho el set de datos y simplificando mucho el problema, hay algunos atributos que apenas tienen poder predictivo. Para ver esto de manera clara, podemos entrenar el modelo utilizando solamente un atributo, M500 (el más relevante), y después ir añadiendo atributos en función de su importancia, re-entrenando el modelo cada vez para ver cómo afectan estas adiciones al poder predictivo del modelo. Realizando este proceso en multiobjetivo con RF (es decir, predeciendo todos los targets de una vez) obtenemos el resultado que se presenta en la figura 6:

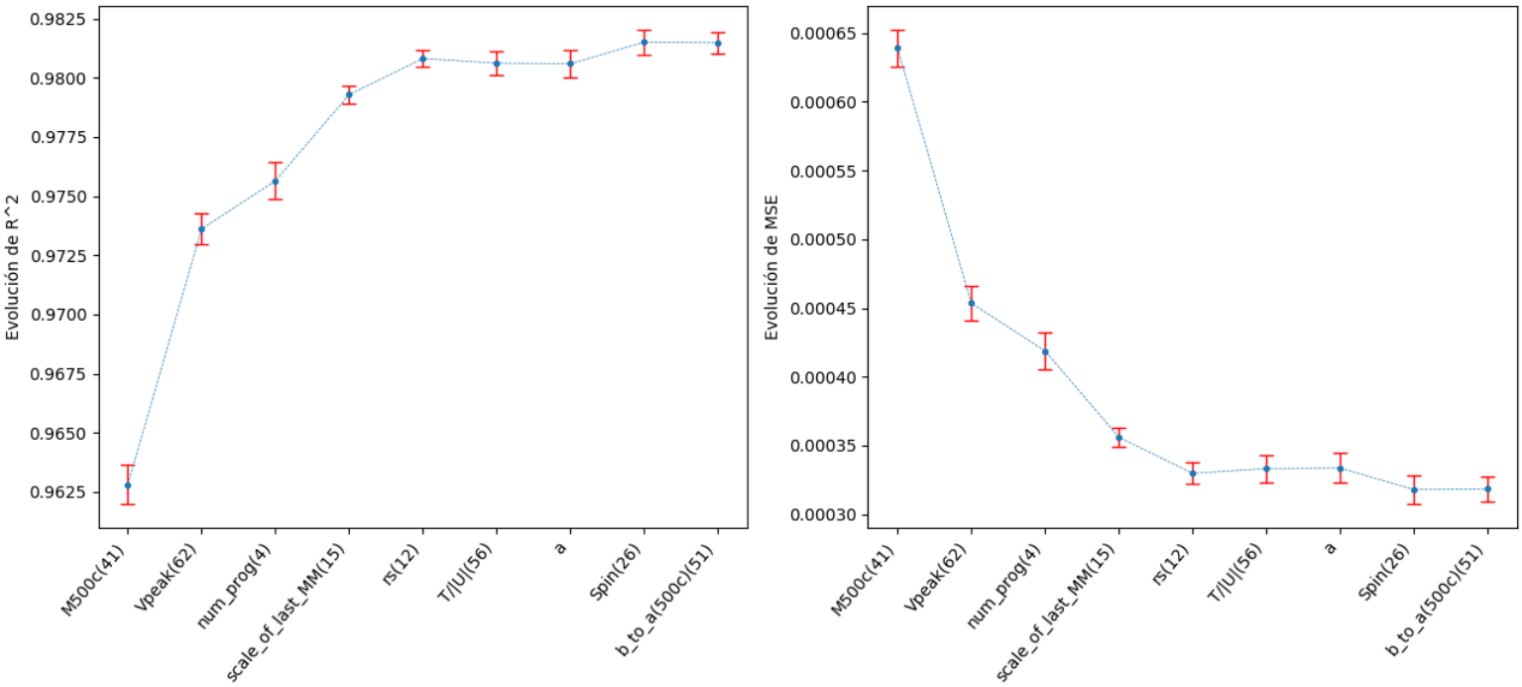

***Figura 6**: Evolución de los scores al ir añadiendo atributos en el entrenamiento. **Izq**: evolución de R^2 , **Der**: evolución de MSE.*

Para estudiar la evolución hemos usado los dos scores, el R^2 y el MSE, que naturalmente proporcionan resultados equivalentes. Es sorprendente el buen resultado que obtenemos utilizando solamente un atributo, M500 (el más relevante como hemos visto), para realizar las predicciones, obteniendo un R^2 de 0.9625 y un MSE de 0.00065. Después, tal y como esperábamos al ir añadiendo atributos y volviendo a entrenar, los resultados van mejorando describiéndose una curva asintótica, donde los últimos atributos que añadimos mejoran el score muy poco o en algunos casos no lo mejoran.

3. 2. ENTRENAMIENTO

Tras el primer estudio preliminar en el que hemos reducido considerablemente la dimensionalidad de nuestro dataset, pasamos a entrenar nuestros modelos en multiobjetivo mediante los algoritmos de RF y NN que posteriormente se utilizarán para realizar predicciones sobre datasets nuevos.

Para comprobar la calidad de nuestro entrenamiento en los dos algoritmos, utilizaremos la ecuación 6 del error relativo donde se compararán las predicciones con los Y_{test}. Es decir (*recordar figura 2*), entrenamos el modelo con X_{train} y Y_{train} para después realizar las predicciones con X_{test}, predicciones que compararemos con Y_{test} para calcular el error relativo. Es importante destacar que los datos con los que se ha trabajado están tomados en logaritmo de 10, de manera que el error relativo será sobre los valores en logaritmo de 10.

El error relativo, al igual que se hará con las predicciones, se representará frente al logaritmo del valor real de M500, que como hemos visto en el apartado anterior es la variable más importante en nuestro estudio. Por otro lado, debido a que en nuestro dataset hay datos tomados para distintos corrimientos al rojo ("z", redshift), siendo "a" (factor de escala, recordemos $z = \frac{1}{a} - 1$) uno de los atributos, en las representaciones cada punto tendrá un color asociado en función del valor de a (los valores de a para nuestro dataset van de 0.68 hasta 1).

Los resultados obtenidos se presentan en las figuras 7 y 8:

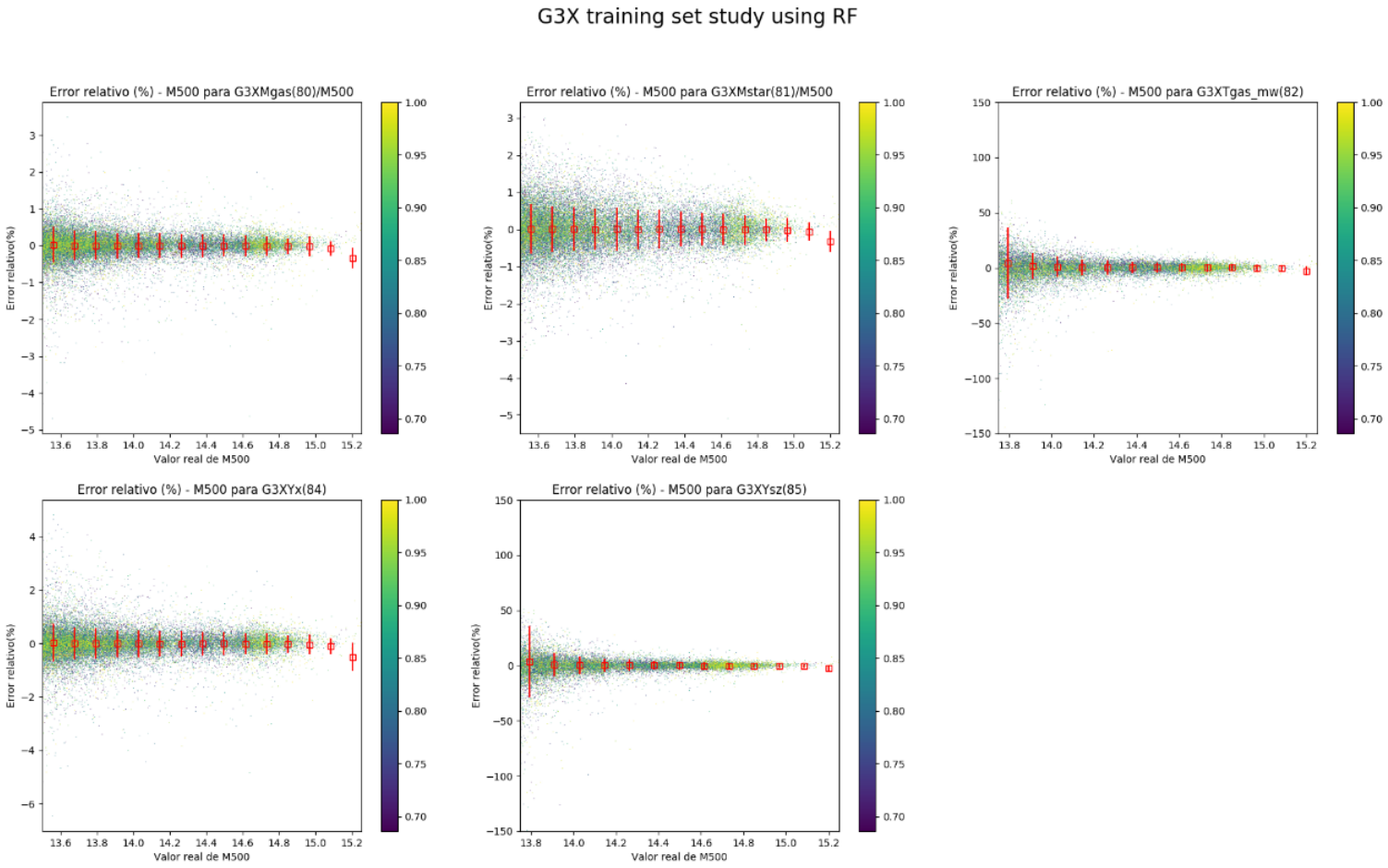

***Figura 7**: Error relativo de los datos de entrenamiento por RF frente al valor real de M500.*

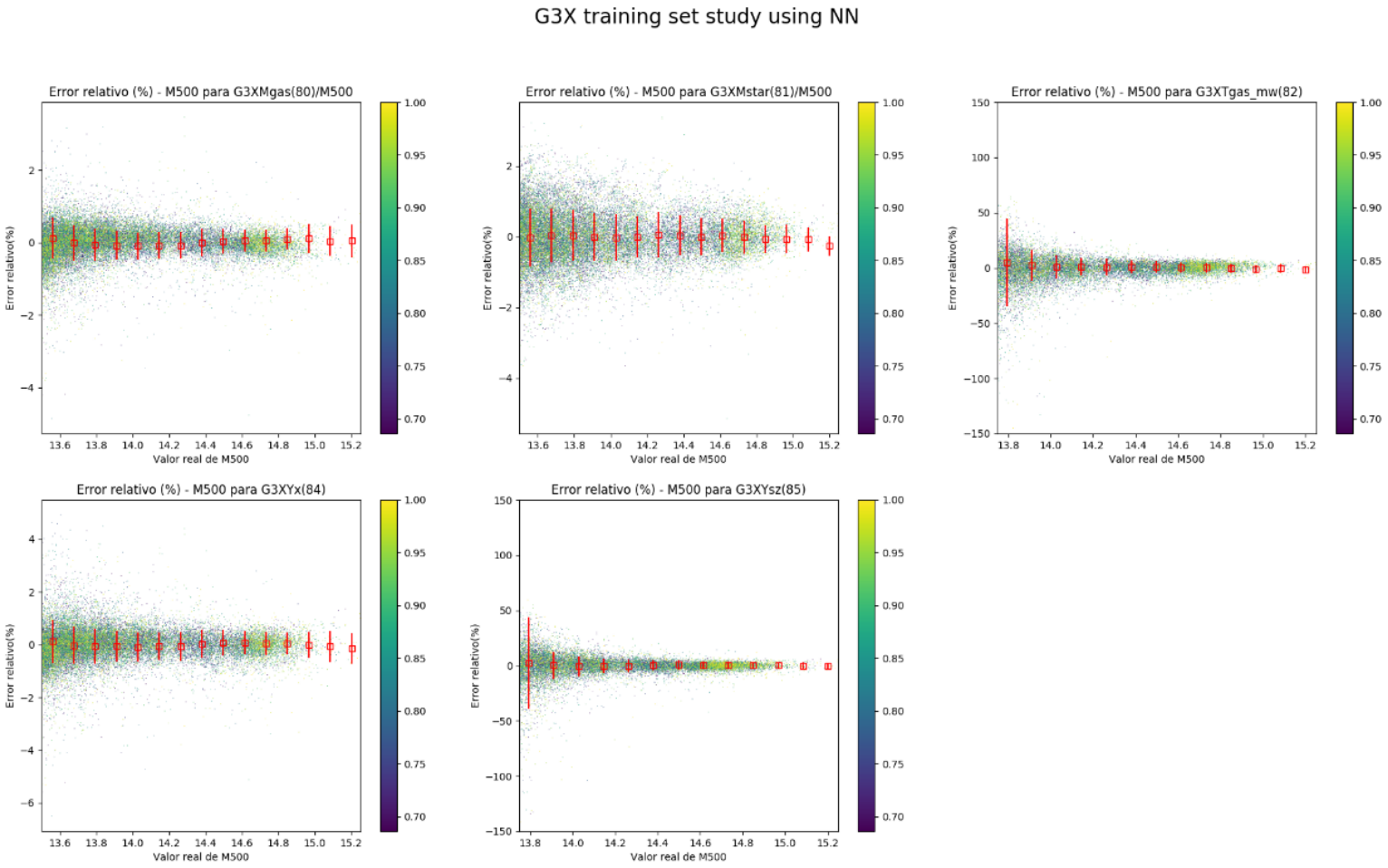

***Figura 8**: Error relativo de los datos de entrenamiento por NN frente al valor real de M500*

En las figuras se han añadido cajas o “bins” que promedian los puntos y presentan la dispersión con unas barras de error.

El primer resultado destacable que sacamos de estas gráficas es que aunque las predicciones para algunos targets presenten mayor dispersión, el error relativo está distribuido alrededor del 0, lo cual indica que las predicciones realizadas por el modelo entrenado son razonablemente buenas. No obstante, vemos que hay dos regiones en las que las predicciones no son tan buenas: para los valores de masa más bajos, la dispersión es muy grande y las predicciones por tanto peores, y para los valores de la masa más grandes, debido a que hay pocos datos, las predicciones también son menos precisas.

Por último, se representarán los targets del dataset de entrenamiento en función de log(M500)(ya que como hemos visto es el atributo más relevante) para conocer la tendencia y posteriormente compararlas con las tendencias obtenidas en las predicciones con datasets nuevos (ya que en las predicciones no existirán valores reales de los target para comparar). Representamos estas relaciones en las figuras 6,7 y 8 del *Anexo 2.*

3. 3. PREDICCIONES

Tal y como se ha adelantado, las predicciones se han realizado sobre las simulaciones de solo DM de UNIT. Mientras que el dataset que hemos utilizado hasta ahora si que había sido utilizado anteriormente para entrenas algoritmos de ML (Franciso Robledo, pendiente de publicación), el dataset UNIT sobre el cual realizaremos las predicciones no ha sido hasta ahora uso de un estudio de este tipo.

Una vez tenemos el nuevo dataset preparado solo queda aplicarle los algoritmos entrenados para obtener la predicción. Para representar estas predicciones se graficarán los valores de los targets predichos (target/M500 para M_{gas} y M_{star}) en función de M500.

Las simulaciones UNIT estaban realizadas para distintos valores del factor de escala dentro de nuestro rango de entrenamiento, en concreto para 18 valores distintos de a (desde a=0.6862 hasta a=1). Se han realizado predicciones para todos los casos aunque por razones obvias no se expondrán todos los resultados en este informe. Por ello se han elegido arbitrariamente algunos valores de "a" distintos. Las predicciones para a=0.6862 para RF y NN se presentan en las figuras siguientes. Para RF:

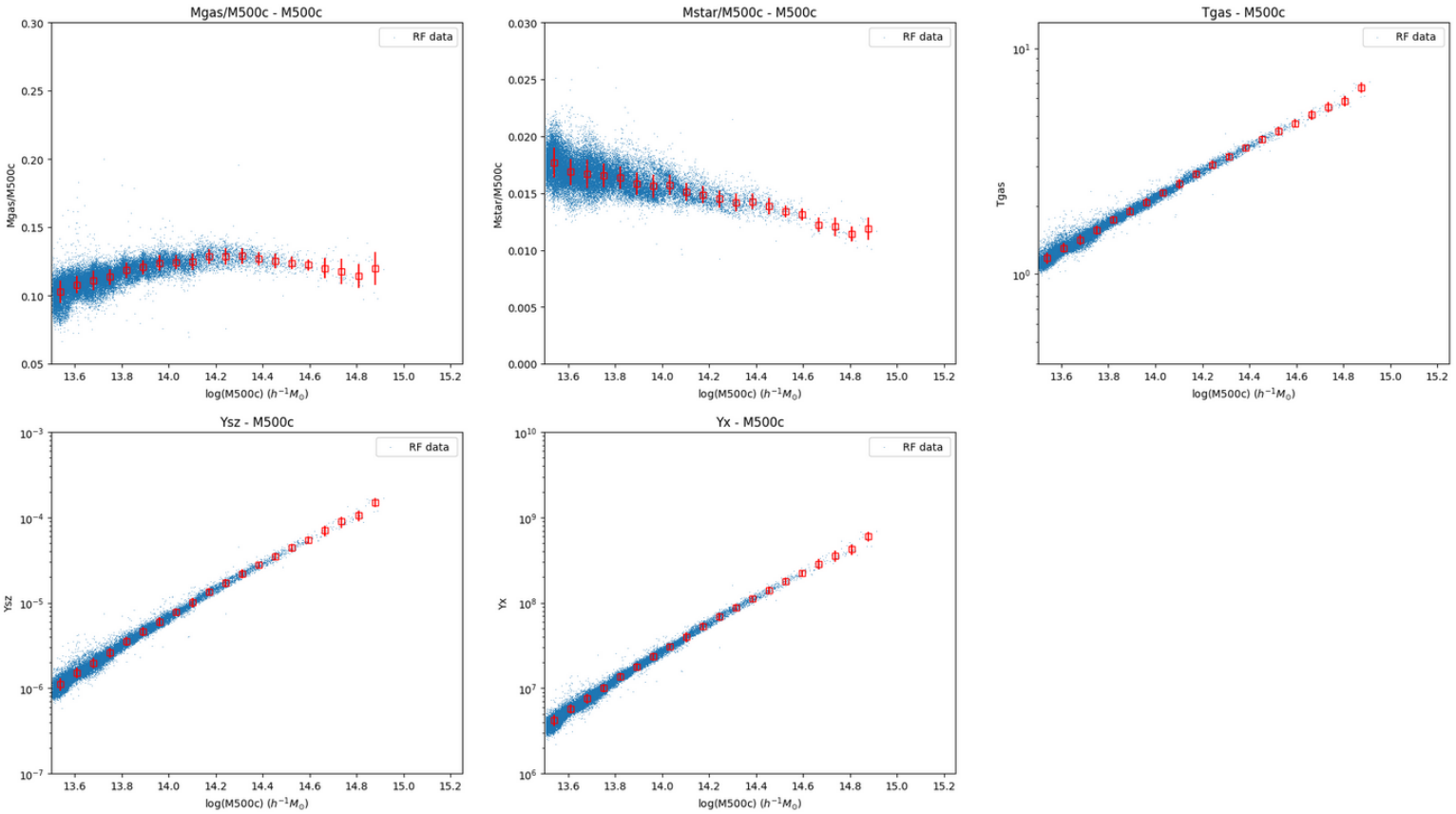

***Figura 9**: Predicciones realizadas con el algoritmo de RF sobre el dataset UNIT. Representación de los distintos targets en función de M500.*

Y representando la misma gráfica pero teniendo en cuenta la densidad de puntos obtenemos:

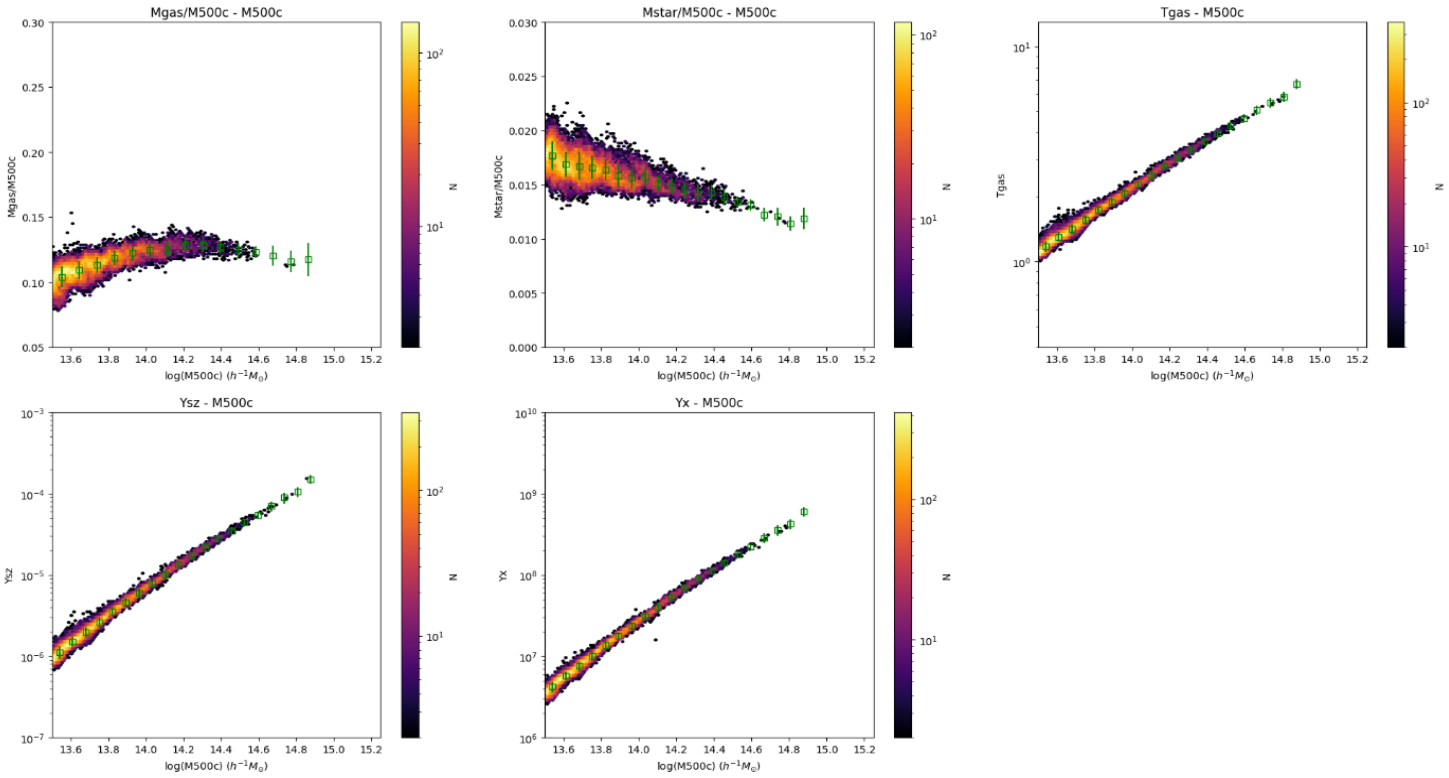

***Figura 10**: Predicciones realizadas con el algoritmo de RF sobre el dataset UNIT. Representación con densidad de datos de los disintos targets en función de M500.*

Las predicciones realizadas con NN:

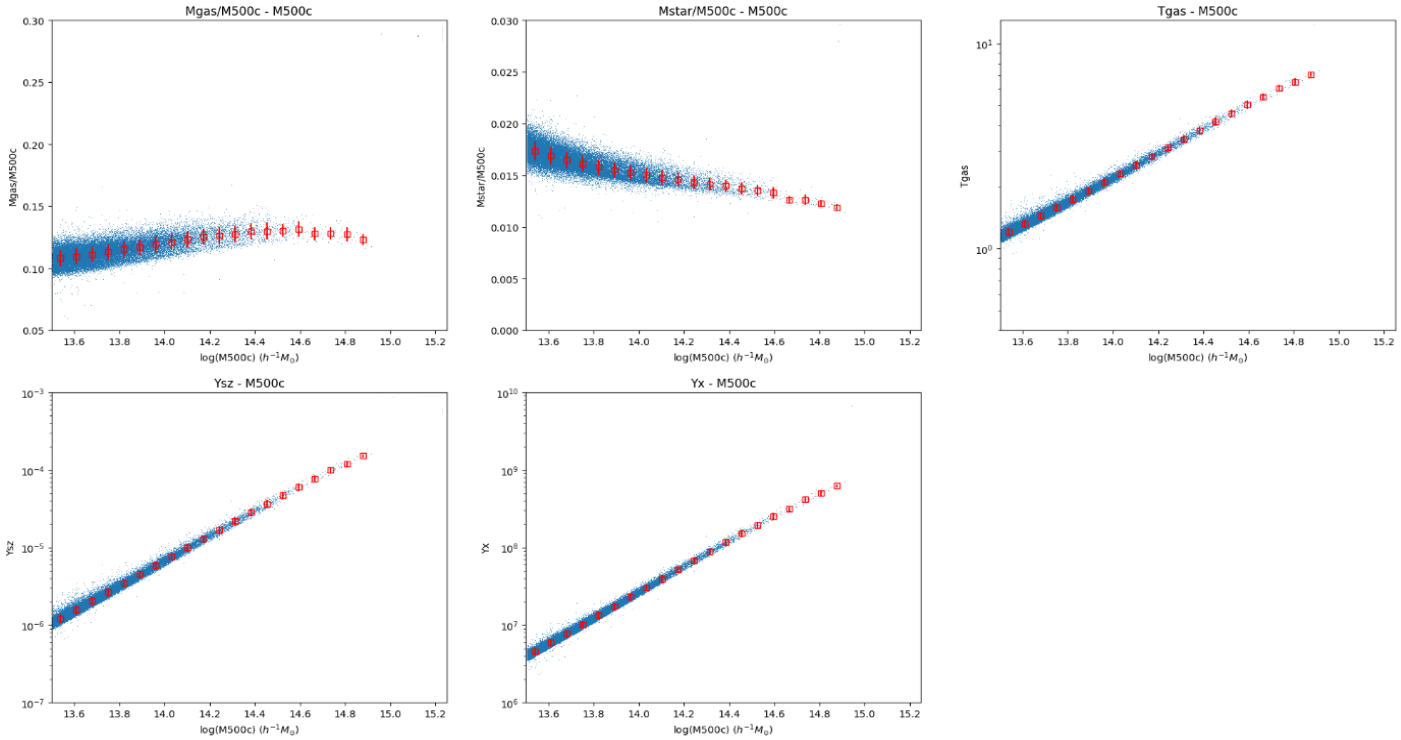

***Figura 11**: Predicciones realizadas con el algoritmo de NN sobre el dataset UNIT.*

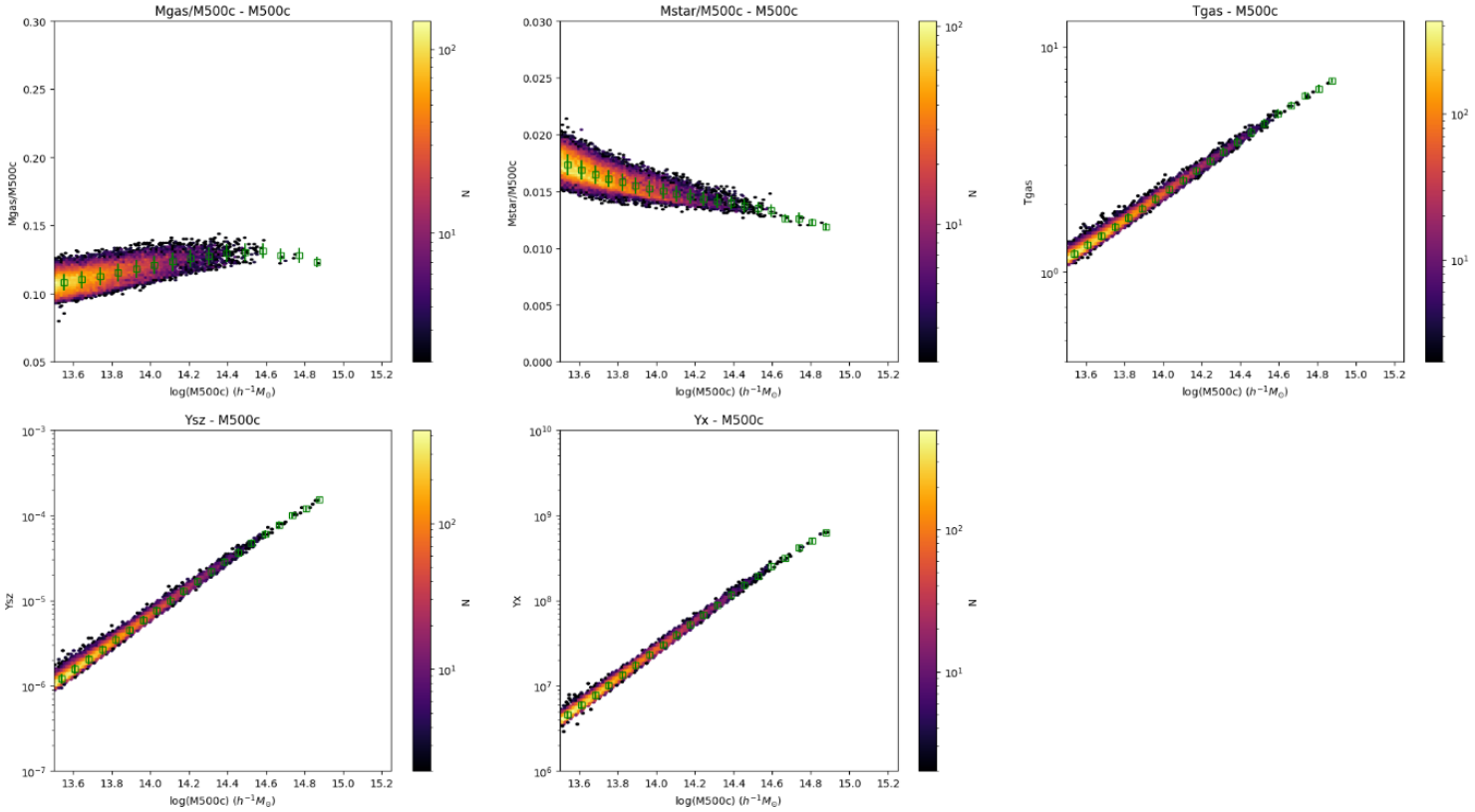

***Figura 12**: Predicciones realizadas con el algoritmo de NN sobre el dataset UNIT. Representación con densidad de datos.*

Las predicciones para otros valores de "a" con densidad de puntos presentan las mismas tendencias para todos los targets.

Vemos como las predicciones que realizan nuestros modelos de RF y NN sobre el nuevo dataset son razonablemente buenas, ya que las tendencias que describen los target en función de M500 son las mismas que presentan en los datasets de entrenamiento (Anexo 2, figuras 6,7 y 8). Para T_{gas}, Y_x y Y_z es donde más claramente se ve como se predice una relación claramente lineal entre los targets y M500. Además, vemos como las predicciones de NN presentan menor dispersión en los datos que las predicciones de RF, en los que (figuras 9 y 10) para masas más pequeñas las predicciones no parecen tan buenas.

Si en lugar de trabajar con un valor concreto de "a", juntamos todos los datasets y realizamos las predicciones sobre el dataset completo para ambos algoritmos, variando el color de cada punto de la gráfica en función del valor de "a", obtenemos los resultados que se presentan en las figuras 13 y 14.

En esta figura, vemos como el razonamiento anterior de que las predicciones son buenas es aplicable para todos los datos. Además, el algoritmo predice los targets teniendo en cuenta la variación de "a" ya que aunque en M_{gas} y M_{star} la distribución de datos parece ser independiente de "a", para T_{gas}, Y_z

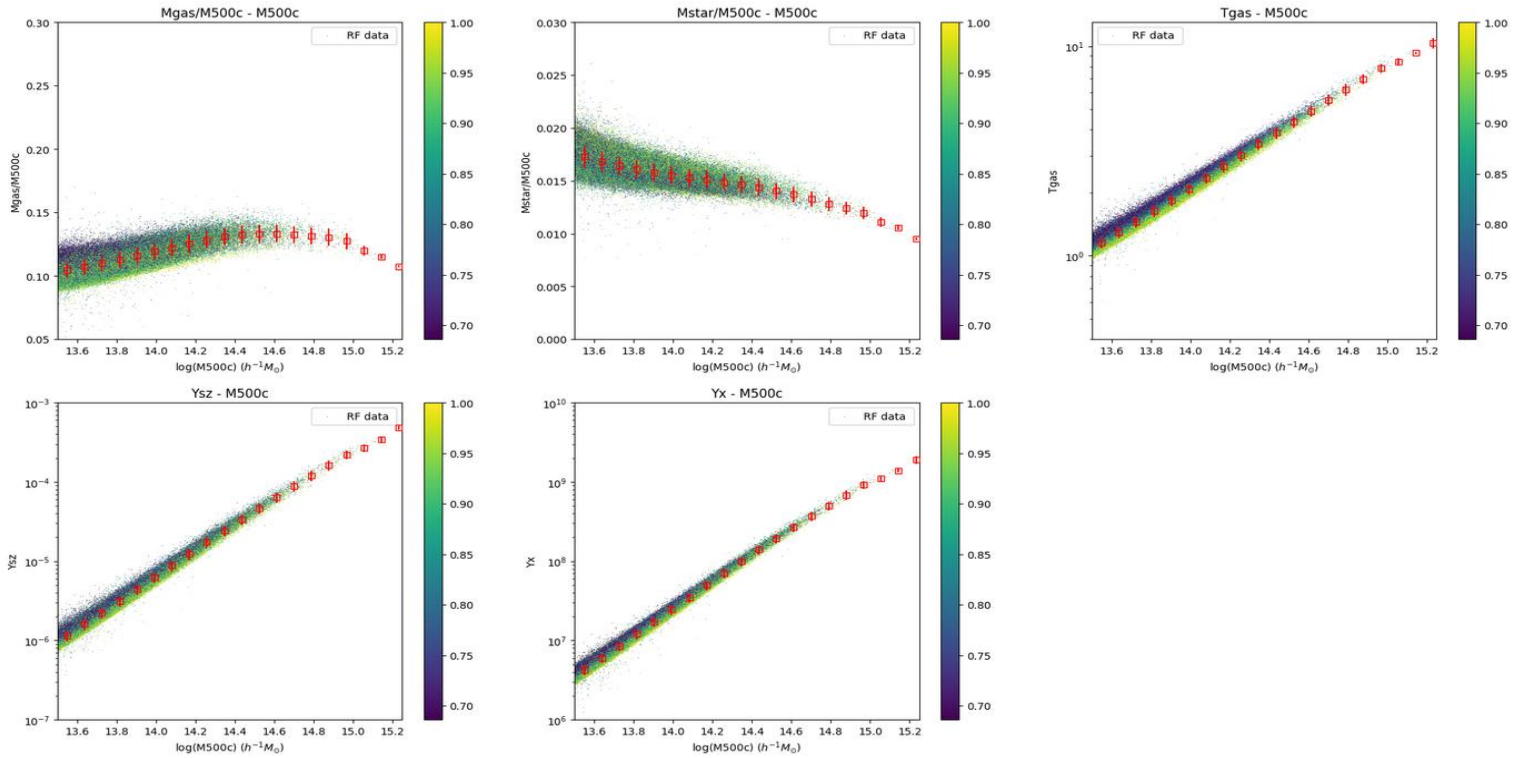

***Figura 13**: Predicciones realizadas con el algoritmo de RF sobre el dataset completo de UNIT.*

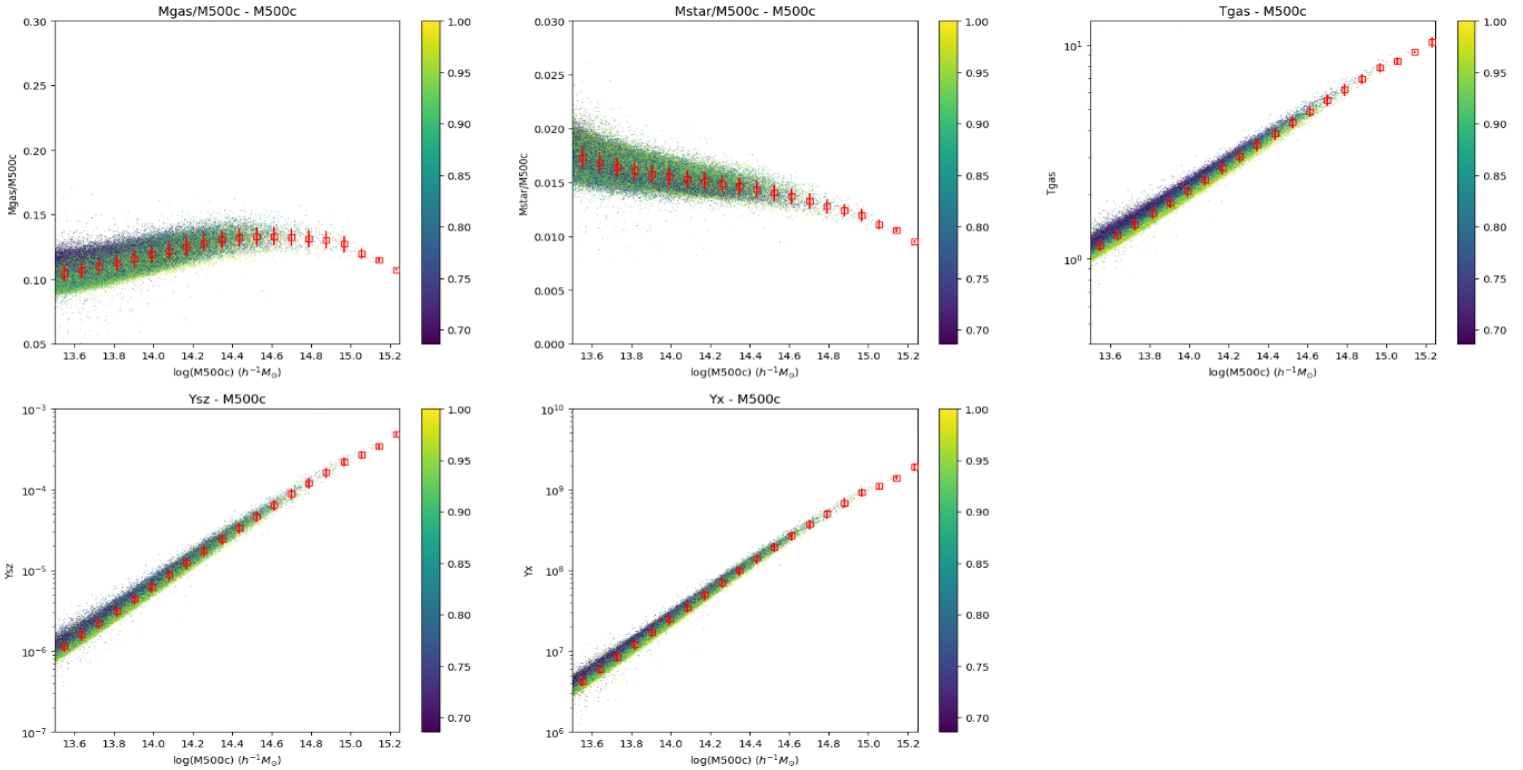

***Figura 14**: Predicciones realizadas con el algoritmo de NN sobre el dataset completo de UNIT.*

y Y_x se puede apreciar como claramente para valores más grandes del target, menor es el valor del factor de escala, a.

Sin embargo, encontramos que hay una región para masas grandes ($\log(M500) > 14.9\, h^{-1}M_o$) en el que las predicciones empiezan a fallar y empiezan a presentar un comportamiento anómalo. Para entender este hecho, tenemos que volver al análisis del entrenamiento de los algoritmos (figuras 7 y 8), donde podremos comprobar que debido a la baja densidad de datos de entrenamiento en esa zona los modelos no están bien entrenados para realizar predicciones para masas tan grandes.

3. 3. 1. EFECTOS DE LA RESOLUCIÓN

Hasta ahora, las simulaciones solo DM de las que hemos obtenido los datos para el entrenamiento y la predicción estaban realizadas por cubos que contenían 4096^3 partículas. En las simulaciones UNIT, también hay una simulación análoga a la que hemos utilizado en la predicción, pero con una resolución de 2048^3 partículas. Es decir, es la misma simulación solo DM donde los objetos tienen ocho veces menos resolución.

Hemos utilizado ese nuevo dataset de menor resolución para ver si el algoritmo es capaz de distinguir entre objetos con distintas resoluciones o si pierde poder predictivo. Para poder comparar las dos resoluciones se ha representado el equivalente a la figura 14 (utilizando NN) de cada resolución utilizando colores diferentes. El resultado se muestra en la figura 15:

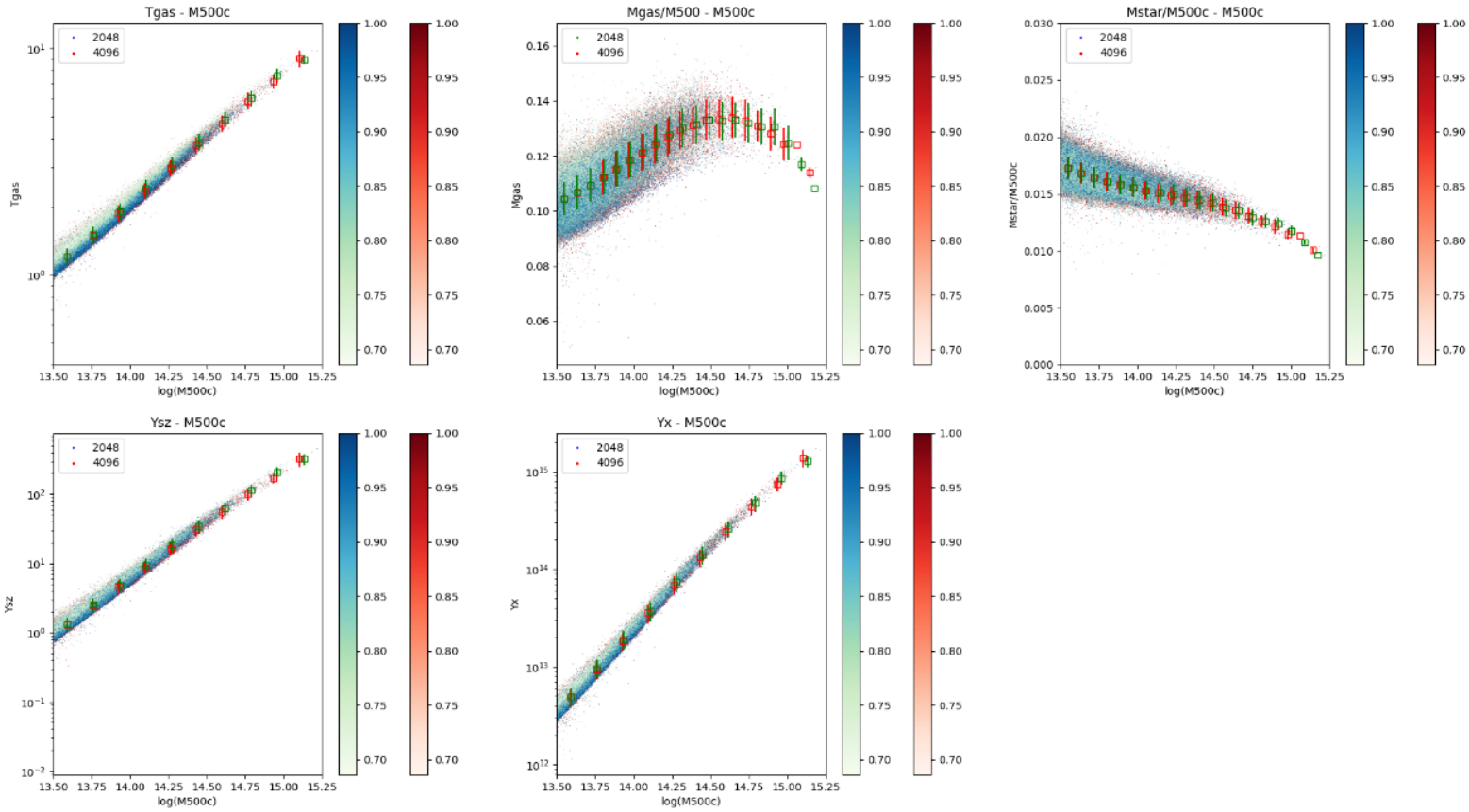

***Figura 14**: Predicciones realizadas con NN para los dataset completos de las dos resoluciones.*

En esta figura podemos ver claramente como las predicciones para el dataset obtenido de la simulación a menor resolución sigue siendo igual de buena. Se han graficado cajas con su error marcando el centro y la dispersión de los datos de las predicciones de cada resolución para poder ver como la tendencia es prácticamente la misma. Las predicciones son tan similares que para Y_x, Y_z y T_{gas} los puntos de datos se sobreponen entre ellos y es difícil diferenciar entre los dos colores.

De la misma manera vemos que para ambas resoluciones las predicciones para M_{gas} y M_{star} para masas grandes no es buena, por las mismas razones que se han comentado en el apartado anterior.

Este resultado es muy significativo, ya que nos indica que partiendo de una simulación más barata (en términos de tiempo y recursos computacionales) pueden predecirse los mismos resultados que de una simulación mucho más cara.

4. CONCLUSIONES

Al empezar a trabajar con nuestro set de datos completo, se ha visto que había varias variables altamente correlacionadas que aportaban a nuestros algoritmos la misma información, por lo que se ha reducido la dimensionalidad del dataset pasando de 32 atributos a 9. De esta manera se ha simplificado mucho nuestro problema, ahorrando también recursos computacionales, sin perder poder predictivo.

Analizando mediante tres métodos distintos la importancia de cada atributo en las predicciones se ha visto como claramente la masa 500 (M500), seguido de la velocidad máxima (Vpeak) son los atributos más influyentes de cara a las predicciones.

Después de entrenar nuestros algoritmos, hemos confirmado que son capaces de realizar predicciones buenas y se ha visto que las predicciones para la región con masas más pequeñas y más grandes son peores y presentan mayor error.

Finalmente, hemos aplicado los dos algoritmos entrenados a la nueva base de datos de UNIT, consiguiendo predecir de manera razonable los targets ($M_{gas}, M_{star}, T_{gas}, Y_Z, Y_x$) para ambos. En estas predicciones se ha visto como la dependencia con los targets con la variable M500 es la misma que la que se presenta en los datos de entrenamiento. También se han realizado las predicciones sobre el dataset obtenido de la misma simulación de UNIT pero con ocho veces menos resolución, viendo que las predicciones son muy similares a las obtenidas para el caso de mayor resolución. Podemos

concluir por tanto que aunque una simulación se pueda realizar con mayor resolución, aplicando técnicas de machine learning podemos obtener la misma información de una simulación con una resolución ocho veces menor.

5. REFERENCIAS

[1] Borgani & Kravtsov, 2011, Advanced Science Letters, Volume 4, Number 2, pp. 204-227.

[2] Ball N. M., Brunner R. J., Myers A. D., Tcheng D., 2006, ApJ, 650, 497.

[3] Fiorentin P. R., Bailer-Jones C., Lee Y., Beers T., Sivarani T., Wilhelm R., Prieto C. A., Norris J., 2007, A&A, 467, 1373.

[4] Banerji M., et al., 2010, MNRAS, 406, 342.

[5] Ball N. M., Brunner R. J., 2010, International Journal of Modern Physics D, 19, 1049.

[6] Gerdes D. W., Sypniewski A. J., McKay T. A., Hao J., Weis M. R., Wechsler R. H., Busha M. T., 2010, ApJ, 715, 823.

[7] Kind M. C., Brunner R. J., 2013, MNRAS, 432, 1483.

[8] Ivezi´c Z., Connolly A. J., VanderPlas J. T., Gray A., 2014, Statis- ˇ tics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data: A Practical Python Guide for the Analysis of Survey Data. Princeton University Press.

[9] Ness M., Hogg D. W., Rix H.-W., Ho A., Zasowski G., 2015, arXiv preprint arXiv:1501.07604.

[10] Kim Edward B. R. C.-K. M., 2015, MNRAS, 453, 507.

[11] Dieleman S., Willett K. W., Dambre J., 2015, MNRAS, 450, 1441.

[12] Xu X., Ho S., Trac H., Schneider J., Poczos B., Ntampaka M., 2013, ApJ, 772, 147.

[13] Kamdar, Harshil M., Turk, Matthew J., Brunner, Robert J., 2016a, MNRAS, 455, 642.

[14] Federico Sembolini, TFM Universidad Autónoma de Madrid, 2018.

[15] Sembolini F., Yepes G., De Petris M., Gottl¨ober S., Lamagna L., Comis B., 2013, MNRAS, 429, 323.

[16] Prada F., Klypin A. A., Cuesta A. J., Betancort-Rijo J. E., Primack J., 2012, MNRAS, 423, 3018

[17] Chia-Hsun Chuang, Gustavo Yepes, Francisco-Shu Kitaura, Marcos Pellejero-IbanezUNIT project: Universe N-body simulations for the Investigation of Theoretical models from galaxy surveys; arXiv:1811.02111.

[18] Pedregosa F., et al., 2011, The Journal of Machine Learning Research, 12, 2825.

[19] Sunyaev R. A., Zeldovich Y. B., 1970, Ap&SS, 7, 3.

[20] Biffi V., Sembolini F., De Petris M., Valdarnini R., Yepes G., Got- tl¨ober S., 2014, MNRAS, 439, 588.

ANEXO 1

Atributo	***Descripción***
Num_prog	Number of progenitors
Sam_mvir	Halo mass, smoothed across accretion history; always greater than sum of halo masses of contributing progenitors (Msun/h). On ly for use with select semi-analytical models.
mvir	Halo mass (Msun/h)
rvir	Halo radius (kpc/h comoving)
rs	Scale radius (kpc/h comoving).
vrms	Velocity dispersion (km/s physical)
Scale_of_last_MM	Scale factor of the last major merger (Mass ratio >0.3)
vmax	Maxmimum circular velocity (km/s physical).
Spin	Halo spin parameter
Rs_Kyplin	Scale radius determined using Vmax and Mvir (see Rockstar paper).
Mmvir_all	Mass enclosed within the specified overdensity, including unbound particles (Msun/h)
M200b-M500-M2500	Mass enclosed within specified overdensities (Msun/h)
Xoff	Offset of density peak from average particle position (kpc/h comoving
Voff	Offset of density peak from average particle velocity (km/s physical
Spin_Bullock	Bullock spin parameter (J/(sqrt(2)*GMVR))

Halfmass_scale	: Scale factor at which the MMP reaches 0.5*Mpeak.
First_Acc_Mvir, First_Acc_Vmax	: Mvir and Vmax at First_Acc_Scale.
Vmax\@Mpeak	: Halo Vmax at the scale at which Mpeak was reached
a	Scale factor, z=1/a -1
b_to_a , c_to_a	Ratio of second and third largest shape ellipsoid axes (B and C) to largest shape ellipsoid axis (A) (dimensionless)
T/|U|	ratio of kinetic to potential energies
Macc , Vacc	Mass and Vmax at accretion.
Mpeak, Vpeak	Peak mass and Vmax over mass accretion history

ANEXO 2

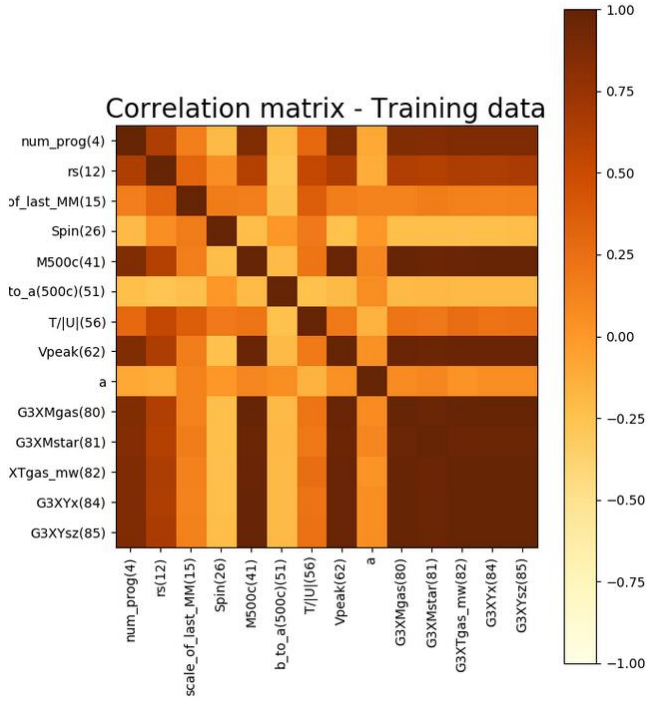

***Figura 1**: Matriz de correlación para el dataset con la dimensionalidad reducida.*

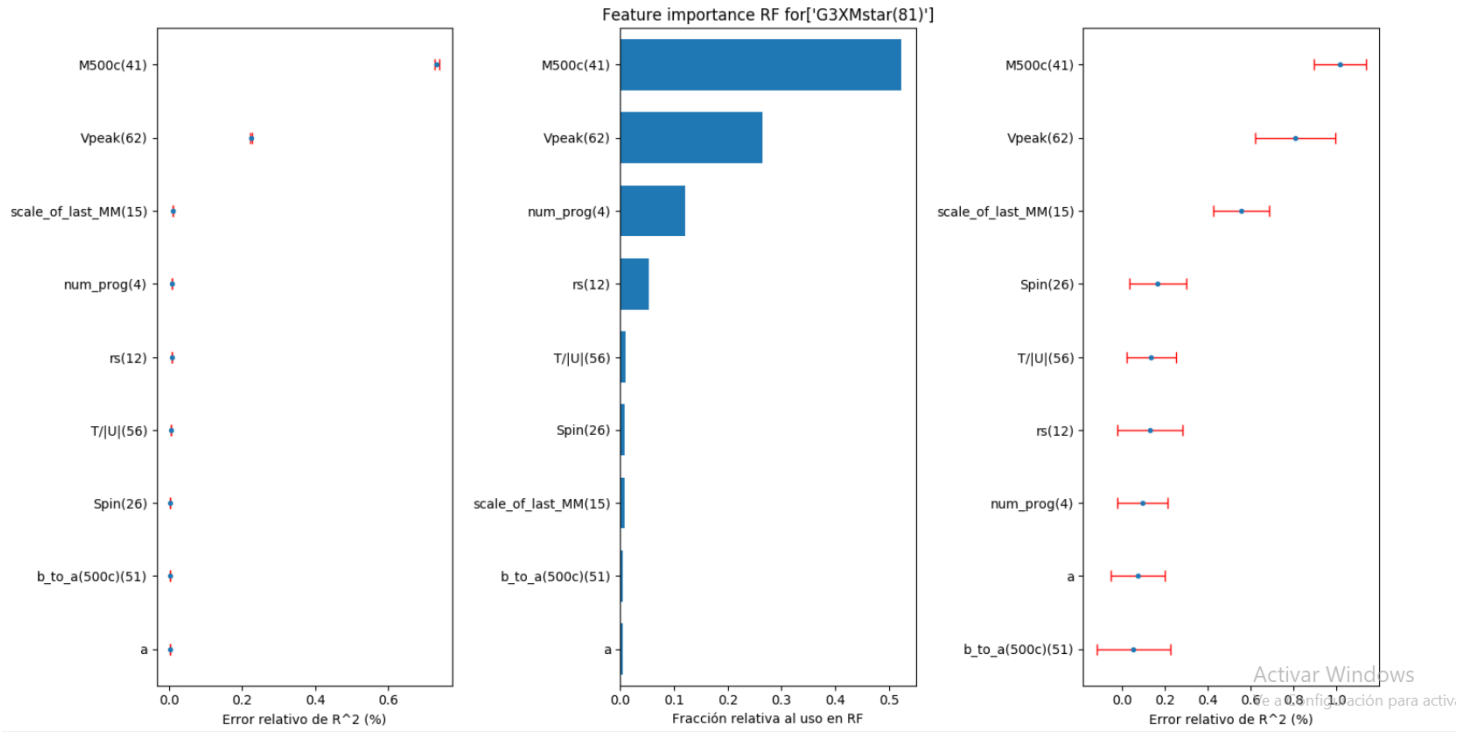

***Figura 2**: Feature importance de M_{star} obtenido para el set de datos reducido para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

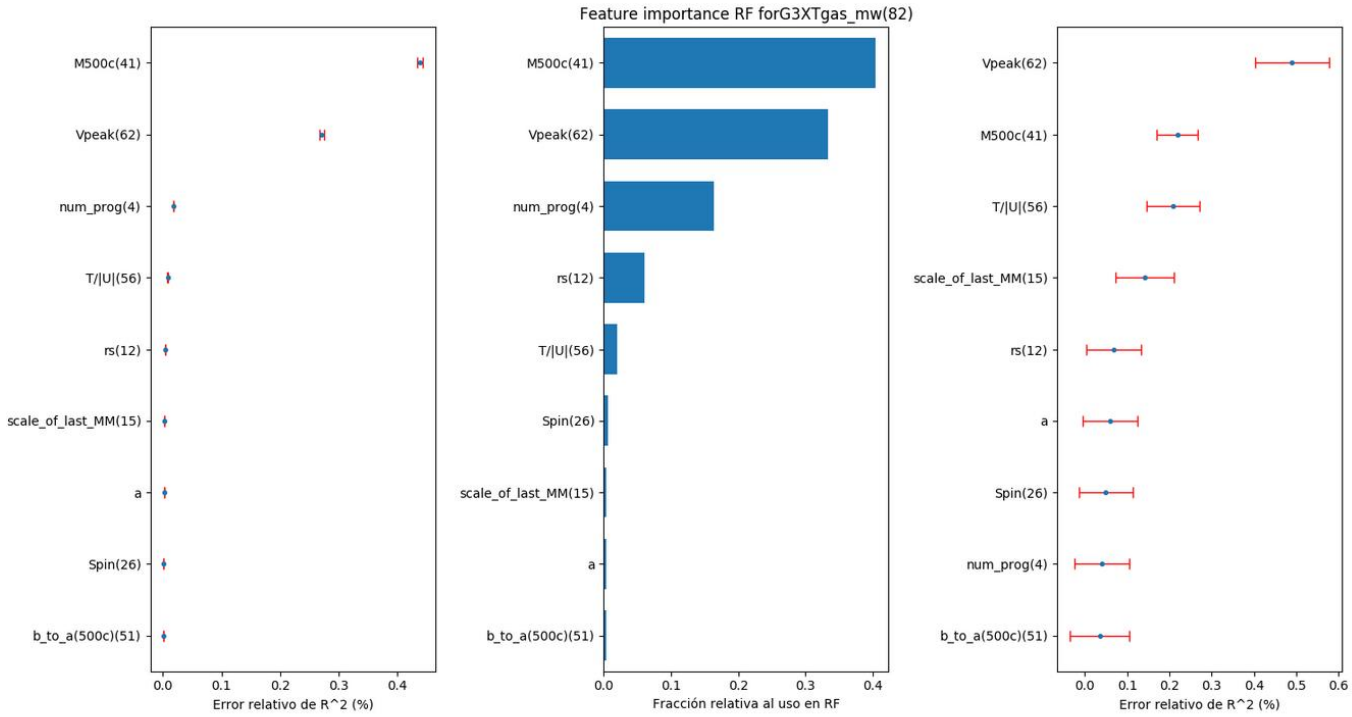

***Figura 3**: Feature importance de T_{gas} obtenido para el set de datos reducido para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

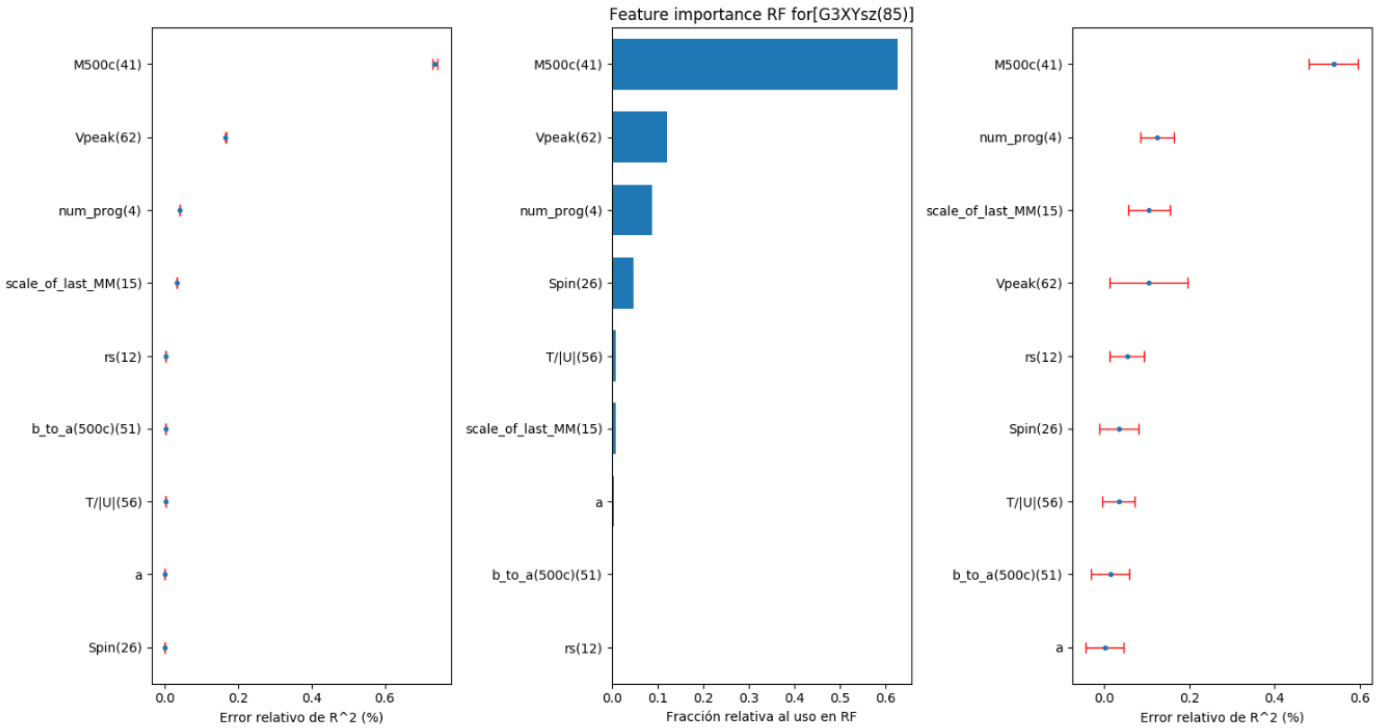

***Figura 4**: Feature importance de Y_{sz} obtenido para el set de datos reducido para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

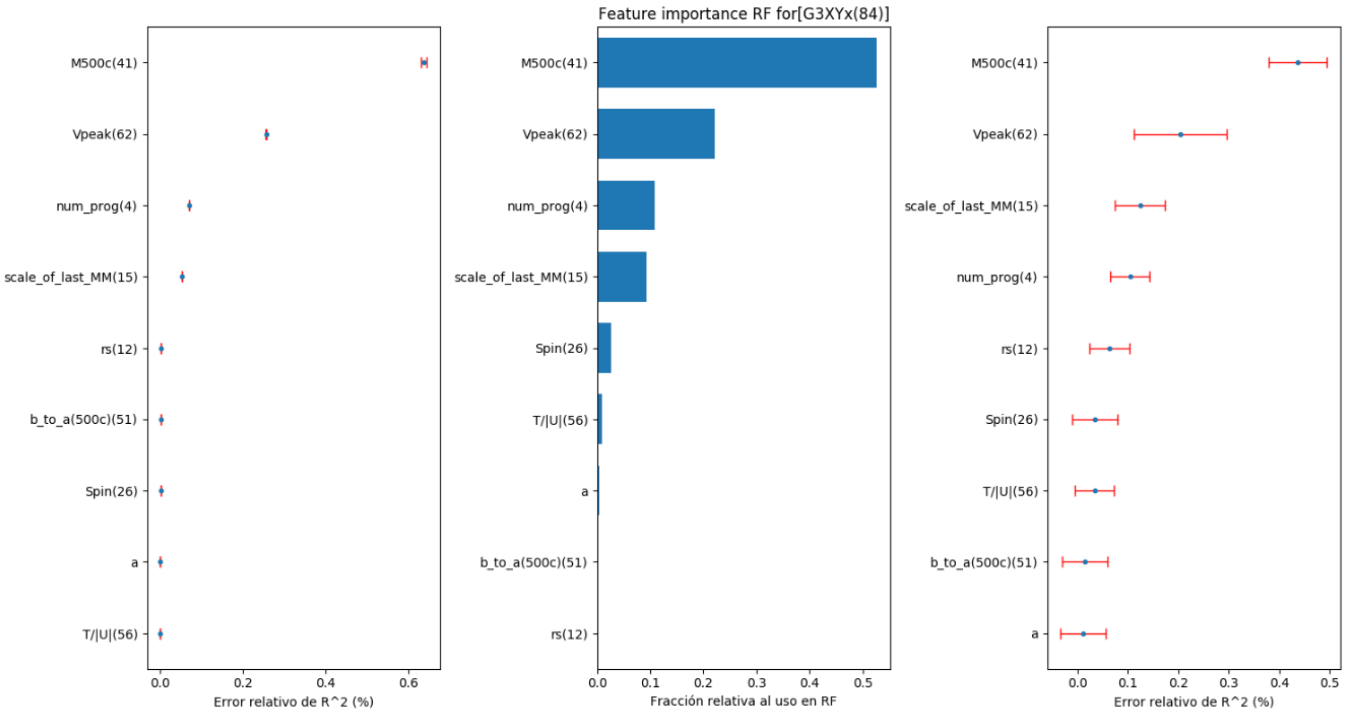

***Figura 5**: Feature importance de Y_{sx} obtenido para el set de datos reducido para **Izq**:método 1, **Centro**:método2 , **Der**: método 3.*

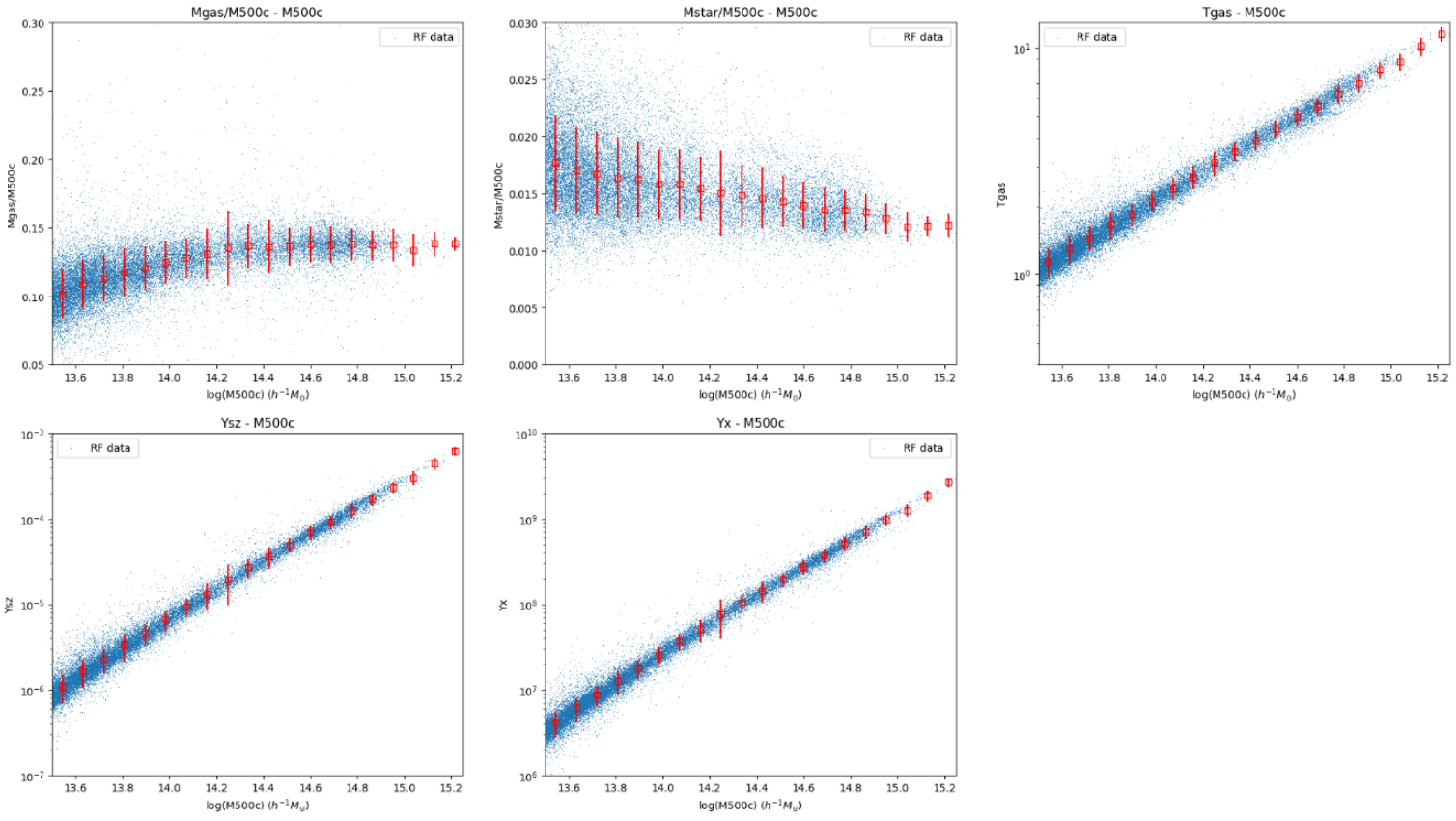

***Figura 6**: Representación de los targets en función de log(M500) para los datos de entrenamiento.*

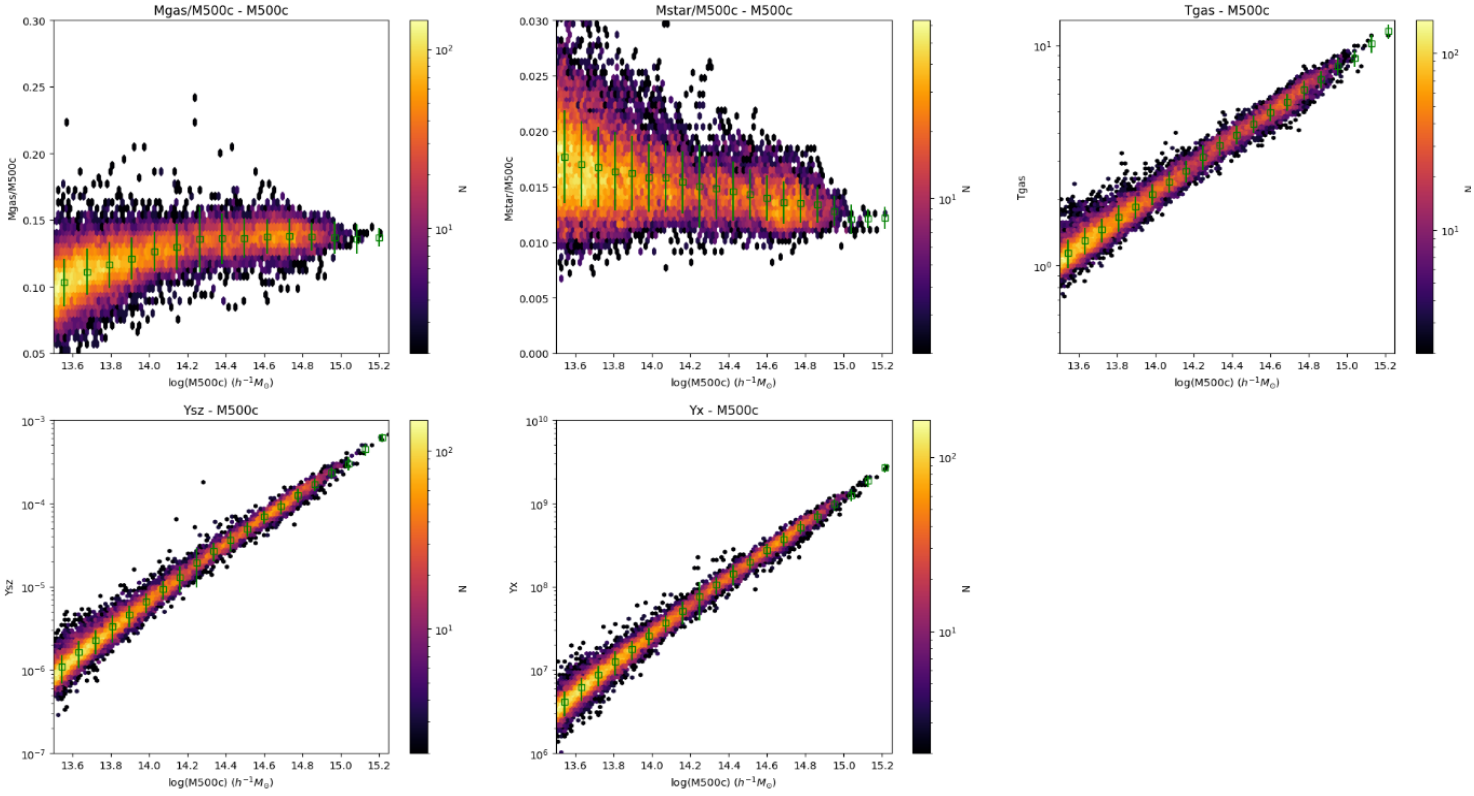

***Figura 7**: Representación de los targets con densidad de puntos en función de log(M500) para los datos de entrenamiento.*

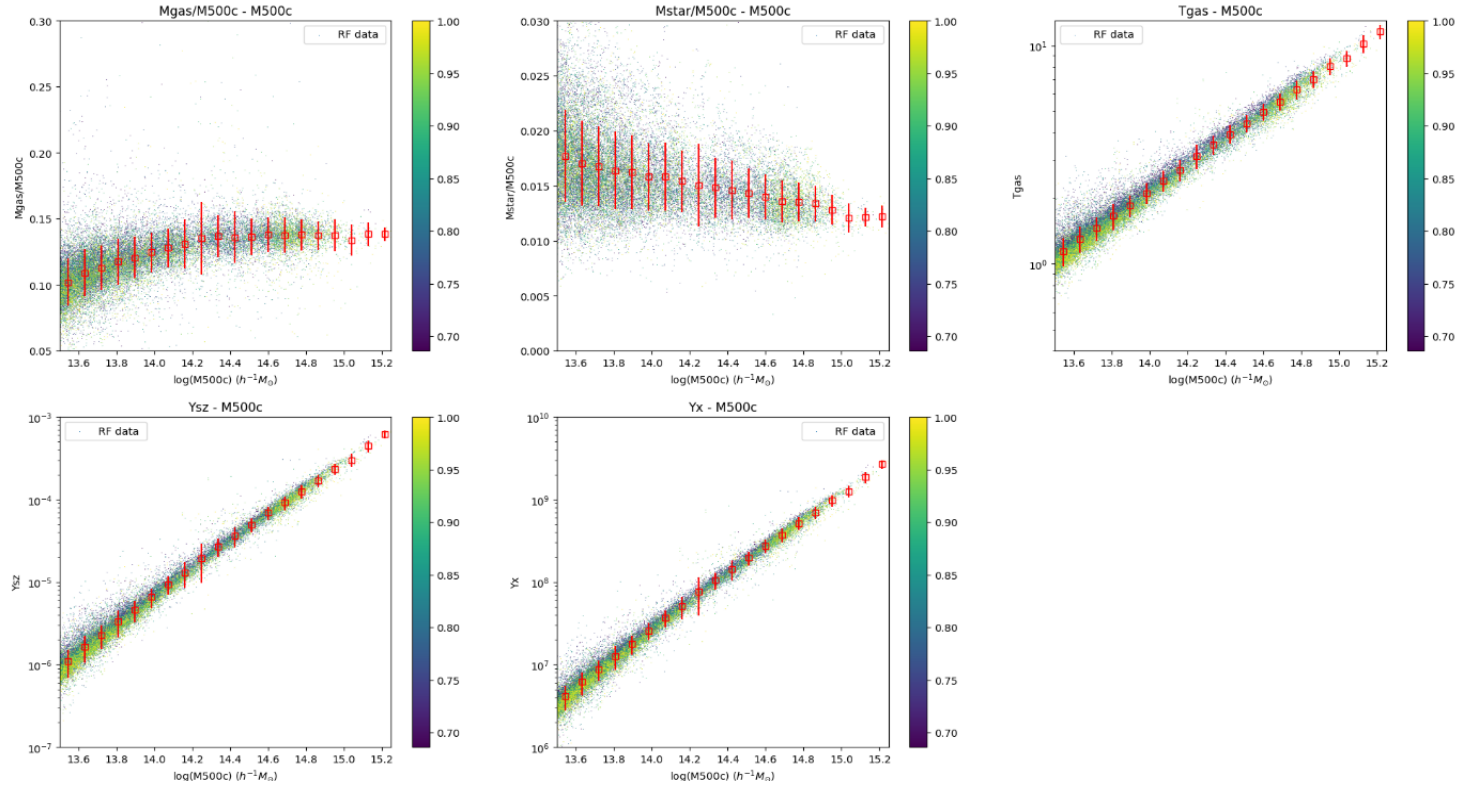

***Figura 8**: Representación de los targets con en función de log(M500) para los datos de entrenamiento, asignando a cada punto un color en función del valor de "a".*