Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Estimación de riesgo de colisión de objetos orbitantes utilizando técnicas de Inteligencia Artificial

Autor: Begoña Lozano Arrue

Tutor: Federico Cuesta Rojo

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Estimación de riesgo de colisión de objetos orbitantes utilizando técnicas de Inteligencia Artificial

Autor:
Begoña Lozano Arrue

Tutor:
Federico Cuesta Rojo
Profesor Titular

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Estimación de riesgo de colisión de objetos orbitantes utilizando técnicas de Inteligencia Artificial

Autor: Begoña Lozano Arrue
Tutor: Federico Cuesta Rojo

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar, agradecer a mi tutor Federico por ayudarme durante la realización de este trabajo.

Dar las gracias a mi familia por saber sobrellevarme todo lo que ha durado el grado y a mis amigos que han hecho los días en la universidad amenos.

Resumen

Evitar basura espacial es el día a día para las misiones espaciales orbitando alrededor de la Tierra. Para ello, la Agencia Espacial Europea lleva a cabo la monitorización de todos los objetos orbitantes alrededor de la Tierra a partir de los CDMs (Conjuction Data Message), estimando la colisión de riesgo entre la basura y los satélites a través de métodos basados en Inteligencia Artificial. A partir de esto, la ESA propone la siguiente competición: estimar la colisión de riesgo en las órbitas basándose en los CDMs.

La Inteligencia Artificial hoy en día está en alza. Una técnica utilizada es el Machine Learning donde no es necesario una programación explicita de algoritmos. Para completar el desafío propuesto se hace uso de dos métodos de Machine Learning supervisados (árboles de decisión y redes neuronales) implementados en Python.

Abstract

Nowdays spatial missions orbiting Earth have to be aware of the space debris in order to avoid it. To this end, the European Space Agency monitors of all these orbiting objects through CDMs (Conjuction Data Message). With the information contained in those messages the ESA can estimate the collision risk between the debris and the satellites. This is done using Artificial Intelligence.

Today Artificial Intelligence is on the rise. A technique used by AI is Machine Learning, where there is no need to program an algorithm explicitly. To complete the challenge proposed two methods are being used: Decision Trees and Neuronal Networks. The implementatiopn is done using Python.

Índice Abreviado

Índice

Glosario

CDM	Conjuction Data Message
CV	Cross-validation / Validación cruzada
ESA	European Space Agency / Agencia Europea del Espacio
IA	Inteligencia Artificial
ML	Machine Learning

1 Desafío: Evitar colisiones en el espacio

Evitar la basura espacial y realizar maniobras para evitarla está a la orden del día para los satélites que orbitan alrededor de la Tierra. La mayoría de ésta se encuentra monitorizada y categorizada por la Space Surveillance Network. La oficina de Spce Debris de la Agencia Espacial Europea (ESA) se encarga de apoyar distintas misiones europeas así como distintas empresas comerciales en la supervisión de la basura espacial.

La monitorización se lleva a cabo a través de lo denominado "Conjuction Data Message" (CDM), mensajes con información sobre distintos atributos de un posible evento de colisión como los son el tiempo hasta la posible colisión, el tipo de objeto o un riesgo auto-computado. Las posibles colisiones se detectan con la propagación de la orbita de cada objeto. Normalmente se publican 3 CDM por día, por lo que se puede crear una línea temporal de un posible evento. En la mayoría de los casos la oficina de *Space Debris* avisa con un día o dos de antelación a los equipos de control para que comienzan a planear la maniobra necesaria para evitar la colisión.

A partir de esto, la ESA propone el siguiente desafío [2]: crear un modelo que sea capaz de estimar el riesgo de colisión entre dos objetos orbitantes.

1.1 Estrategia actual de la ESA

Hoy en día la estrategia para evitar colisiones [19] [18] está basada en el análisis de los CDMs, proporcionados por el decimoctavo escuadrón espacial de los Estados Unidos, y el análisis de maniobras orbitales de las misiones a monitorizar.

El procedimiento seguido por la ESA es el siguiente: una vez se tienen los CDMs descargados, se propaga el estado de los perseguidores, consiguiendo un catálogo "temporal" de todos los objetos cercanos a las naves o satélites que se estén monitorizando. Tras esto, se comienzan a analizar las distintas opciones de maniobras. Para ello, utiliza dos herramientas: CORCOS (COllision Risk COmputation Software) y CAMOS (Collision Avoidance Optimization Software). El primero se encarga de calcular el riesgo de colisión entre dos objetos y el último de evaluar las distintas estrategias posibles para maniobrar.

La ESA considera que para órbitas LEO [1], el riesgo de colisión se reduce en un noventa por ciento si se considera un umbral de riesgo de 10^{-4}.

[1] LEO (Low Earth Orbit) (órbitas terrestres bajas): son órbitas de altitud menor que 2000 km.

1.2 Desafío propuesto

El modelo a diseñar utiliza CDMs de hasta dos días antes del acercamiento más cercano para predecir el riesgo de un posible evento de colisión.

Para poder clasificar si el evento es de riesgo o no riesgo se adopta el siguiente criterio: cuando el riesgo[2] es mayor que -5 se toma como un evento de alto riesgo y cuando es menor se toma de bajo riesgo, o lo que es lo mismo, en este caso no hay riesgo de colisión [5].

1.2.1 Conjunto de Datos

La ESA pone a disposición dos conjuntos de datos: *train_data.zip*, para entrenar el modelo, y *test_data.csv*, al cual se le aplicará el modelo para poder realizar las predicciones. Éstos provienen de la misma base de datos, cuya información está recogida de la misma forma, pero cada uno de ellos ha sido seleccionado para poder sobrerepresentar los casos donde existe riesgo de colisión. Cada conjunto cuenta con varios eventos únicos de encuentros cercanos, contando el set de entrenamiento con significativamente más eventos.

La principal diferencia entre los datos de entrenamiento y los de validación es la columna *time_to_tca*, que se explicará más adelante. Los datos de entrenamiento cuentan con eventos donde esta característica es menor de dos días. En el set de validación solo hay eventos cuyo *time_to_tca* es mayor que dos, debido a que en la práctica cuando quedan dos días para la posible colisión se comienza a preparar la estrategia para la maniobra a seguir.

Los conjuntos de datos son tablas donde las filas representan un solo CDM y las columnas informan sobre las características de cada CDM. Cada posible colisión está formada por varios CDM, ya que es una serie temporal de ellos, de los cuales el riego que interesa para determinar si existe o no encuentro es el último de la serie.

Descripción de las características

Cada CDM cuenta con 103 columnas o características. Existen dos tipos de atributos: con nombre propio o correspondientes al objeto objetivo o al perseguidor (mismas propiedades pero para cada objeto respectivo).

Las columnas con nombre propio son las siguientes [3][4]:

- *event_id*: identificador único por cada evento de colisión.
- *riesgo*: valor en base de 10 del riesgo autocalculado.
- *time_to_tca*: intervalo de tiempo (días) entre la creación del CDM y el momento de aproximación más cercano.
- *mission_id*: identificador de la misión que se verá afectada.
- *max_risk_estimate*: probabilidad máxima de colisión, obtenida escalando la covarianza combinada.
- *max_risk_scaling*: factor utilizado para calcular la probabilidad máxima de colisión.
- *miss_distance*: posición relativa (m) entre el objetivo y el perseguidor.
- *relative_speed*: velocidad relativa (m/s) entre el objetivo y el perseguidor.
- *relative_position_n*: posición relativa normal (m) entre el objetivo y el perseguidor en línea recta.

[2] El riesgo viene dado en base 10.

[3] Los términos relativos a la mecánica orbital se explican en el apéndice A

[4] Los términos relativos a la estadística se explican en el apéndice B

- *relative_position_r*: posición relativa radial (m) entre el objetivo y el perseguidor.
- *relative_position_t*: posición relativa (m) entre el objetivo y el perseguidor siguiendo la cónica de la órbita.
- *relative_velocity_n*: velocidad relativa normal (m/s) entre el objetivo y el perseguidor en línea recta.
- *relative_velocity_r*: velocidad relativa radial (m/s) entre el objetivo y el perseguidor.
- *relative_velocity_t*: velocidad relativa(m/s) entre el objetivo y el perseguidor siguiendo la cónica de la órbita.
- *c_object_type*: tipo de objeto que está en riesgo de colisión con el satélite.
- *geocentric_latitude*: latitud (grados) del punto del encuentro.
- *azimuth*: componente del vector velocidad relativa, en concreto el ángulo de azimuth (grados)
- *elevation*: componente del vector velocidad relativa, en concreto el ángulo de elevación. (grados).
- *mahalanobis_distance*: distancia de Mahalanobis.
- $F10$: índice de flujo de radio 10.7 cm (10221022 W/(m2m2 Hz)).
- *AP*: índice geomagnético.
- $F3M$: media móvil de 81 días de F10.7 (más de 3 rotaciones solares).
- *SSN*: número de Wolf.

Las columnas compartidas entre el perseguidor y el objetivo se distinguen por la letra que les precede: c para el perseguidos (chaser) y t para el objetivo (target). Éstas son las que se exponen a continuación, la "x" sustituye a la c y/o la t:

- *x_sigma_rdot*: covarianza, desviación estándar de la velocidad radial (m/s).
- *x_sigma_n*: covarianza, desviación estándar de la posición estándar (m).
- *x_cn_r*: covarianza, correlación entre la posición normal y posición radial.
- *x_cn_t*: covarianza, correlación entre la posición normal y posición transversal.
- *x_cndot_n*: covarianza, correlación entre la velocidad normal y posición normal.
- *x_sigma_ndot*: covarianza, desviación estándar de la velocidad normal (m/s).
- *x_cndot_r*: covarianza, correlación entre la velocidad normal y posición radial.
- *x_cndot_rdot*: covarianza, correlación entre la velocidad normal y velocidad radial.
- *x_cndot_t*: covarianza, correlación entre la velocidad normal y posición transversal.
- *x_cndot_tdot*: covarianza, correlación entre la velocidad normal y velocidad transversal.
- *x_sigma_r*: covarianza, desviación estándar de la posición radial (m).
- *x_ct_r*: covarianza, correlación entre la velocidad transversal y posición radial.
- *x_sigma_t*: covarianza, desviación estándar de la posición transversal (m).
- *x_ctdot_n*: covarianza, correlación entre la velocidad transversal y posición normal.
- *x_crdot_n*: covarianza, correlación entre la velocidad radial y posición normal.

- *x_crdot_t*: covarianza, correlación entre la velocidad radial y posición transversal.
- *x_crdot_r*: covarianza, correlación entre la velocidad radial y posición radial.
- *x_ctdot_r*: covarianza, correlación entre la velocidad transversal y posición radial.
- *x_ctdot_rdot*: covarianza, correlación entre la velocidad transversal y velocidad radial.
- *x_ctdot_t*: covarianza, correlación entre la velocidad transversal y la posición transversal.
- *x_sigma_tdot*: covarianza, desviación estándar de la velocidad transversal (m/s).
- *x_position_covariance_det*: determinante de la covarianza.
- *x_cd_area_over_mass*: coeficiente balístico (kg/mm^2).
- *x_cr_area_over_mass*: coeficiente de radiación solar (A/m), equivalente al coeficiente balístico.
- *x_h_apo*: apogeo (km).
- *x_h_per*: perigeo (km).
- *x_ecc*: excentricidad.
- *x_j2k_inc*: inclinación (grados).
- *x_j2k_sma*: semieje mayor (km).
- *x_sedr*: tasa de disipación de energía (W/kg).
- *x_span*: tamaño (m) utilizado por el algoritmo de cálculo del riesgo de colisión (se supone un diámetro mínimo de 2 m para el perseguidor).
- *x_rcs_estimate*: área (m^2)de la sección transversal del radar
- *x_actual_od_span*: duración real del intervalo (días) de actualización para la determinación de la órbita.
- *x_obs_available*: número de observaciones disponibles para la determinación de la órbita, por cada CDM.
- *x_obs_used*: número de observaciones utilizadas para la determinación de la órbita, por cada CDM.
- *x_recommended_od_span*: duración recomendada del intervalo (días) de actualización para la determinación de la órbita.
- *x_residuals_accepted*: residuos de la determinación de órbita.
- *x_time_lastob_end*: final del intervalo de tiempo (con respecto a la época de creación del CMD) desde la última observación aceptada utilizada en la determinación de la órbita.
- *x_time_lastob_start*: inicio del tiempo en días (con respecto a la época de creación del CMD) desde la última observación aceptada utilizada en la determinación de la órbita.
- *x_weighted_rms*: media cuadrática de los mínimos cuadrados utilizados en la determinación de órbitas.

2 Inteligencia Artificial

Alan Turing, considerado por muchos el padre de la IA, se preguntaba en un artículo publicado en 1950 si las máquinas podían pensar [28]. En él, entre otras cuestiones, expone un juego de imitación con una máquina así como diferentes opiniones contrarias a la suya. Llega a la conclusión de que la cuestión de si las máquinas piensan es un problema de programación principalmente y que una de las características principales será que el programador, o profesor de la máquina, podrá predecir el comportamiento del problema a analizar pero sin saber muy bien que pasa pro dentro. John McCarthy, famoso ingeniero informático, definía la Inteligencia Artificial como "la ciencia e ingeniería capaz de hacer máquinas inteligentes, en concreto, programas informáticos"[16].

Para poder resolver un problema en un ordenador se necesita un algoritmo, capaz de llevar a cabo instrucciones a partir de ciertas entradas. Para ello, se debe programar el algoritmo y proporcionarselo al ordenador. Pero, hay ocasiones en las que no es necesario programarlo, el propio ordenador puede extraer un posible algoritmo basándose en los patrones del conjunto de datos a examinar. En esto se basa la Inteligencia Artificial (IA). De forma simple, se puede definir la IA como el campo que combina la programación con conjuntos robustos de datos.

Se pueden diferenciar dos tipos de IA principalmente: débil y fuerte. La Inteligencia Artificial débil es aquella que se entrena en resolver unas tareas en específico, como Siri de Apple o los vehículos autonomos. Dentro de la IA fuerte se distinguen dos tipos: Inteligencia Artificial General (Artificial general intelligence, AGI), la máquina tendría la misma inteligencia que un humano, es decir, sería autoconsciente; y Super Inteligencia Artificial (Artificial Super Intelligence, ASI), aquella capaz de superar la inteligencia humana.

El Machine Learning (ML) es una técnica de IA donde no es necesario una programación explicita. Mediante distintos tipos de algoritmos la máquina es capaz de reconocer patrones de datos y sacar conclusiones de forma autónoma. Los algoritmos de ML se pueden clasificar en cuatro subcategorías [27]:

- Aprendizaje Supervisado (Supervised ML): a partir de un set de entrenamiento, basado en datos anteriores y etiquetas, se realiza un aprendizaje previo. El algoritmo compara los resultados obtenidos, mediante su función interna, con los esperados, modificando el modelo de algoritmo si es necesario.
- Aprendizaje Sin Supervisión (Unsupervised ML): en este caso el conjunto de datos de entrenamiento no esta ni clasificado ni etiquetado. El sistema identifica observaciones en los patrones de los datos
- Aprendizaje Semi-Supervisado(Semi-supervised ML): utilizada tanto datos etiquetados como no etiquetados en el proceso de aprendizaje.

- Aprendizaje Reforzado (Reinforcement ML): el algoritmo se relaciona con el ambiente mediante acciones y localizar los errores.

En este documento, se tratarán técnicas de aprendizaje supervisado, en concreto: Árboles de Decisión y Redes Neuronales.

2.1 Árbol de Decisión

Un Árbol de Decisión es un método de aprendizaje supervisado no paramétrico, el cual sigue una estrategia de "divide-y-venceras", utilizado para la clasificación y la regresión. Se puede considerar como una aproximación constante por partes.

Su estructura es jerárquica y recursiva, contando un un nodo raíz del cual parten las ramas y distintos nodos. Los nodos internos son de decisión, es decir, según las características de los datos en él se crean distintas ramificaciones. Los nodos finales son los llamados nodos hojas y representan los resultados del árbol.

Debido a la estrategia seguida por este algoritmo, cuánto más simple sea el Árbol de Decisión mejor serán los resultados. Al ser no parámetrico la estructura no está creada a priori, el árbol se va creando según los datos de entrada.

Las ventajas principales de un Árbol de Decisión son su fácil implementación e interpretación y el hecho de que los datos pueden ser de distinto tipo (continuos y discretos). En cambio, alguna desventaja puede ser el propenso sobreajuste, lo que hace que no se alcancen buenos resultados, pequeñas variaciones dentro de un árbol puedan llevar a soluciones muy diferentes.

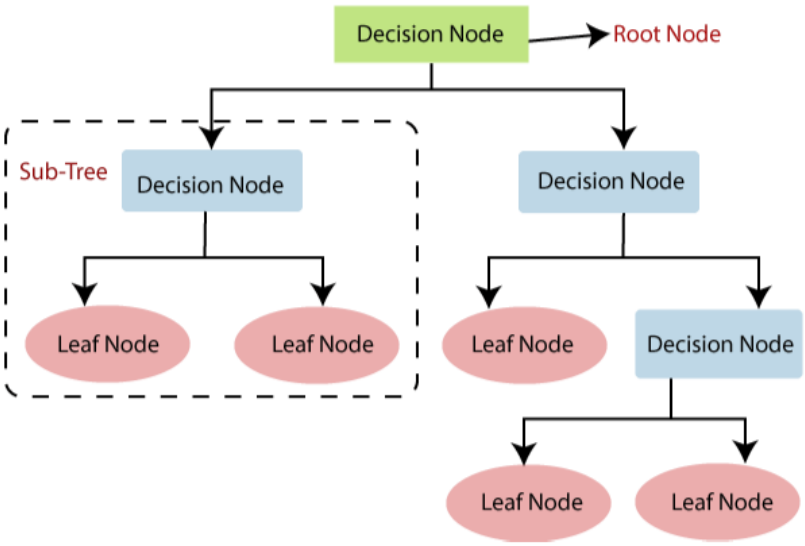


Figura 2.1 Estructura jerárquica de un Árbol de Decisión.

2.2 Redes Neuronales

Las Redes Neuronales, combinando estadística y técnicas informáticas, intenta emular el cerebro humano.

La estructura interna está compuesta de neuronas las cuales se conectan entre sí. Las conexiones entre éstas pueden ser más fuertes o más débiles, dependiendo del valor del peso que tenga asociado cada enlace. Por lo que, los pesos representan la intensidad de relación entre neuronas. Además, Las neuronas aprenden y se adaptan según los datos del proceso de entrenamiento y necesitan una función de activación, la cual ofrece el estado actual de la neurona a partir del estado anterior.

Existen diversos tipos de redes, las más comunes son las siguientes:

- Redes Neuronales de Perceptrones Multicapa (MLP): o redes de propagación hacia delante. Esta compuesta por neuronas sigmoides y por una capa de entrada, varias intermedias y una de salida.
- Redes Neuronales Convolucionales: son similares a las anteriores, pero utilizan técnicas de álgebra lineal, como la multiplicación matricial, para identificar patrones, normalmente en una imagen.
- Redes Neuronales Recurrentes: se emplean para hacer predicciones de futuro debido a su estructura de retroalimentación.

Las Redes Neuronales se asocian al Deep Learning. Pese a esto, no todas ellas se consideran dentro de esta categoría, para incluirlas las redes deberán de tener mas de tres capas intermedias.

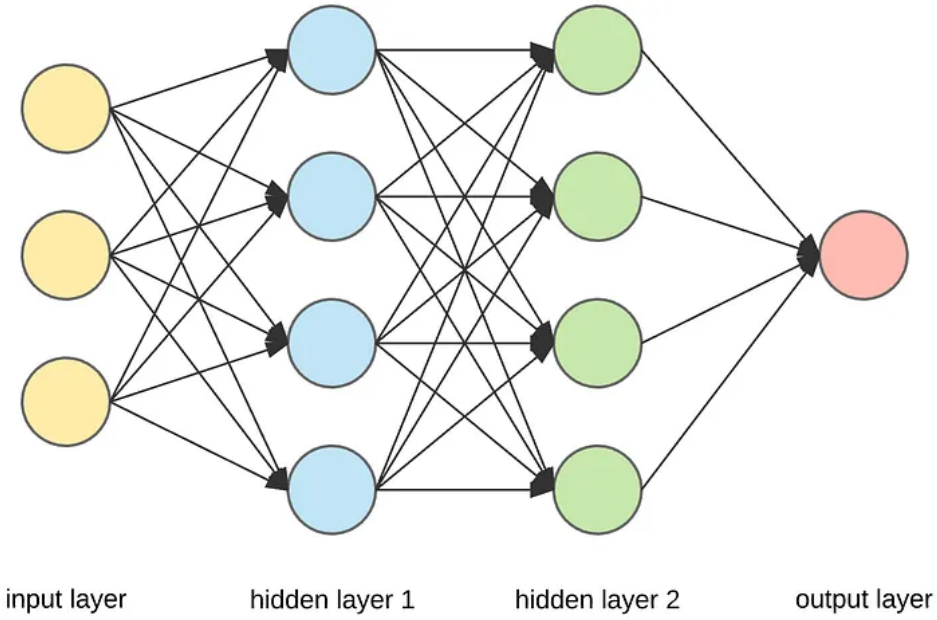


Figura 2.2 Estructura interna de las Redes Neuronales.

3 Programación

En este capítulo se presenta tanto el lenguaje de programación como las librerías utilizadas.

3.1 Python

Python, creado por Guido van Rossum, es un lenguaje de programación de alto nivel aplicable a distintos tipos de problemas. Admite los siguientes paradigmas de programación: orientada a objetos, funcional y de procedimientos. Incluye un gran número de librerías estándares propias, como protocolos de internet o procesamiento de caracteres. Además, existen librerías de terceros, como son Pandas [22], Numpy [9] o scikit-learn [7], los cuales ofrecen más áreas de trabajo en este lenguaje.

3.2 Numpy

NumPy es una librería que permite trabajar con matrices multidimensionales, como realizar operaciones matemáticas de álgebra lineal, estadísticas o transformadas de Fourier.

La principal diferencia de esta librería con lo que incluye Python de por sí, es las dimensiones fijadas de las matrices, es decir, no pueden crecer de manera dinámica. Pese a esto, NumPy permite realizar operaciones matemáticas avanzadas en un número grande de datos.

Algunos de los cálculos que se pueden realizar y utilizarán se expresa en Código 3.1.

Código 3.1 Numpy.

```
>>> import numpy as np
# Crear una matriz vacía
>>> mt_vacia = np.empty([n,m])
# Transformar un Dataframe en array
>>> df_mt = df.to_numpy()
```

3.3 Pandas

Pandas es una librería que permite el análisis y la manipulación de *DataFrames*[1]. Gracias a ésta, se pueden leer y escribir diferentes tipos de documentos externos al programa, como csv o excel.

A continuación, se exponen los comandos que se usarán.

3.3.1 Lectura y escritura de datos

Pandas permite tanto la lectura como la escritura de distintos formatos de datos como texto (CSV, Excel, HTML, LaTex, etc.) o binario (MS Excel, OpenDocument, etc.)

Código 3.2 Lectura y escritura de datos.

```
>>> import pandas as pd
# Lectura de datos. Se sustituye csv por el formato de datos a leer.
>>> pd.read_csv(r'')
# Escritura de datos. Se sustituye csv por el formato de datos a
   leer.
>>> pd.to_csv(r'')
```

3.3.2 Trabajar con DataFrames

Una vez importados los datos, se puede realizar distintas operaciones, como por ejemplo juntar distintas series de datos o extraer columnas de un conjunto.

Código 3.3 Trabajar con DataFrames.

```
>>> import pandas as pd
# Juntar serie de datos
>>> df = pd.concat([
    serie1,
    serie2
    ],
    axis=1, join="inner"
)
# Escoger una columna de un dataframe
>>> df_columna = df.iloc[a,b] # localizar por indice
>>> df_columna = df.loc[:,''] # localizar por etiqueta
# Transformar un array en un DataFrame
>>> mt_df = pd.DataFrame(df)
```

[1] Estructura de datos con columnas y filas etiquetadas.

3.4 Matplotlib

Matplotlib es una librería para crear distintas visualizaciones, estática, animadas o interactivas. Permite exportar estas figuras además de personalizarlas previamente.

Código 3.4 Matplotlib.

```
>>> import matplotlib.pyplot as plt
>>> fig, axs = plt.subplots(m, n)
>>> plt.legend()
>>> plt.title("")
>>> plt.show()
# Guardar una fifura
>>> plt.savefig(r'')
```

3.5 Scikit-learn

Scikit-learn es una librería basada en uso de ML, tanto supervisado como no supervisado, y permite el procesamiento o el análisis de los resultados obtenidos.

3.5.1 Árboles de Decisión

Los Árboles de Decisión de scikit-learn son, como todo Árbol de Decisión, un método de aprendizaje supervisado no paramétrico utilizado para la clasificación y regresión, el cual crea un modelo que es capaz de predecir a través de normas de decisión simples.

Decision Tree Classifier

El clasificador del Árbol de Decisión es capaz de producir una multi-clasificación a partir de un conjunto de datos. Este método es ML supervisado, por lo que primero debe de llevar a cabo un entrenamiento. Para ésto, la función cuenta con dos entradas: una matriz de m muestras y n características y una matriz con m clases.

Una vez entrenado el clasificador, éste se puede utilizar para predecir las categorías de otro set de datos con la misma estructura.

Código 3.5 DecisionTreeClassifier.

```
>>> from sklearn import tree
>>> X = [m_samples, n_features]
>>> Y = [m_class,]
>>> clasificador = tree.DecisionTreeClassifier()
>>> clf = clasificador.fit(X, Y)
>>> Z = [m_samples, n_features]
>>> clf.predict(Z)
```

El *DecisionTreeClassifier* cuenta con distintos parámetros internos que se pueden ajustar, algunos de mayor interés son:

- *criterion*: mide la calidad de la división de las particiones del árbol. Puede ser de tres tipos: "gini", "entropy", "*log_loss*".
- *splitter*: estrategia utilizada para elegir la división de cada nodo. Hay dos tipos "best", estrategia que elige la mejor divisón, y "random", lo hace de forma aleatoria.
- *max_features*: número de características consideradas cuando se busca la mejor división. Puede ser un número, entero o decimal, o seguir tres estrategias: "auto","sqrt" o "log2".

Código 3.6 Parámetros del DecisionTreeClassifier.

```
tree.DecisionTreeClassifier(*, criterion='gini', splitter='best',
    max_depth=None, min_samples_split=2, min_samples_leaf=1,
    min_weight_fraction_leaf=0.0, max_features=None, random_state=
    None, max_leaf_nodes=None, min_impurity_decrease=0.0,
    class_weight=None, ccp_alpha=0.0)
```

Decision Tree Regressor

Esta función utiliza la regresión y espera como entradas valores numéricos decimales. Sigue el mismo procedimiento que con el clasificador, es decir, primero se entrena y después se puede usar para predecir.

Código 3.7 DecisionTreeRegressor.

```
>>> from sklearn import tree
>>> X = [m_samples, n_features]
>>> Y = [m_class,]
>>> clasificador_regresor = tree.DecisionTreeRegressor()
>>> clf_reg = clasificador_regresor.fit(X, Y)
>>> Z = [m_samples, n_features]
>>> clf_reg.predict(Z)
```

Sus parámetros internos relevantes para este estudio es:

- *criterion*: mide la calidad de la división de las particiones del árbol. Puede ser: "*square_error*", "*friedman_mse*" o "*absolute_error*".
- *splitter*: estrategia utilizada para elegir la división de cada nodo. Hay dos tipos "best", estrategia que elige la mejor división, y "random", lo hace de forma aleatoria.
- *max_features*: número de características consideradas cuando se busca la mejor división. Puede ser un número, entero o decimal, o seguir tres estrategias: "auto","sqrt" o "log2".

Código 3.8 Parámetros del DecisionTreeRegressor.

```
tree.DecisionTreeRegressor(*, criterion='squared_error', splitter='
    best', max_depth=None, min_samples_split=2, min_samples_leaf=1,
    min_weight_fraction_leaf=0.0, max_features=None, random_state=
    None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha
    =0.0)
```

3.5.2 Redes Neuronales

Las Redes Neuronales que ofrece esta librería son redes perceptrón multicapa (MLP). Éstas son un algoritmo de machine learning de aprendizaje supervisado que aprende una función a partir de un conjunto de datos de entrenamiento dados y un objetivo. Entre la capa de entrada y salida existen múltiples capas intermedias.

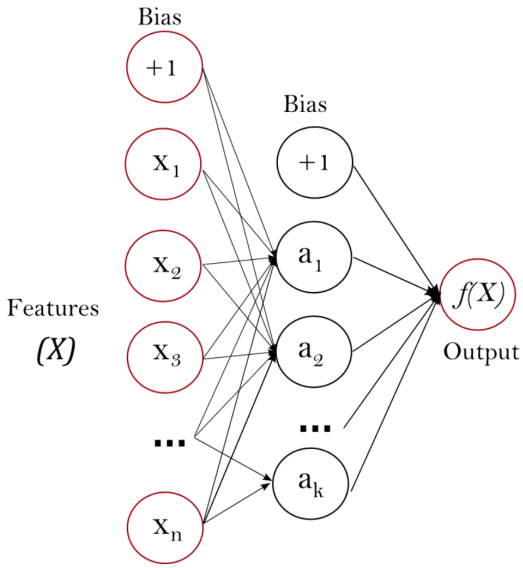


Figura 3.1 Ejemplo de una capa intermedia en una Red MLP.

MLPClassifier

Este clasificador implementa un algoritmo MLP, el cual entrena utilizando utilizando *backpropagation*, es decir, ajusta los parámetros de la red para reducir la tasa de error.

Entrena a partir de dos sets de datos: una matriz de n muestras y m características, que contiene las muestras, y una matriz de n muestras, con las clases que se requieran. Tras el entrenamiento, se puede llevar a cabo las predicciones del nuevo conjunto de datos.

Código 3.9 MLPClassifier.

```
>>> from sklearn import MLPClassifier
>>> X = [m_samples, n_features]
>>> Y = [m_class,]
>>> clasificador = MLPClassifier()
>>> clf = clasificador.fit(X, Y)
>>> Z = [m_samples, n_features]
>>> clf.predict(Z)
```

Algunos parámetros internos de este clasificador son:

- *hidden_layer_sizes*: numero de neuronas en las capas intermedias.
- *activation*: función de activación de las capas intermedias. Existen cuatro tipos: "identity", útil para casos lineales ($f(x) = x$); "logistic", función sigmoide ($f(x) = 1/(1 + exp(-x))$); "tanh", función tangente hiperbólica ($f(x) = tanh(x)$); "relu", función rectificadora ($f(x) = max(0,x)$).

- *solver*: método utilizado para optimización de los pesos. Son: "lbfgs", método quasi-Newtoniano; "sgd", descenso de gradiente estocástico; y "adam", descenso de gradiente estocástico propuesto por Kingma, Diederik, y Jimmy Ba.
- *learning_rate*: tasa de actualización de los pesos, solo es utilizado para el solver "sgd". Hay tres tipos: "constant", tasa constante; "invscaling", decrecimiento de la tasa para cada paso del tiempo; "adaptive", mantienen la tasa constante siempre y cuando no existan pérdidas en el entrenamiento.

Código 3.10 Parámetros del MLPClassifier.

```
>>> neural_network.MLPClassifier(hidden_layer_sizes=(100,),
   activation='relu', *, solver='adam', alpha=0.0001, batch_size='
   auto', learning_rate='constant', learning_rate_init=0.001,
   power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol
   =0.0001, verbose=False, warm_start=False, momentum=0.9,
   nesterovs_momentum=True, early_stopping=False,
   validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
    n_iter_no_change=10, max_fun=15000)
```

MLPRegressor

Este método de regresión implementa una Red perceptón multicapa que entrena utilizando *backpropagation* sin una función de activación en la capa de salida. Por ello, usa el error cuadrático como la función de pérdida.

Código 3.11 MLPRegressor.

```
>>> from sklearn import MLPRegressor
>>> X = [m_samples, n_features]
>>> Y = [m_class,]
>>> clasificador_regresor = MLPRegressor()
>>> clf_reg = clasificador_regresor.fit(X, Y)
>>> Z = [m_samples, n_features]
>>> clf.predict(Z)
```

Los parámetros internos son los mismo que para la Red clasificadora.

Código 3.12 Parámetros del MLPRegressor.

```
>>> neural_network.MLPRegressor(hidden_layer_sizes=(100,),
   activation='relu', *, solver='adam', alpha=0.0001, batch_size='
   auto', learning_rate='constant', learning_rate_init=0.001,
   power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol
   =0.0001, verbose=False, warm_start=False, momentum=0.9,
   nesterovs_momentum=True, early_stopping=False,
   validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
    n_iter_no_change=10, max_fun=15000)
```

3.5.3 Preprocesadores Estimadores

Scikit incluye un paquete *sklearn.preprocessing* con funciones para poder transformar los datos, como estandarizar, normalizar o escalar en un rango.

Standard Scaler

La estandarización es el proceso por el cual se hace que los datos tengan una distribución normal, es decir, los datos deben tener media cero y varianza unidad. La expresión matemática que consigue esto es la mostrada en (3.1), donde z es elvalor transformado, x es el valor a transformar, u es la media de los datos y s es la desviación estándar.

$$z = \frac{x-u}{s} \tag{3.1}$$

Código 3.13 Standard Scaler.

```
>>> from sklearn import preprocessing
>>> from sklearn.preprocessing import MaxAbsScaler
>>> X_train = [m, n]
>>> scaler = StandardScaler()
>>> X_scaled = scaler.fit_transform(X_train)
```

MinMax Scaler

Este estimador transforma los datos dentro de un rango dado, según Ecuación 3.2 donde

$$z = x_{desviac} \cdot (x_{max} - x_{min}) + x_{min} = \frac{x - x_{min}}{x_{max} - x_{min}} \cdot (x_{max} - x_{min}) + x_{min} \tag{3.2}$$

Código 3.14 MinMax Scaler.

```
>>> from sklearn import preprocessing
>>> from sklearn.preprocessing import MinMaxScaler
>>> X_train = [m, n]
>>> scaler = MinMaxScaler(feature_range=(min, max))
>>> X_scaled = scaler.fit_transform(X_train)
```

MaxAbs Scaler

El *MaxAbsScaler* transforma cada característica (columna) según su valor máximo. No centra los datos y tampoco tiene en cuenta si la matriz es dispersa.

Código 3.15 MaxAbs Scaler.

```
>>> from sklearn import preprocessing
>>> from sklearn.preprocessing import MaxAbsScaler
>>> X_train = [m, n]
>>> scaler = MaxAbsScaler()
>>> X_scaled = scaler.fit_transform(X_train)
```

4 Resolución del desafío

Se va a completar al desafío utilizando los métodos de aprendizaje supervisado explicados en el Capítulo 3. En rasgos generales, se comenzará leyendo los datos y extrayendo lo que serán las matrices para el entrenamiento, una de características y otra de la estimación del riesgo, y para la predicción. Se aplicarán distintos preprocesadores y métodos de análisis previos para ver como afectan a los resultados. Tras esto, se entrenará, medirá y puntuará el modelo elegido. Además, se dibujarán curvas de aprendizaje y de análisis para los modelos con las mejores predicciones.

La ESA publicó un artículo [5] con varias observaciones respecto al desafío. En él, se establece, en un principio, que existen varias características de los conjuntos de datos son más importantes en el análisis, las cuales son: *event_id*, *time_to_tca*, *mission_id*, *miss_distance*, *c_object_type*, *t_span*, *max_risk_estimate*, *max_risk_scaling*, *mahalanobis_distance*, *c_obs_used* y *c_position_covariance_det*. Más adelante, se exponen en una tabla que hay una serie de atributos que reducen el error a la hora de predecir el riesgo, los cuales son: *max_risk_estimate*, *mahalanobis_distance*, *c_sigma_t*, *c_position_covariance_det*, *c_sedr*, *max_risk_scaling*, *c_sigma_rdot*, *miss_distance*, , *c_sigma_n*, *time_to_tca*, *c_sigma_r*, *SSN*, *c_obs_used*, *c_sigma_ndot*, *relative_position_n*, *relative_position_r*, *relative_speed*, *c_crdot_t* y *c_recommended_od_span*. Se comprobarán como afectan estos comentarios a los modelos que se implementen, el primer set de columnas se denominará "Características del artículo" y el segundo "Características de la tabla".

En los datos para validar los modelos también aparece una columna riesgo, de esta forma se puede comparar lo estimado con lo verdadero. Así, se puede obtener cómo de exacto es cada procedimiento utilizado. Por tanto, los resultados obtenidos son la comparación de las predicciones de los métodos frente a el riesgo estimado verdadero dado por el *test data*.

4.1 Árboles de Decisión

Los Árboles de Decisión solo pueden trabajar con valores del tipo *float32*. Para ello, se transforman los datos de entrenamiento y estimación, los valores que son mayores de el máximo de *float32* se cambian a ese valor máximo (3.4028235e38).

4.1.1 Clasificación

Los árboles clasificadores pueden trabajar con categorías como caracteres o números enteros. En este documento, se van a considerar las categorías como los números enteros del riesgo. Por ello, se truncan dejando la parte entera del valor, los valores de la columna de *riesgo* de los dos conjuntos de datos (entrenamiento y predicción) (Tabla 4.1).

Tabla 4.1 Clasificación en números enteros de la columna de riesgo.

Columna original	→	Columna Transformada
riesgo	→	riesgo
-7,561299467	→	-7
-9,248105312	→	-9
-10,85047299	→	-10
-30	→	-30
...		...

Características del artículo

Se aplica los escaladores explicados en 3.5.3 obteniendo los resultados mostrados en la Tabla 4.2. Se observa como el mayor porcentaje de aciertos se tiene para los datos sin escalar y después para *MaxAbs Scaler*. Esto se debe a que los CDMs tienen valores muy dispares, unos muy grandes y otros muy pequeños, y *MaxAbs Scaler* escala cada columna de forma independiente, por lo que transformará cada atributo conforme a los valores de éste, sin tener cuentas otros datos.

Tabla 4.2 Aciertos y fallos de los escaladores para las características del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	94,11	5,89	23043	1441
StandardScaler	33,73	66,27	8259	16225
MaxAbs Scaler	63,75	36,25	15608	8876
MinMax Scaler (-10,10)	32,29	67,70	7907	16577

Además, se prueba como afectan los distintos parámetro internos del *DecisionTreeClassifier* (3.5.1).Para poder deducir conclusiones de los distintos parámetros se obtienen los resultados de Tabla 4.3. Se puede observar como para *criterion* la versión *log_loss* es la que produce mejor resultados. En éste la medida de cualidad de la división del árbol tiene la expresión mostrada en Ecuación 4.1, lo cual va incluido en la expresión de calidad de la división de los árboles para scikit(Ecuación C.1 en el apéndice C). En cuanto a la estrategia para decidir la rama en cada nodo (*splitter*) *random* consigue menor número de acierto, ya que la decisión se hace de forma aleatoria. $max\_features = log2$ ($max\_features = log2(n\_features)$) tiene el porcentaje verdadero más elevado.

Tabla 4.3 Resultados del cambio de los parámetros del clasificador del Árbol de Decisión para las características del artículo.

Iteracion		Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Criterion	gini	94,10	5,90	23039	1445
	entropy	94,80	5,20	23210	1274
	log_loss	94,85	5,15	23224	1260
Splitter	best	94,07	5,93	23031	1453
	random	75,59	24,41	18507	5977
Max Features	auto	68,89	31,11	16868	7616
	sqrt	67,91	32,09	16627	7857
	log2	69,24	30,76	16952	7532

$$H(Q_m) = -\sum p_{mk} log(p_{mk}) \tag{4.1}$$

A partir de ahora, cuando se mencionen "mejores parámetros" será la combinación de los mejores parámetros anteriores,

Código 4.1 "Mejores parámetros" del Árbol de Decisión para las columnas del artículo.

```
>>> clasificador_opt = DecisionTreeClassifier(
                          criterion = "log_loss",
                          splitter = "best",
                          max_features = "sqrt")
```

Una vez hecho este análisis previo, se entrena, puntúa y compara el clasificador para los casos: predeterminado, escalando con el mejor estimador (estudiado anteriormente), mejores parámetros y datos escalados con los mejores parámetros.

Tabla 4.4 Aciertos y fallos de las distintas iteraciones para las características del artículo del Árbol clasificador.

	Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Predeterminado	94,17	5,83	23056	1428
MaxAbs Scaler	63,87	36,13	15638	8846
Mejores Param Sin Escalar	77,50	22,50	18974	5510
Mejores Param Escalado	67,19	32,81	16451	8033

Los peores resultados se obtienen cuando se escalan los datos con la combinación de los mejores parámetros internos. El árbol predeterminado tiene los mejores porcentajes, ya que éste viene optimizado desde la librería. Se debe tener en cuenta que estos valores de aciertos y fallos se dan cuando se categoriza el riesgo en valores de número entero (desde -1 hasta -30). Si se categoriza en "riesgo de colisión" o "no riesgo de colisión" los porcentajes varían, ya que cuando el valor de riesgo es menor que cinco se considera "no riesgo" y los resultados obtenidos con los árboles, al compararlos con los valores verdaderos, no son tan diferentes como para cambiar la categoría de riesgo verdadero. Por ejemplo, clasificando en las dos categorías anteriores, los árboles obtienen un 100 % de aciertos.

Como se ha mencionado antes, también es posible dibujar las curvas de aprendizaje y de análisis del modelo. Se van a mostrar las gráficas para el modelo con mejores resultados. La función *learning_curve* genera un nuevo número de muestras a partir del conjunto de entrada de muestras de entrenamiento dadas al método de aprendizaje. Los puntos que aparecen son los subconjuntos de validación dentro del entrenamiento.

En la Figura 4.1 se puede ver una representación de la puntuación (sobre uno) que recibe el árbol de clasificación frente al número de muestras de entrenamiento empleadas. Conforme aumenta el número de muestras, el método va aumentando su puntuación a la hora de entrenar.

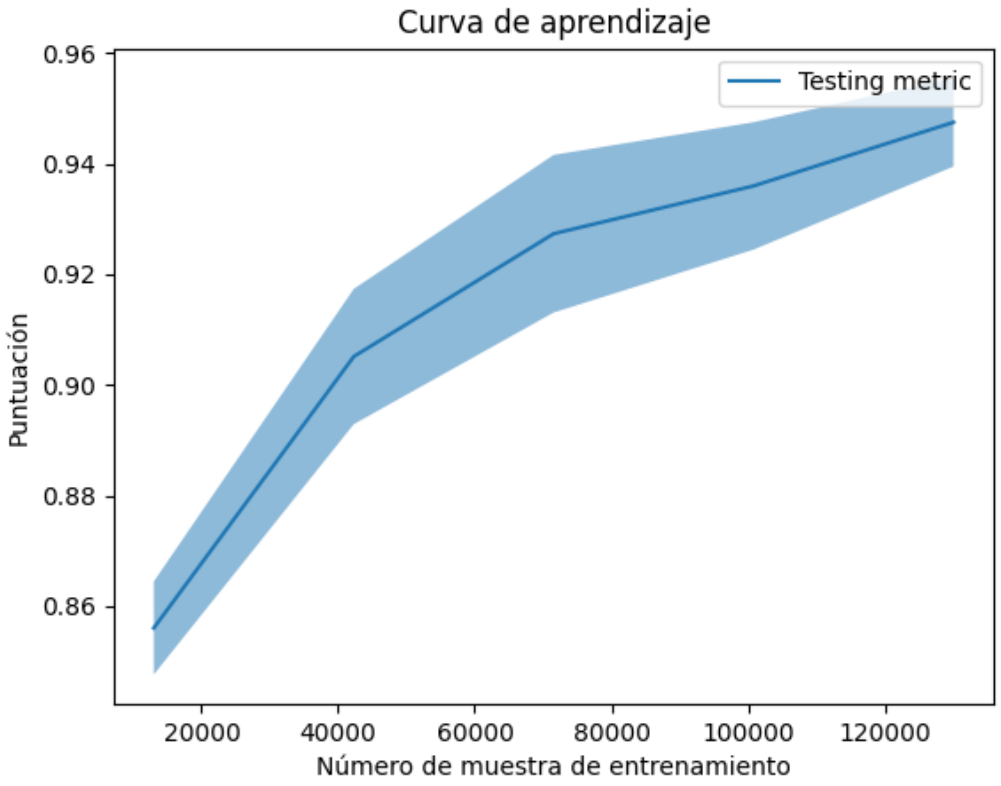
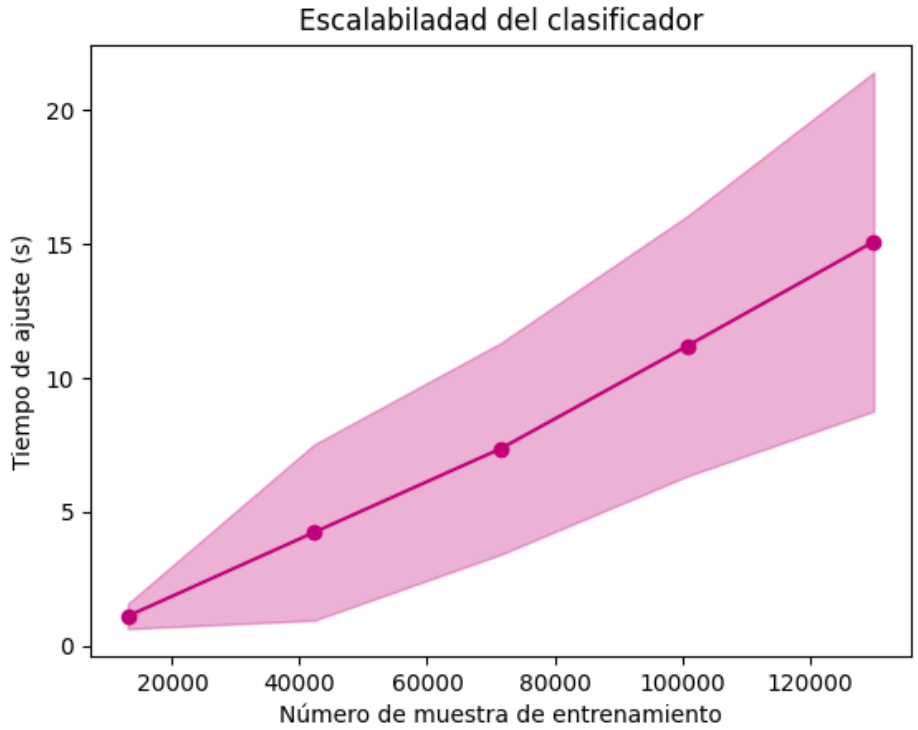
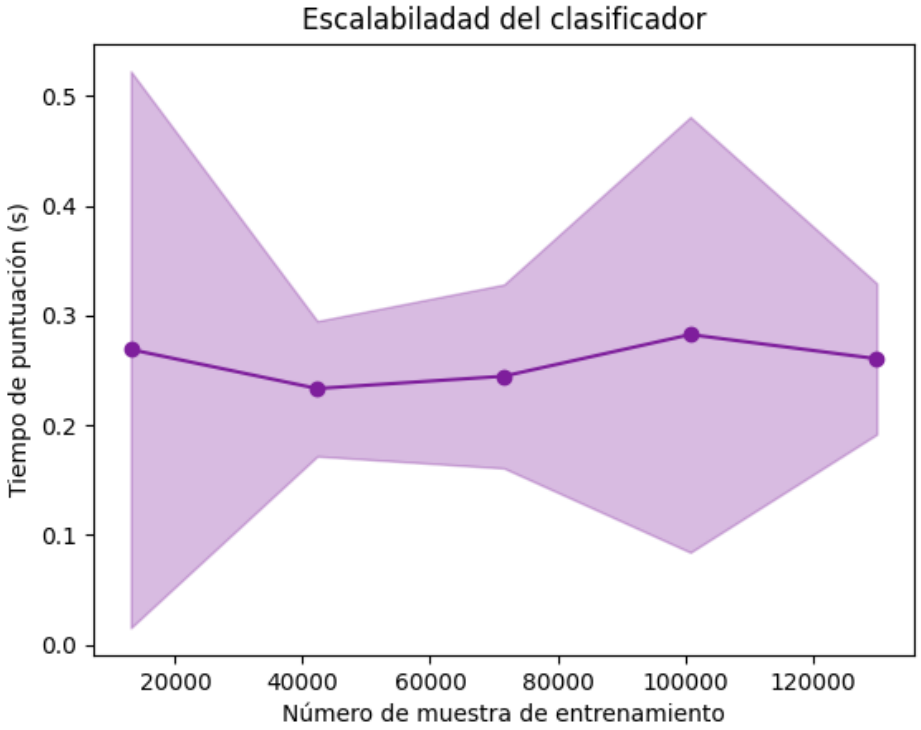


Figura 4.1 Curva de aprendizaje de las características del artículo para el Árbol clasificador.

El análisis de complejidad (Figura 4.2), muestra como el tiempo de ajuste aumenta conforme lo hace el número de muestras, lo que tiene sentido pues existen más filas para entrenar el modelo . En cuanto al tiempo de puntuación, varía muy poco frente al número de muestras.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.2 Análisis de complejidad del Árbol clasificador para las columnas del artículo.

En la curva de la Figura 4.3 se puede buscar el punto donde la validación cruzada no aumenta, es decir, donde exactitud parece estabilizarse para un tiempo de ajuste de 10 s.

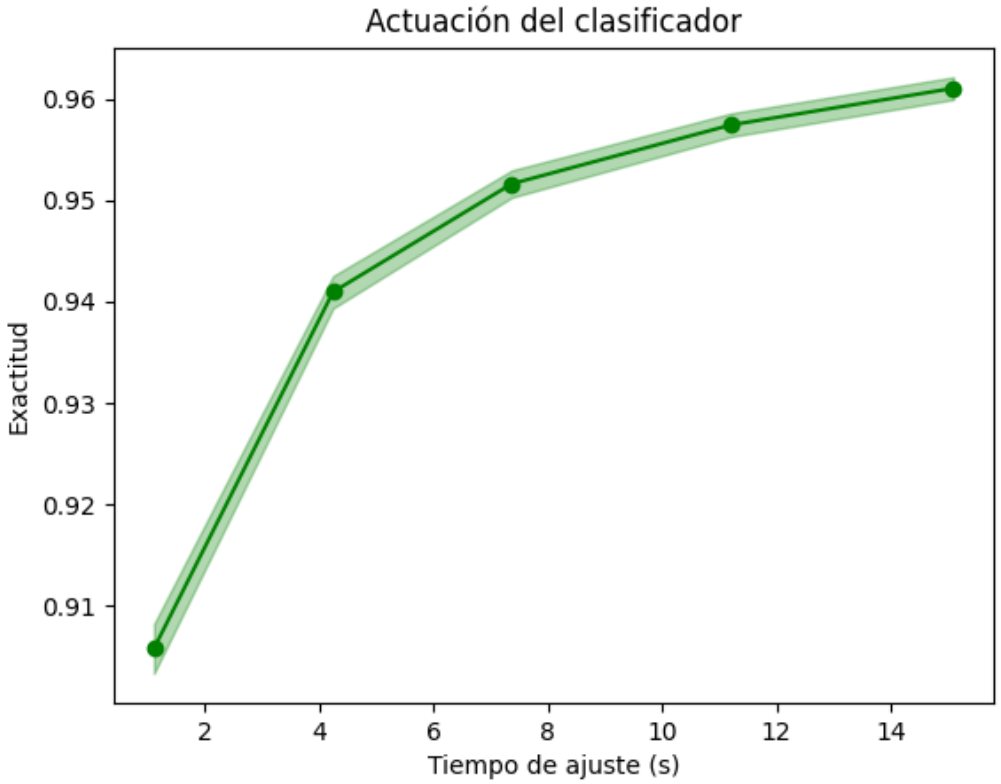


Figura 4.3 Actuación del Árbol clasificador para las características del artículo.

Características de la tabla del artículo

Al aplicar los distintos escaladores al conjunto de datos de entrenamiento y validación se obtienen los aciertos y fallos mostrados en Tabla 4.5. Se observa como el mayor porcentaje de aciertos se tiene para los datos sin escalar y después para *MaxAbs Scaler*, lo cual tiene la misma explicación que en el apartado anterior.

Tabla 4.5 Aciertos y fallos de los escaladores para las características de la tabla del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	94,05	5,95	23026	1458
StandardScaler	41,52	58,48	10165	14319
MaxAbs Scaler	63,92	36,08	15651	8833
MinMax Scaler (-10,10)	31,45	68,55	7699	16785

Tabla 4.6 Resultados del cambio de los parámetros del clasificador del Árbol de Decisión para las características de la tabla del artículo.

Iteracion		Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Criterion	gini	94,04	5,96	23024	1460
	entropy	94,32	5,68	23093	1391
	log_loss	94,31	5,69	23092	1392
Splitter	best	94,09	5,91	23037	1447
	random	64,72	35,28	15846	8638
Max Features	auto	60,27	39,73	14757	9727
	sqrt	61,28	38,72	15004	9480
	log2	56,11	43,89	13739	10745

En este caso, la combinación de mejores parámetros es la mostrada en Código 4.2, siendo el número máximo de atributos elegidos según $max\_features = log2(n\_features)$ y $criterion = entropy$ sigue el mismo procedimiento que log_loss.

Código 4.2 "Mejores parámetros" del Árbol de Decisión para las características de la tabla.

```
>>> clasificador_opt = DecisionTreeClassifier(
                        criterion = "entropy",
                        splitter = "best",
                        max_features = "log2")
```

A continuación, se comparan los resultados obtenidos con distintos parámetros en los clasificadores.

Tabla 4.7 Aciertos y fallos de las distintas iteraciones del Árbol clasificador para las características de la tabla del artículo.

	Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Predeterminado	94,11	5,89	23041	1443
MaxAbs Scaler	63,88	36,12	15640	8844
Mejores Param Sin Escalar	66,08	33,92	16180	8304
Mejores Param Escalado	67,52	32,48	16531	7953

Las mejores estimaciones se obtienen para los parámetros predeterminados del Árbol clasificador, ya que éstos viene optimizados por la librería. Aún así, se ve como estos cambios aumentan el número de aciertos cuando se escalan los datos y se toman los mejores parámetros analizados antes. Los fallos se dan para valores lejanos al límite de las categorías de riesgo comentadas anteriormente (-5), por lo que, si se clasifican en "riesgo" o "no riesgo" se consigue prácticamente un 100 % de aciertos.

En la Figura 4.22 se ve como al aumentar el número de filas para el entrenamiento, aumenta la puntuación del modelo, por tanto, la tendencia es creciente pero con menor pendiente. A su vez, el árbol tiene un tiempo de ajuste superior a cuando se trabaja con las columnas del artículo, ya que ahora hay más características (columnas) que tener en cuenta ((a) Figura 4.5). El tiempo de puntuación se puede considerar igual para todos los subconjuntos, porque el eje y de la gráfica ((b) Figura 4.5). Además, la actuación del clasificador (Figura 4.6) mejora al

aumentar el tiempo de ajuste del modelo, ya que cuenta con más tiempo para poder adapterse mejor.

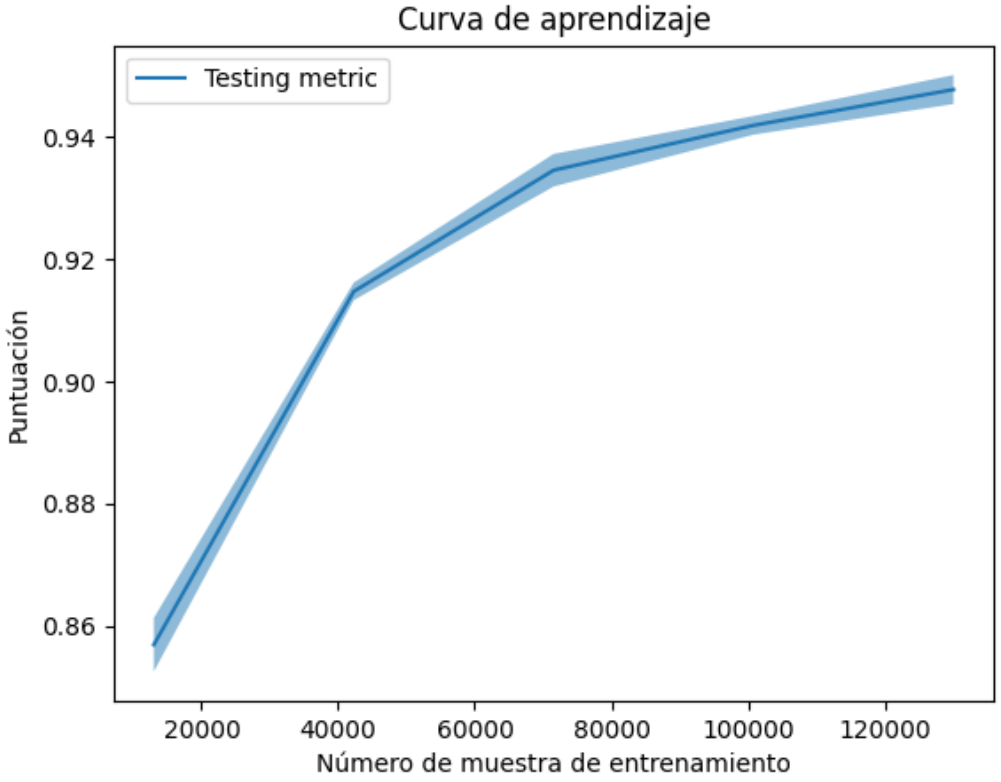
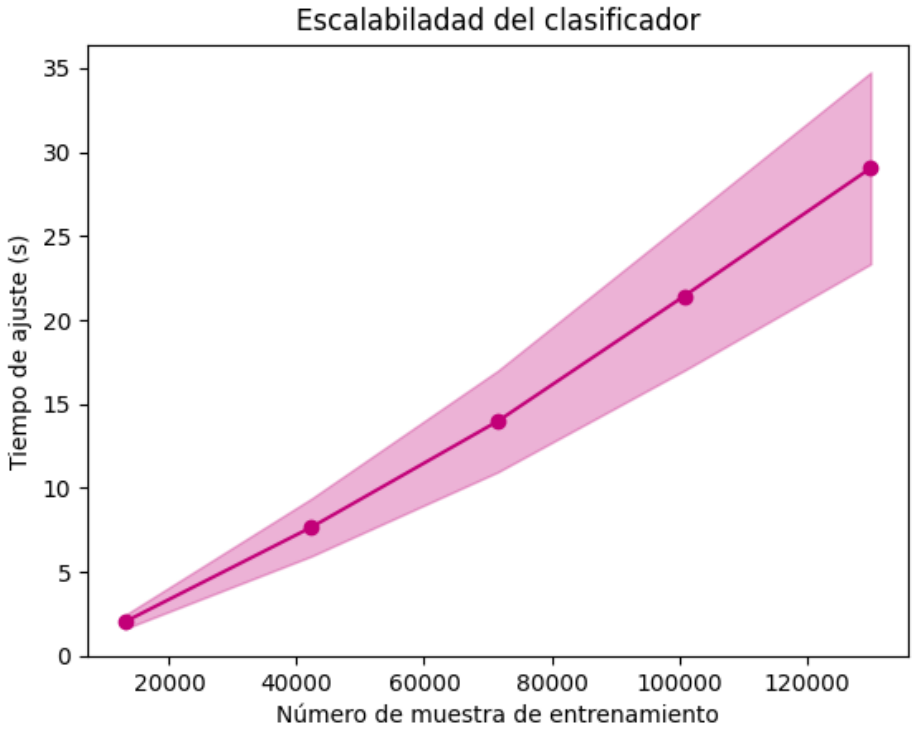
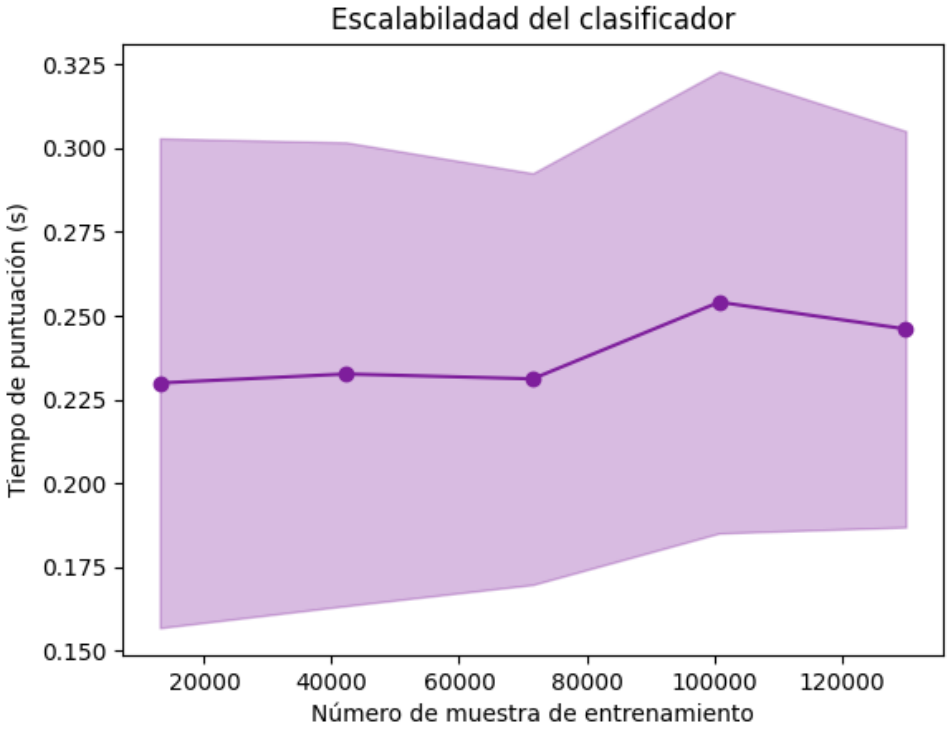


Figura 4.4 Curva de aprendizaje de las características de la tabla del artículo para el Árbol clasificador.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.5 Análisis de complejidad del Árbol clasificador para las características de la tabla del artículo.

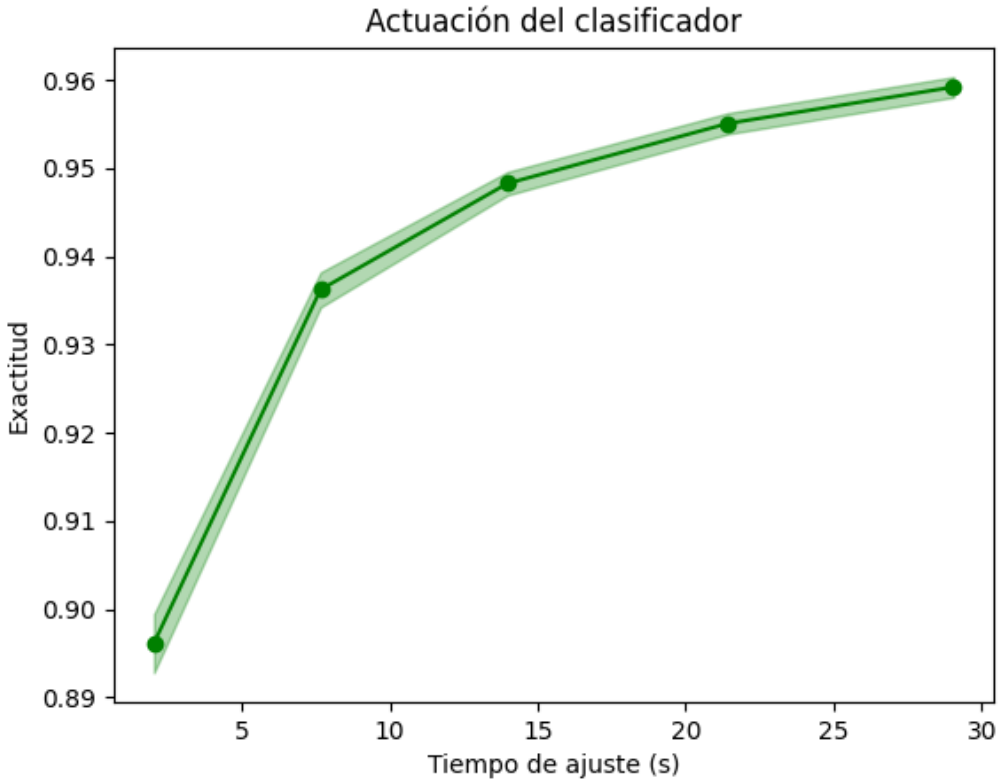


Figura 4.6 Actuación del Árbol clasificador para las características de la tabla del artículo.

Todas las características

Para todas los atributos dados en el conjunto de datos de entrenamiento, el porcentaje de aciertos al validar el modelo al aplicar escaladores es mayor para *MaxAbs Scaler*. Entre el *Standard Scaler* y el *MinMax Scaler*, el porcentaje es prácticamente el mismo. Sin embargo, el número de aciertos (contador verdadero) es algo mayor para el proceso de escalamiento entre un mínimo y un máximo.

Tabla 4.8 Aciertos y fallos de los escaladores para las todas las características.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	92,87	7,13	22738	1746
StandardScaler	32,52	67,48	7963	16521
MaxAbs Scaler	62,00	38,00	15181	9303
MinMax Scaler (-10,10)	32,62	67,38	7986	16498

En la Tabla 4.9 se exponen la precisión del método al modificar parámetros internos. El criterio *log_loss* es el que ofrece las mejores estimaciones. Para la estrategia utilizada para de elección la división de cada nodo (*splitter*), "best" tiene el porcentaje de aciertos superior, ya que el otro modo de elección es aleatoria. En cuanto al número de características a considerar, la forma *sqrt*, donde se toma el número de columnas a tener en cuenta para la división como la raíz cuadrada del número de características, da el mayor número de aciertos.

Tabla 4.9 Resultados del cambio de los parámetros del clasificador del Árbol de Decisión para todas las características.

Iteracion		Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Criterion	gini	92,99	7,01	22767	1717
	entropy	94,12	5,88	23044	1440
	log_loss	93,93	6,07	22998	1486
Splitter	best	92,99	7,00	22768	1716
	random	69,13	30,82	16937	7547
Max Features	auto	43,87	56,13	10740	13744
	sqrt	49,50	50,50	12119	12365
	log2	38,89	61,11	9522	14962

En este caso, los mejores parámetros son:

Código 4.3 "Mejores parámetros" del Árbol de Decisión para todas las características.

```
>>> clasificador_opt = DecisionTreeClassifier(
                          criterion = "entropy",
                          splitter = "best",
                          max_features = "sqrt")
```

Ahora, se realizan las mismas iteraciones que para los datos anteriores, pero al tratarse de todas las columnas se le añade dos más: en la primera se establece que un clasificador se entre teniendo como máximo 11 características de entre todas las que se le dan como entrada y en la segunda se aplica el *SelectKBest*.

Tabla 4.10 Aciertos y fallos de las distintas iteraciones del Árbol clasificador para todas las características.

	Porcentaje Acierto	Porcentaje Fallo	Contador Acierto	Contador Fallo
Predeterminado	92,72	7,28	22702	1782
MaxAbs Scaler	62,14	37,86	15215	9269
Mejores Param Sin Escalar	44,34	55,66	10857	13627
Mejores Param Escalado	70,78	29,22	17330	7154
Max features k=15	51,92	48,08	12713	11771
Selección Best k=15 features	34,60	65,40	8471	16013

Para la selección de las mejores características, en este caso $k = 15$, se obtienen los siguientes atributos: $time\_to\_tca$, $mission\_id$, c_object_type, $c\_time\_lastob\_start$, t_sigma_r, $c\_position\_covariance\_det$, c_sigma_r, t_sigma_t, c_sigma_t, t_sigma_n, c_sigma_n, c_sigma_rdot, t_sigma_tdot, c_sigma_tdot y t_sigma_ndot.

Analizando la Tabla 4.10, se concluye que el preprocesador *SelectKBest* pese a la buena elección de las características con mayor pesos, cuando se valida el árbol entrenado con dicho, éste no realiza un buen trabajo. La iteración *Max Features*, por la cual se trata de que el árbol considere simplemente 15 columnas, tampoco consigue buenos resultados. La combinación

de mejores parámetros y datos escalados consigue un número de aciertos aceptable. El Árbol clasificador predeterminado por scikit es el que más predicciones verdaderas hace.

La puntuación del modelo es mayor cuanto más muestras hay para entrenarlo (Figura 4.7). En cuanto a la escalabilidad (Figura 4.8), el tiempo de ajuste llega a alcanzar los minutos, pues se está trabajando con muchas filas y columnas, y el tiempo de puntuación es similar para todos los subconjuntos pero tiene un valor superior al de los clasificadores anteriores. Además, como debe ocurrir, la exactitud aumenta cuando se deja más tiempo de ajuste y el valor de porcentajes de aciertos al realizar la validación es similar a la exactitud que se consigue en el entrenamiento (Figura 4.9).

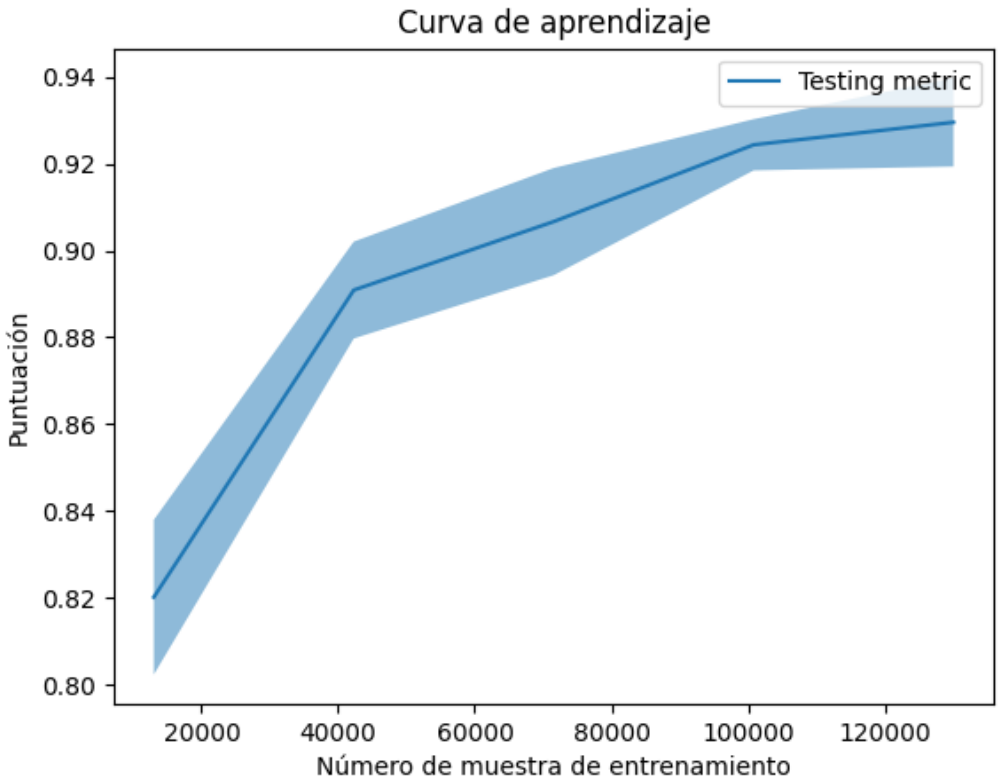
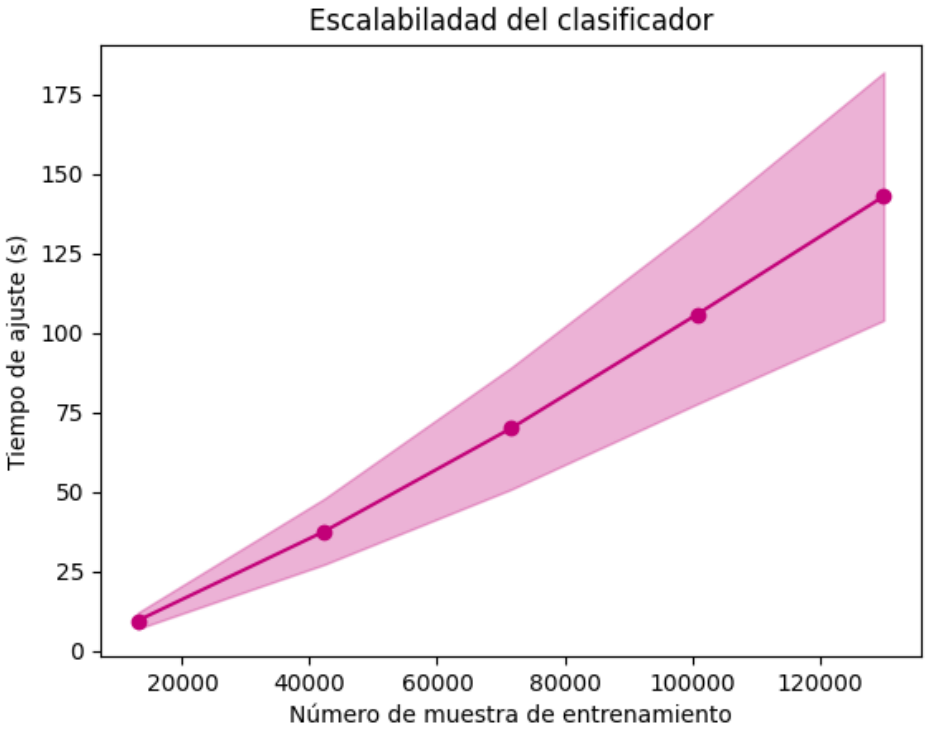
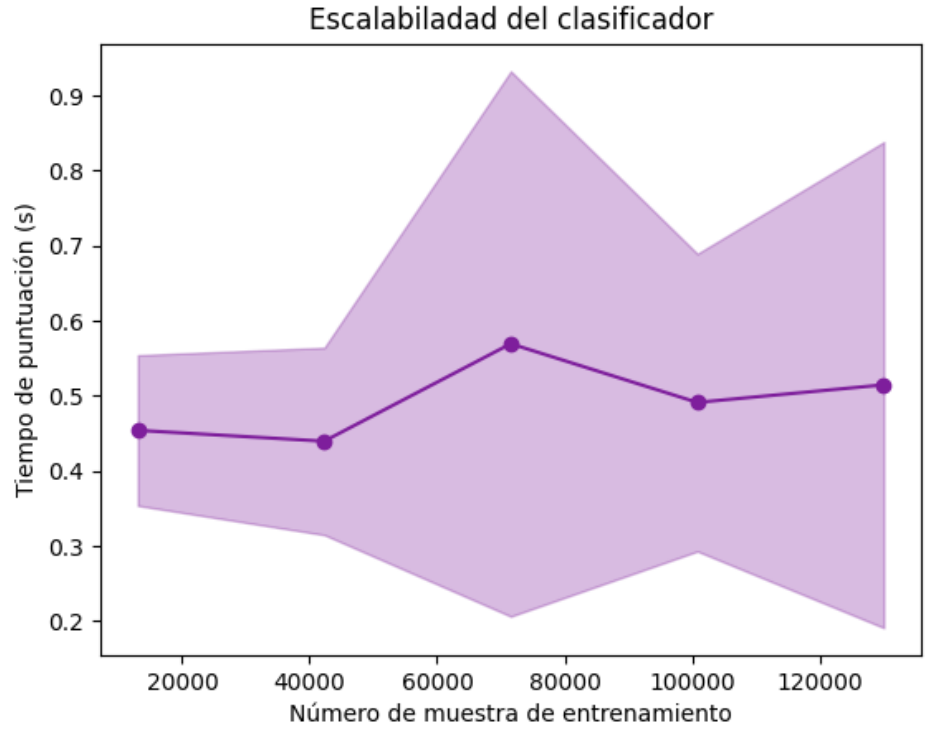


Figura 4.7 Curva de aprendizaje de todas las características para el Árbol clasificador.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.8 Análisis de complejidad del Árbol clasificador para todas las características.

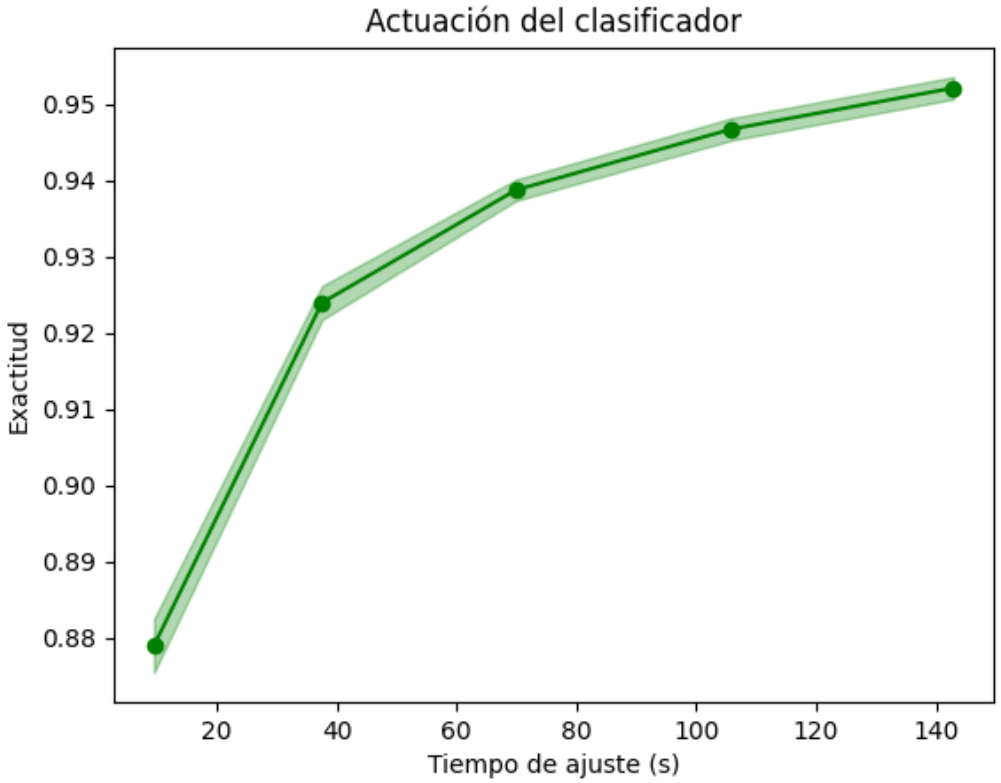


Figura 4.9 Actuación del Árbol clasificador para todas las características.

Comparación de los Árboles de Decisión clasificadores

Todos los árboles clasificadores realizan buenas predicciones. En general, al escalar los datos se obtienen peores estimaciones. La combinación de los parámetros obtiene más fallos que los árboles predeterminados por la librería, ya que estos vienen optimizados. Los mejores resultados se obtienen cuando se entre el Árbol clasificador simplemente con las características del artículo. Las buenas estimaciones van empeorando conforme se añaden columnas (características) en el entrenamiento. Aún así, el porcentaje de aciertos es superior al 90 % en todos los casos.

Gracias a las gráficas, se ve como todos los procesos de entrenamiento tienen el mismo comportamiento. La puntuación aumenta con el número de muestras, al igual que ocurre con el tiempo de ajuste. Este último es más extenso cuanto más características se tiene en consideración. El tiempo de puntuación se puede considerar constante para todos los subconjuntos de entrenamiento. La exactitud aumenta con el tiempo de ajuste. Obviamente, los tiempos aumentan cuando el modelo es entrenado con más características.

4.1.2 Regresión

Para el método de regresión no es necesario modificar los datos de entrada para el entrenamiento.

Características del artículo

El estudio de árbol regresor se comienza aplicando los estimadores vistos. En la Tabla 4.11 se ve como el modelo sin escalar es el que mejor resultados produce. El *MaxAbs Scaler* es el escalador con mayor número de aciertos, lo cual se debe a que éste transforma cada columna de característica de forma independiente, de esta manera la diferencia tan grande entre los valores

de los distintos atributos no afecta al proceso de entrenamiento de forma tan notoria como para los otros escaladores.

Tabla 4.11 Aciertos y fallos de los escaladores para las características del artículo del Árbol de regresión.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Sin Escalar	94,16	5,84	23053	1431
Standard Scaler	23,25	76,75	5693	18791
MaxAbs Scaler	44,24	55,76	10832	13652
MinMax Scaler (-10,10)	21,72	78,28	5318	19166

Acto seguido, se estudia como el cambio de un parámetro interno (3.5.1) del modelo de regresión afecta. En la Tabla 4.12, se tiene que todos los *criterion* obtienen muy buenos resultados. La estrategia aleatoria utilizada para elegir la división (*splitter*) tiene mayor porcentaje de fallos. Para *max_features*, se obtiene más aciertos cuando se consideran todas las características para el entrenamiento.

Tabla 4.12 Resultados del cambio de los parámetros de la regresión del Árbol de Decisión para las características del artículo.

		Porcentaje		Contador	
Iteracion		Acierto	Fallo	Acierto	Fallo
Criterion	squared error	98,14	1,86	24029	455
	friedman MSE	98,08	1,92	24014	470
	absolute error	97,93	2,07	23976	508
Splitter	best	98,08	1,92	24015	469
	random	83,56	1,86	20458	456
Max Features	auto	98,16	1,86	24034	456
	sqrt	76,98	1,86	18848	456
	log2	80,65	1,86	19747	456

Código 4.4 "Mejores parámetros" del Árbol de Decisión de regresión para las características del artículo.

```
>>> regresor_opt = DecisionTreeRegressor(
                          criterion = "squared_error",
                          splitter = "best",
                          max_features = "sqrt")
```

Ahora se entrena el modelo para las siguientes iteraciones: el Árbol de regresión predeterminado por la librería, el árbol predeterminado habiendo escaldo los datos previamente y aplicar los "mejores parámetros" para los casos anteriores (Tabla 4.13). Los resultados en porcentaje del árbol predeterminado y el de los "mejores parámetros" son básicamente iguales, la diferencia se debe a que el último acierta en 10 ocasiones más. En las iteraciones donde los datos están escalados, ambos tienen prácticamente el mismo numero de fallos y el porcentaje de resultados predecidos iguales a los verdaderos es aceptable.

Tabla 4.13 Aciertos y fallos de las distintas iteraciones para las características del artículo del Árbol de regresión.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Predeterminado	98,02	1,98	24000	484
MaxAbs Scaler	44,24	55,76	10832	13652
Mejores Param Sin Escalar	98,06	1,94	24010	474
Mejores Param Escalado	44,31	55,69	10848	13636

Se puede ver la calidad de cada iteración a través de los parámetros característicos de la regresión lineal (Tabla 4.14). Los valores más altos de R^2 se corresponden con los casos con mayor número de aciertos.

Tabla 4.14 Parámetros de la regresión de las distintas iteraciones del Árbol de regresión para las características del artículo.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	0,99968	0,0550	0,0313
MaxAbs Scaler	-0,36834	0,2616	0,1474
Mejores Param Sin Escalar	0,99968	0,0548	0,0315
Mejores Param Escalado	-0,36735	0,2615	0,1473

En la curva de aprendizaje (Figura 4.10) se puede considerar que la puntuación se comienza a establecerse a partir de las 40000 muestras. En realidad, la diferencia entre los valores de puntuación está en la cuarta cifra significativa por lo que se puede concluir que el árbol regresor obtiene buenos resultados sin hacerle falta más muestras de las dadas en el conjunto inicial de entrenamiento.

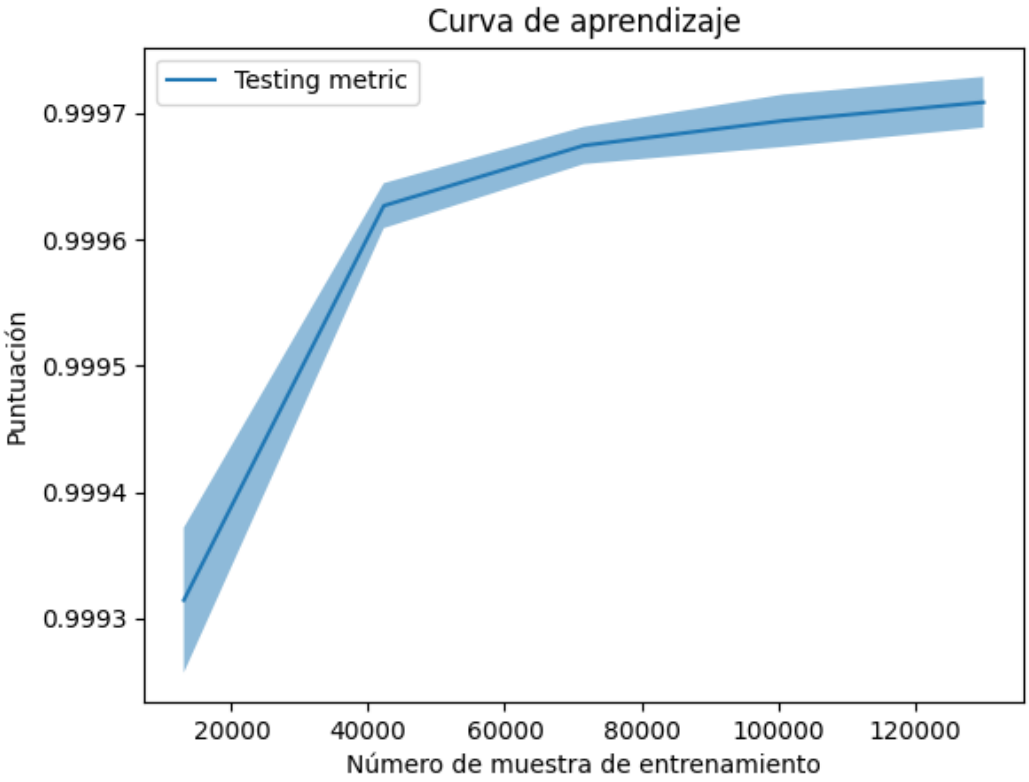
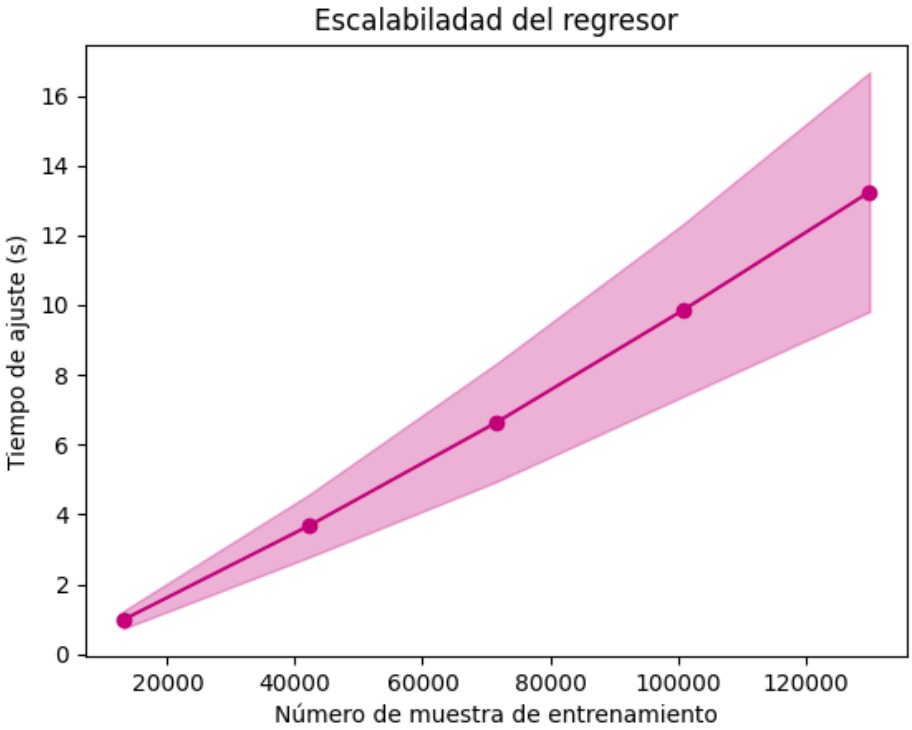
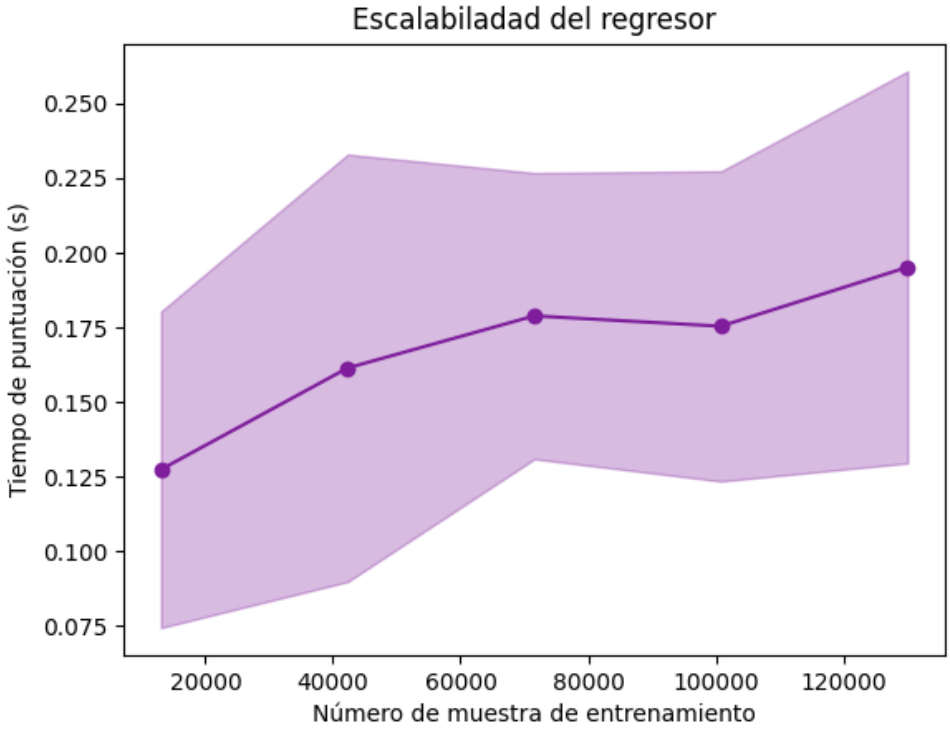


Figura 4.10 Curva de aprendizaje de las características del artículo para el Árbol de regresión.

Al realizar el análisis de complejidad (Figura 4.11), se identifica que el tiempo de ajuste es menor de medio minuto y el de puntuación menor de 1 segundo. Además, se ve como el tiempo de ajuste aumenta de forma considerable conforme se tienen más muestras. En cambio, el tiempo de puntuación aumenta pero de forma menos significativa, tanto que se puede considerar estable durante todo el entrenamiento.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.11 Análisis de complejidad del Árbol de regresión para las columnas del artículo.

El árbol mejora su actuación cuando el tiempo de ajuste es mayor,yaa que tiene más tiempo para adaptarse y realizar las decisiones acertadas. Pese a esto, el cambio es para la cuarta cifra significativa por lo que se puede considerar que la exactitud del regresor es constante.

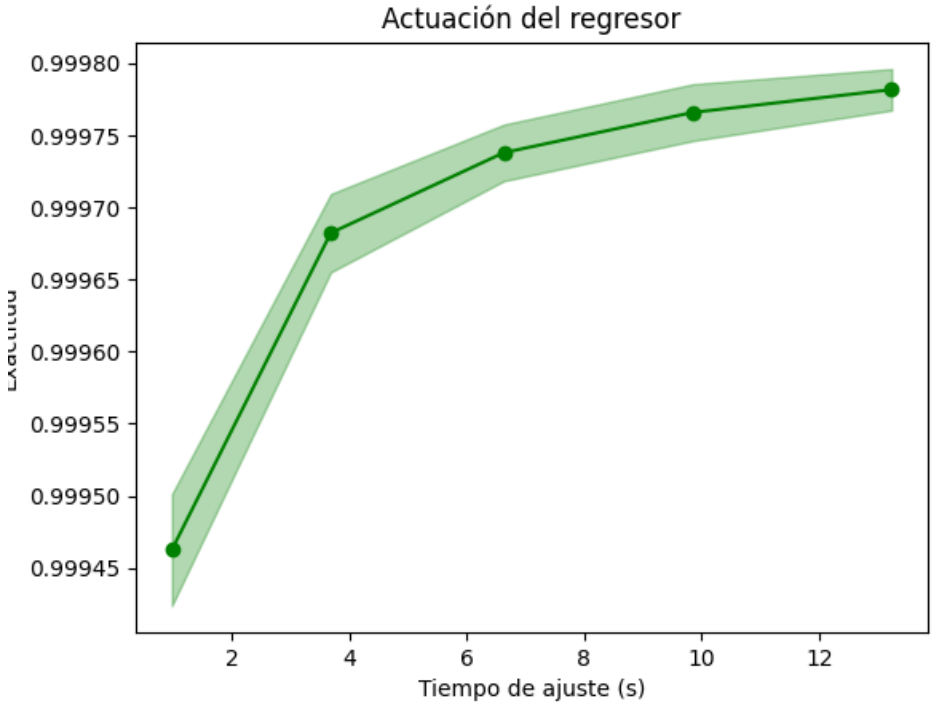


Figura 4.12 Actuación del Árbol de regresión para las características del artículo.

Características de la tabla del artículo

Se comienza aplicando los escaladores. El proceso de estandarización y de escalar entre un rango de números dan las peores estimaciones. El *MaxAbs Scaler* es el preprocesador que mayor numero de aciertos tiene, pero tampoco es un valor de porcentaje de aciertos muy altos. Al no escalar los datos, se alcanzan menos fallos.

Tabla 4.15 Aciertos y fallos de los escaladores del Árbol de regresión para todas las características de la tabla del artículo.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Sin Escalar	93,96	6,04	23006	1478
Standard Scaler	23,86	76,14	5843	18641
MaxAbs Scaler	43,90	56,10	10749	13735
MinMax Scaler (-10,10)	21,54	78,46	3613	20871

Para los parámetros internos del árbol (Tabla 4.16), los *criterion squared_error* y $friedman\_MSE$ tienen porcentajes aproximadamente iguales, la diferencia está en que el primero falla 8 veces más. $splitter = random$ produce peores resultados, por la estrategia aleatoria que sigue para elegir la división. En cuanto a la elección de los atributos, las opciones $log2$ y *sqrt* dan porcentajes parecidos no muy buenos. En cambio, $max\_features = auto$ produce un valor

alto de aciertos.

Tabla 4.16 Resultados del cambio de los parámetros de la regresión del Árbol de Decisión para las características de la tabla del artículo.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Criterion	squared error	98,26	1,74	24057	427
	friedman MSE	98,29	1,71	24065	419
Splitter	best	98,19	1,81	24065	442
	random	80,15	19,85	19624	4860
Max Features	auto	98,28	1,72	24063	421
	sqrt	65,77	34,23	16103	8381
	log2	62,76	37,24	15365	9119

Con esto, se pude decir que los "mejores parámetros" para las características de la tabla del artículo son:

Código 4.5 "Mejores parámetros" del Árbol de Decisión de regresión para las características de la tabla del artículo.

```
>>> regresor_opt = DecisionTreeRegressor(
                          criterion = "friedmas_mse",
                          splitter = "best",
                          max_features = "auto")
```

Una vez realizado el estudio de los parámetros internos, se estudian distintos casos para el Árbol de regresión al trabajar con las características de la tabla del artículo (Tabla 4.17). El modelo predeterminado y el modelo entrenado al aplicar los "mejores parámetros" obtienen resultados muy parecidos, siendo él último el que obtiene un mayor número de aciertos. Para las iteraciones en los que se ha escalado los datos previamente, el árbol predeterminado consigue menos fallos. Entre todos los casos, el de los mejores parámetros sin escalar es el que tiene el porcentaje verdadero superior.

Tabla 4.17 Aciertos y fallos de las distintas iteraciones para las características de la tabla del artículo del Árbol de regresión.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Predeterminado	98,20	1,80	24043	441
MaxAbs Scaler	43,90	56,10	10749	13735
Mejores Param Sin Escalar	98,21	1,79	24046	438
Mejores Param Escalado	43,88	56,12	10743	13741

En la Tabla 4.18 se exponen algunas de las características de la regresión. Se ve como el mayor error absoluto se corresponde con el peor de los resultados.

Tabla 4.18 Parámetros de la regresión de las distintas iteraciones del Árbol de regresión para todas las características de la tabla del artículo.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	0,9996	0,0565	0,0351
MaxAbs Scaler	-0,3690	0,2625	0,1475
Mejores Param Sin Escalar	0,9996	0,0561	0,0351
Mejores Param Escalado	-0,3709	0,2627	0,1477

En la curva de aprendizaje de la mejor de las opciones anterior (Figura 4.13), se ve como la puntuación del árbol va aumentando con las muestras de entrenamiento pero este cambio se produce en la cuarta cifra significativa del numero decimal, por lo que es un cambio muy pequeño. El análisis de complejidad (Figura 4.32) muestra como el tiempo de ajuste va aumentando de forma notable. Sin embargo, el tiempo de puntuación aumenta de forma casi imperceptible. Además, en la Figura 4.15 se observa como la exactitud aumenta cuando más tiempo de ajuste tiene el modelo.

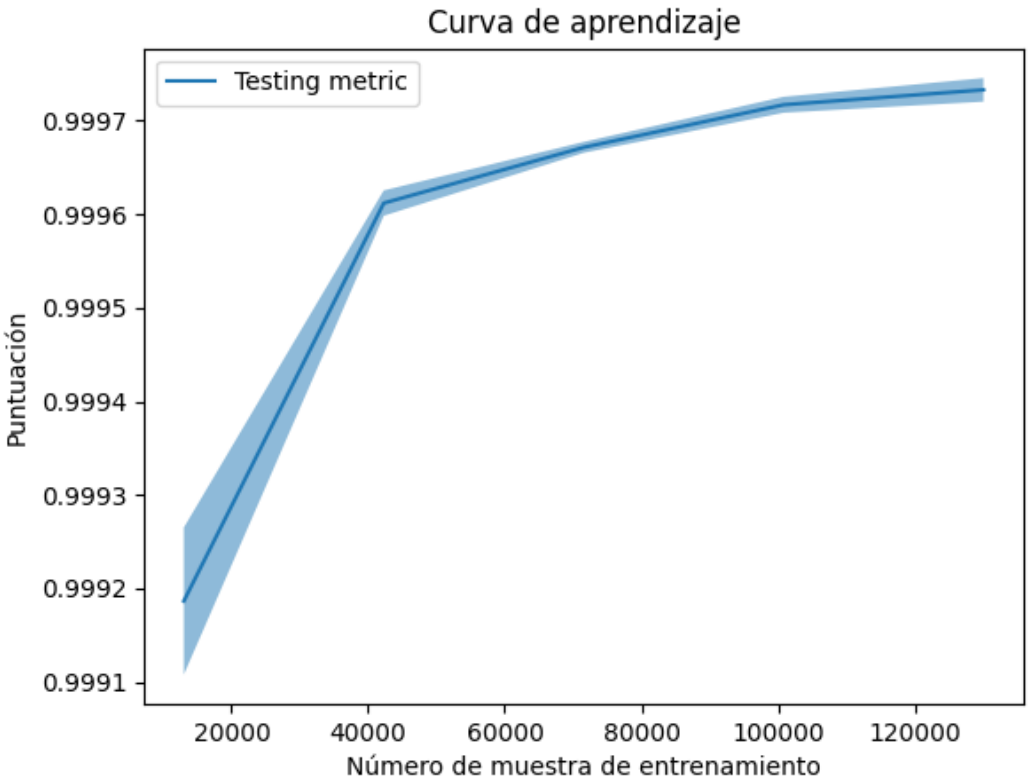
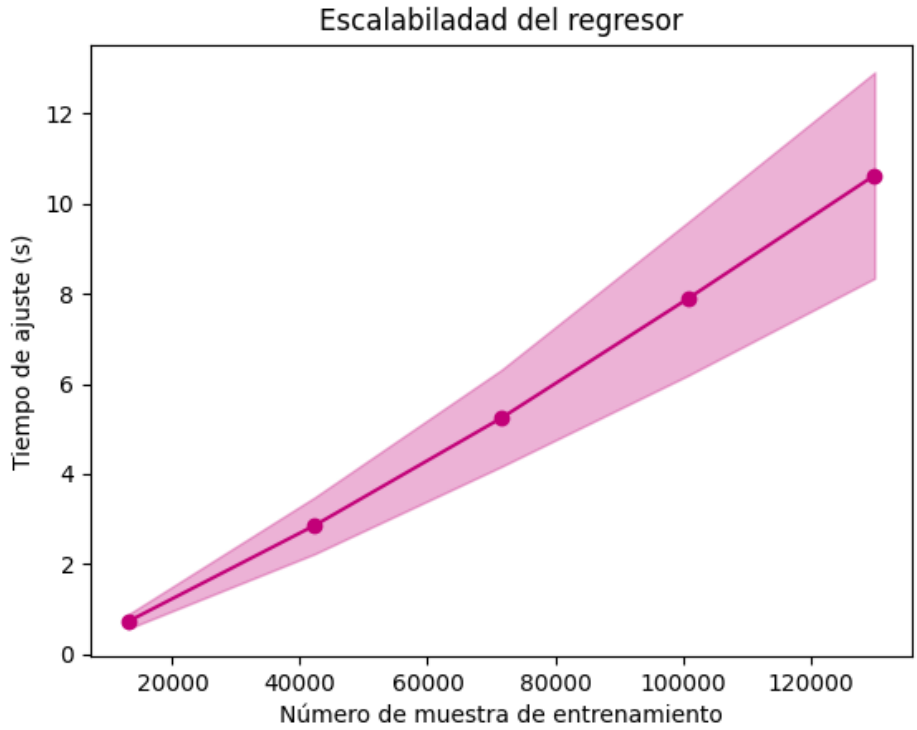
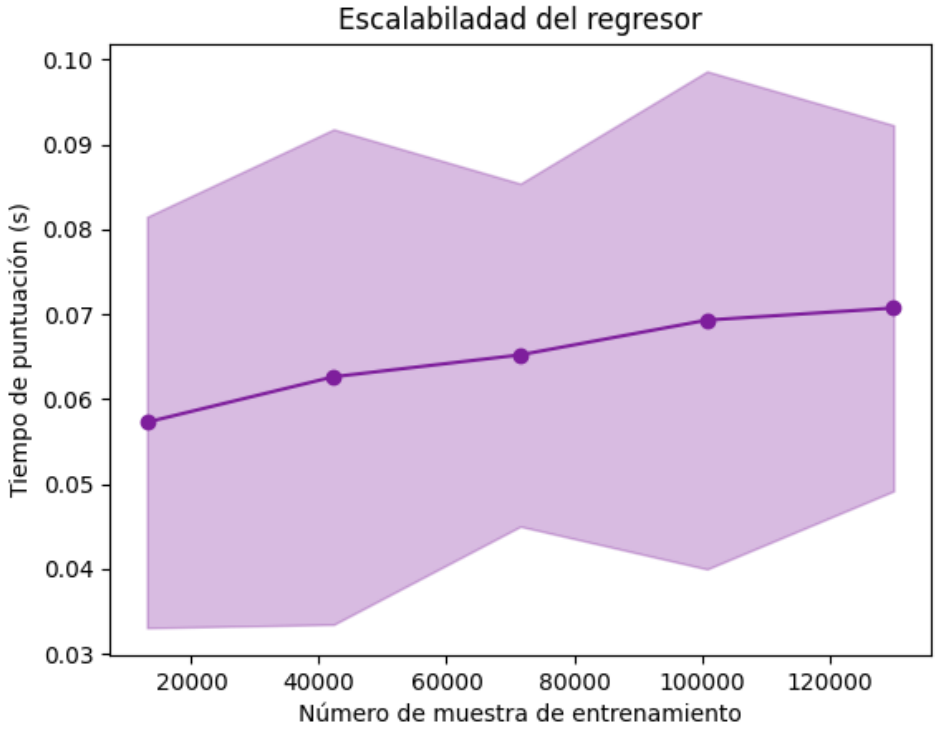


Figura 4.13 Curva de aprendizaje de las características de la tabla del artículo para el Árbol de regresión.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.14 Análisis de complejidad del Árbol de regresión para las columnas de la tabla del artículo.

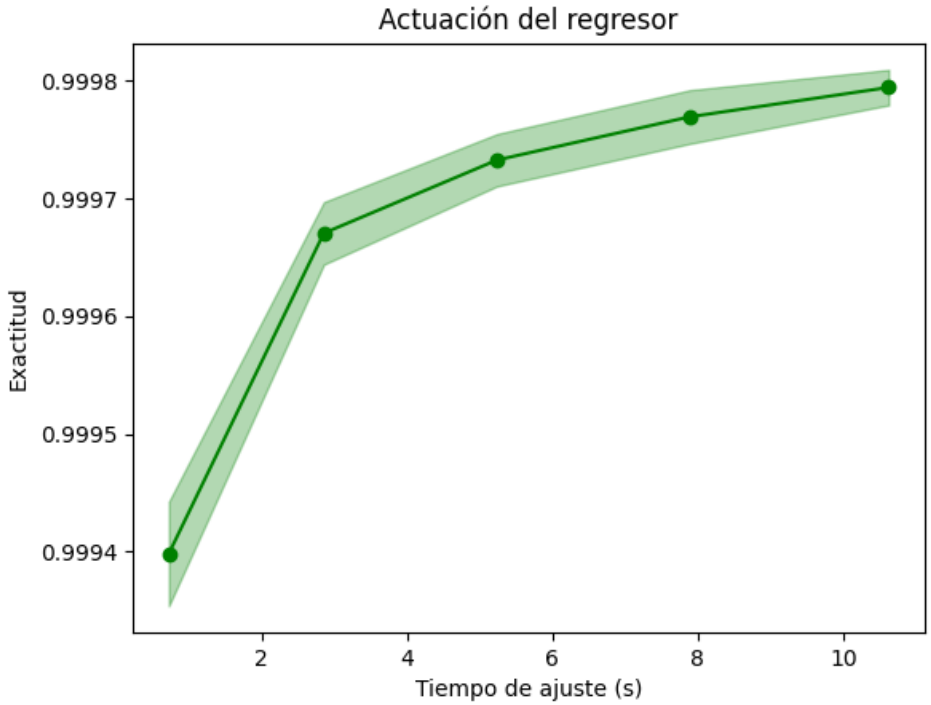


Figura 4.15 Actuación del Árbol de regresión para las características de la tabla del artículo.

Todas las características

Como se ve en la Tabla 4.19, los resultados para el *Standard Scaler* y el *MinMax Scaler* son los peores, hay más fallos que aciertos. el *MaxAbs Scaler* es el escalador con mejores resultados, por el mismo razonamiento que para los casos anteriores.

Tabla 4.19 Aciertos y fallos de los escaladores para todas las características del Árbol de regresión.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Sin Escalar	93,35	6,65	22856	1628
Standard Scaler	26,13	73,87	6399	18085
MaxAbs Scaler	45,21	54,79	11069	13415
MinMax Scaler (-10,10)	23,86	76,14	5841	18643

Para $criterion = absolute\_error$ el porcentaje verdadero es el menor, entre las otras dos posibilidades el porcentaje es lo mismo de bueno. La estrategia de división aleatoria es peor que la mejor. Para el número de atributos a considerar, con el parámetro "auto" se obtienen el mayor número de aciertos.

Tabla 4.20 Resultados del cambio de los parámetros de la regresión del Árbol de Decisión para todas las características.

Iteracion		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Criterion	squared error	93,19	6,81	22817	1667
	friedman MSE	93,14	6,86	22804	1680
	absolute error	79,62	20,38	19494	4990
Splitter	best	93,09	6,91	22792	1692
	random	63,21	36,79	15476	9008
Max Features	auto	93,01	6,99	22772	1712
	sqrt	43,55	56,45	10662	13822
	log2	38,10	61,90	9328	15156

Así, se puede considerar los mejores parámetros para el clasificador cuando se consideran todas las columnas de características como:

Código 4.6 "Mejores parámetros" del Árbol de regresión para todas las características.

```
>>> regresor_opt = DecisionTreeRegressor(
                          criterion = "squared_error",
                          splitter = "best",
                          max_features = "auto")
```

A continuación, se analizan los resultados del árbol regresor para los siguientes caso: predeterminado, escalado, "mejores parámetros" sin escalar, "mejores parámetros" escalados, 15 columnas como características y la elección por los pesos de las 15 mejores características. Cuando el modelo se entrena y valida considerando solo 15 columnas, los fallos aumentan. En este caso, *SelectKBest* obtiene que las columnas más importantes son: *relative_velocity_n*, *c_object_type*, *c_obs_used*, *c_position_covariance_det*, *t_sigma_r*, *c_sigma_r*, *t_sigma_t*, *c_sigma_t*, *t_sigma_n*, *c_sigma_n*,*t_sigma_rdot*,*c_sigma_rdot*, *t_sigma_tdot*, *c_sigma_tdot* y *t_sigma_ndot*. Entre las iteraciones del árbol predeterminado y el de los mejores parámetros, el porcentaje de aciertos es elevado y solo se diferencian en que el primero acierta en seis ocasiones más.

Tabla 4.21 Aciertos y fallos de las distintas iteraciones del Árbol de regresión para todas las características.

	Porcentaje		Contador	
	Verdadero	Falso	Verdadero	Falso
Predeterminado	98,26	1,74	24058	426
MaxAbs Scaler	45,21	54,79	11069	13415
Mejores Param Sin Escalar	98,24	1,76	24052	432
Mejores Param Escalado	45,92	54,08	11242	13242
Max features	48,58	51,42	11894	12590
Best K feature	25,65	74,35	6279	18205

Además, comparando los parámetros característicos de cada regresión se ve como para un R^2. El valor más alto de éste no coindice exactamente con el caso con el mayor porcentaje de

estimaciones verdaderas, pero por muy poco. La peor de las iteraciones tiene el valor del error absoluto más grande.

Tabla 4.22 Parámetros de la regresión de las distintas iteraciones del Árbol de regresión para todas las características.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	0,99966	0,0597	0,0329
MaxAbs Scaler	-0,22228	0,2422	0,1317
Mejores Param Sin Escalar	0,99970	0,0588	0,0293
Mejores Param Escalado	-0,22857	0,2430	0,1324
Max features (k=15)	0,876235499	1,6586	11,9997
Best K feature (k=15)	-0,182852466	7,2253	114,6847

Como en casos anteriores, la puntuación de la curva de aprendizaje (Figura 4.16)aumenta con el número de muestras. En un principio crece muy rápido para luego llegar a crecer muy poco a poco. El tiempo de ajuste (Figura 4.17) crece rápidamente, llegando a pasar los dos minutos, cuando el número de muestras aumenta. En cambio, el tiempo de puntuación no es muy grande y su variación es mínima. La actuación del regresor (Figura 4.18) va aumentando con el tiempo de ajuste, cuando se pasan los dos minutos la exactitud es casi del 100 %.

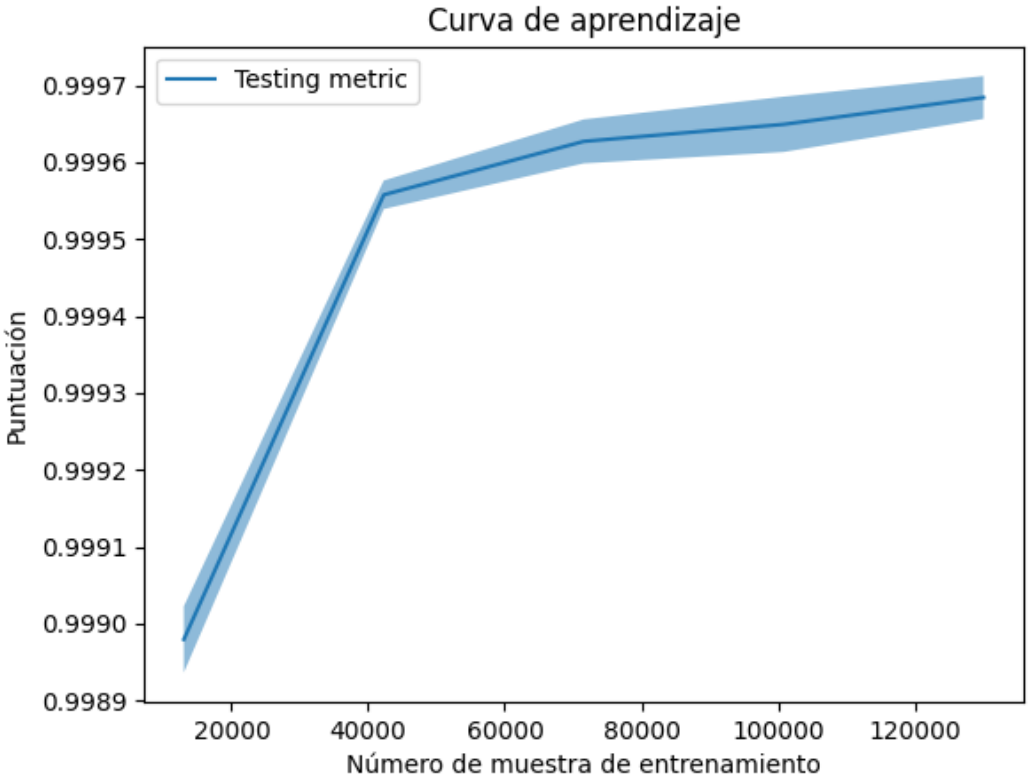
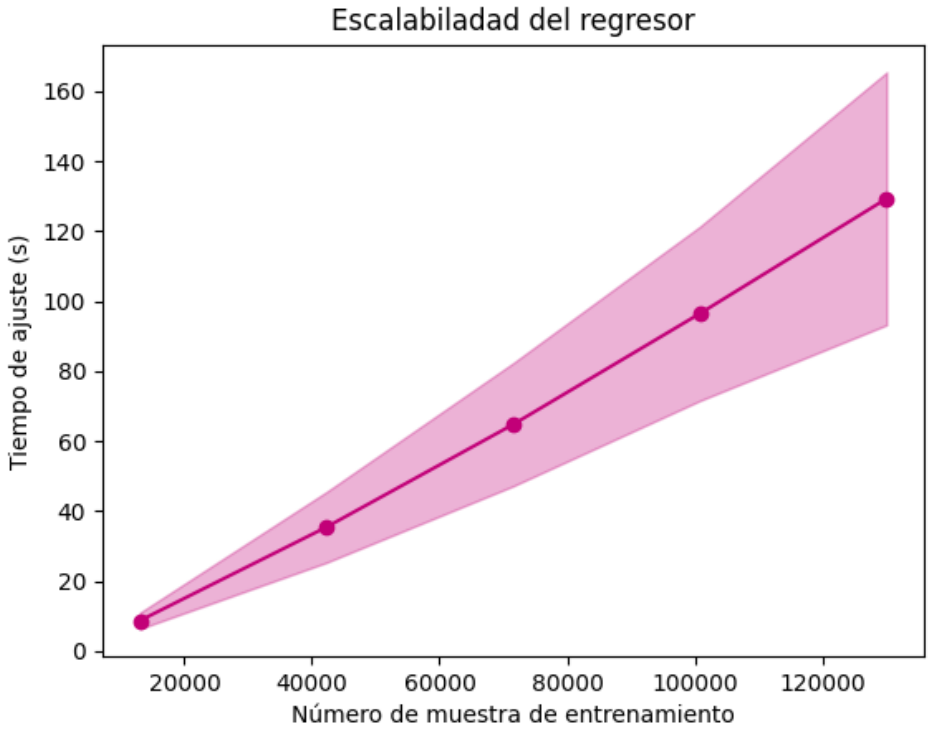
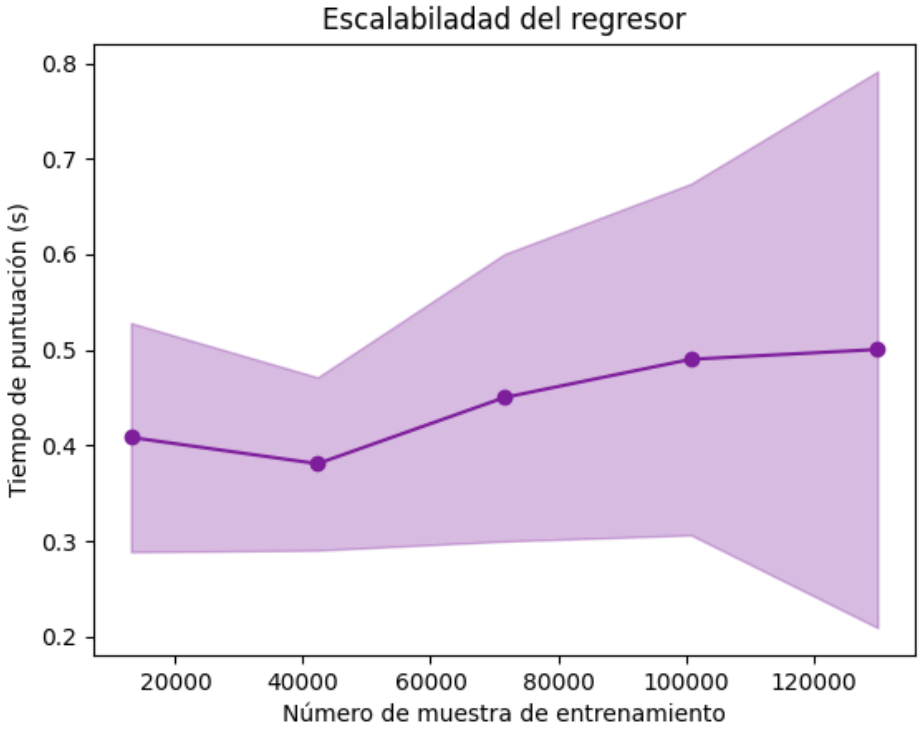


Figura 4.16 Curva de aprendizaje de todas las características para el Árbol de regresión.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.17 Análisis de complejidad del Árbol de regresión para todas las características.

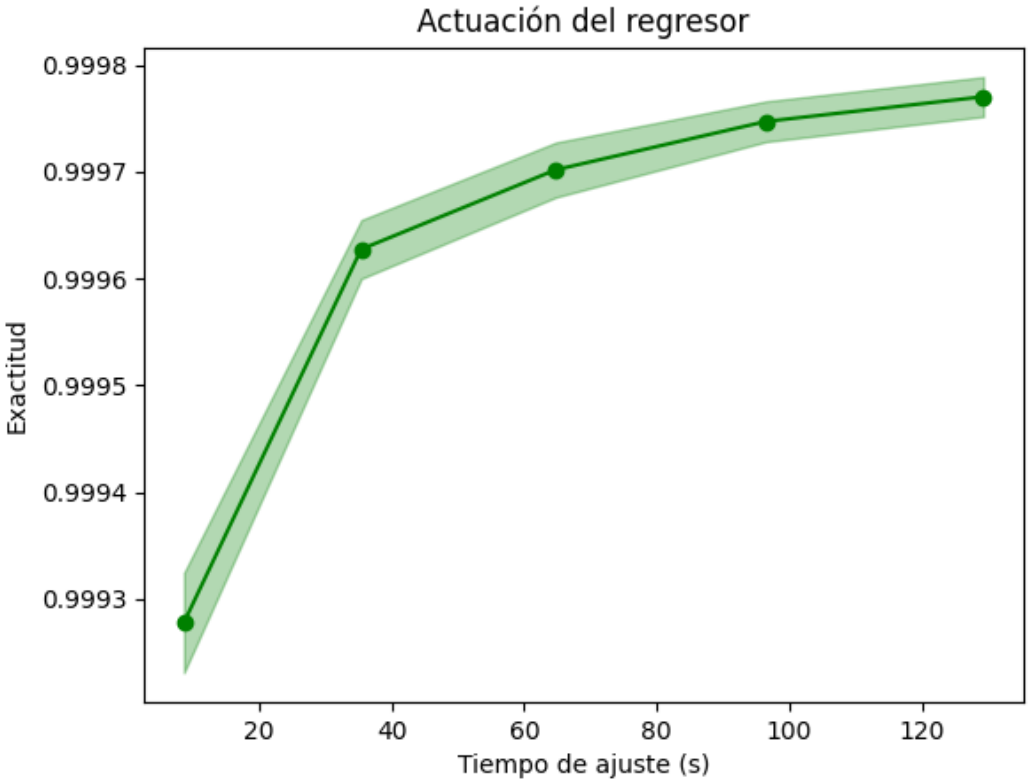


Figura 4.18 Actuación del Árbol de regresión para todas las características.

Comparación de los Árboles de Decisión de regresión

Para los árboles de regresión, los parámetros predeterminados consiguen muy buenos resultados, todos mayores del 98 %, al igual que la aplicación de los "mejores" parámetros. Al escalar los datos de forma previa, se obtienen unas predicciones acertadas en torno al 70 %. El mayor número de aciertos se da cuando se entrena el modelo con todas las características, pero solo se diferencia de 6 fallos frente al entrenamiento con las columnas de la tabla del artículo.

En general, se pueden considerar que los distintos árboles de regresión consiguen resultados aceptables, excepto cuando se aplica algún cambio que afecta a las características (columnas) como reducir el número de *max_features* a considerar o aplicar antes del entrenamiento *SelectKBets* o se escalan datos.

Las curvas dibujadas muestran lo mismo que en los clasificadores. Los modelos han sido bien entrenados y van aumentando su exactitud conforme lo hace el tiempo de ajuste y el número de muestras.

4.2 Redes Neuronales

Las Redes Neuronales pueden trabajar con valores del tipo *float64*, por lo que no es necesario ningún procesamiento previo de éstos.

4.2.1 Clasificación

Al igual que ocurre con el clasificador del Árbol de Decisión, también se debe transformar la columna a clasificar, se realiza según la Tabla 4.1.

Características del artículo

Se comienza con el estudio de los escaladores, tal y como en los árboles clasificadores. Los resultados del *Standard Scaler* y del *MaxAbs Scaler* son parecidos, siendo el último ligeramente superior. El *MinMax Scaler* produce pocos aciertos, pero la red entrenada con los datos sin escalar tiene el mayor número de fallos.

Tabla 4.23 Aciertos y fallos de los escaladores de la Red clasificadora para las características del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	7,72	92,28	1890	22594
Standard Scaler	70,13	29,87	17171	7313
MaxAbs Scaler	78,46	21,54	19211	5273
MinMax Scaler (-10,10)	37,03	62,97	9067	15417

A continuación, se pasa al análisis de los cambios en los resultados que produce modificar algunos de los parámetros internos de la Red clasificadora. En general, cambiar un parámetro interno de la red y dejar los demás de forma predeterminada, no produce buenos resultados. Para los *solver*, el mejor de los casos es el *lbfgs* y su porcentaje de aciertos no es alto. Respecto a la función de activación, la función tangente hiperbólica y la logística obtienen el mismo numero de aciertos y fallos. Para la tasa de actualización de los pesos (*learning_rate*), la de decrecimiento de la tasa (*invscaling*) acierta en más ocasiones.

Tabla 4.24 Resultados del cambio de los parámetros de la Red clasificadora para las características del artículo.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	0	100	0	24484
	sgd	0	100	0	24484
	lbfgs	0,76	99,24	185	24299
Activation	identity	0	100	0	24484
	logistic	28,12	71,88	6886	17598
	tanh	28,12	71,88	6886	17598
	relu	0	100	0	24484
Learning Rate	constant	0	100	0	24484
	adaptive	0	100	0	24484
	invscaling	28,12	71,88	6884	17600
Hidden Layers	500	0	100	0	24484
	900	0	100	0	24484

En la combinación de los parámetros al no establecer el *solver* como "sgd", la tasa de actualización (*learning_rate*) se descarta.

Código 4.7 "Mejores parámetros" de la Red Neuronal clasificadora para las columnas del artículo.

```
>>> clasificador_opt = MLPClasssifier(
                          solver = "lbfgs",
                          activation = "logistic")
```

Una vez hecho este análisis previo, se entrena, puntúa y compara el clasificador para diversos casos, mostrados en la Tabla 4.24. En ella, se ve como los casos en los que los datos son escalados es cunado se producen los mejores resultados. Además, cuando se entrena el clasificador con la combinación de los mejores parámetros, se obtienen el mayor número de aciertos. La aplicación del análisis de componente principales no obtiene buenas estimaciones, lo que tiene sentido, pues se han reducido el número de característica a las consideradas más relevantes por la ESA.

Tabla 4.25 Aciertos y fallos de las distintas iteraciones para las características del artículo de la Red clasificadora.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Predeterminado	20,93	79,07	5124	19360
MaxAbs Scaler	77,22	22,78	18907	5577
Mejores Param Sin Escalar	28,12	71,88	6884	17600
Mejores Param Escalado	79,16	20,84	19381	5103
PCA	3,72	96,28	912	23572
Mejores Param PCA	28,97	71,03	7094	17390

A continuación, se dibujan la curva de aprendizaje y los análisis de complejidad y de actuación de la mejor Red clasificadora anterior. En la Figura 4.19, se ve como la puntuación del clasificador se estabiliza cuando se está en un rango entre 0.8 y 0.82, lo que tiene sentido pues el porcentaje de aciertos es más o menos de esos valores. En el análisis de complejidad (Figura 4.20), se observa como el tiempo de ajuste del clasificador aumenta con el numero de muestras. Además, este tiempo sobrepasa la marca de los dos minutos. El tiempo de puntuación varía dependiendo del número de muestras, de forma más notable que en otras ocasiones. Respecto a la actuación del clasificador (Figura 4.21), la exactitud frente al tiempo de ajuste no varía demasiado. Pese al aumentar el tiempo de ajuste del entrenamiento, la exactitud varía en en un rango de +/- 0,001.

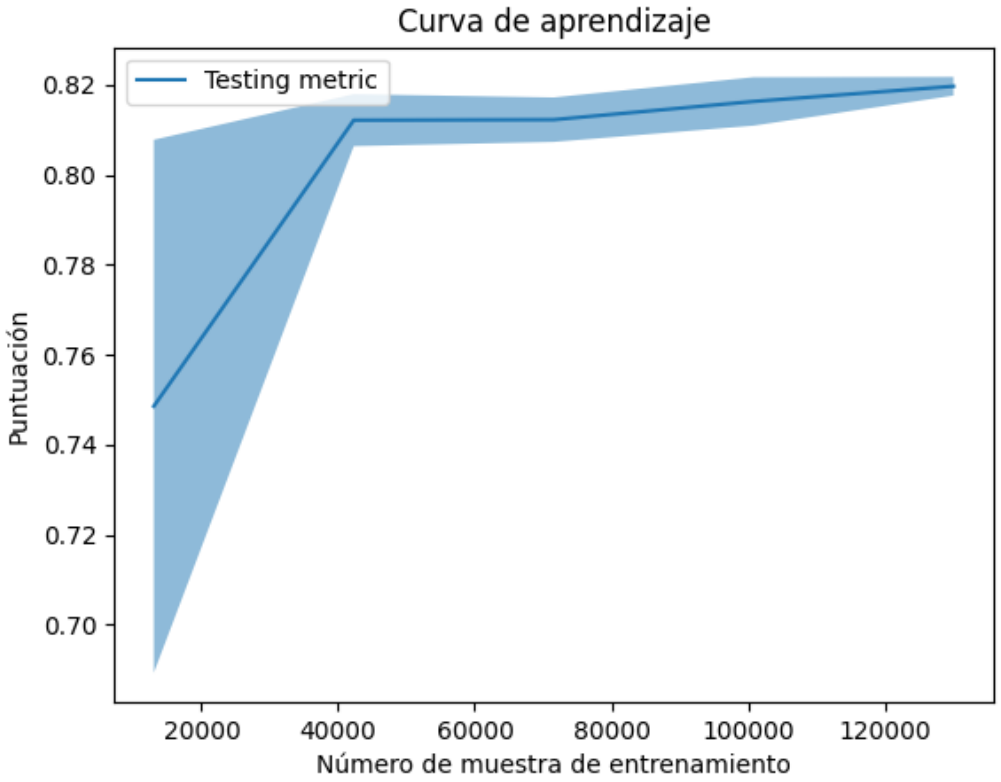
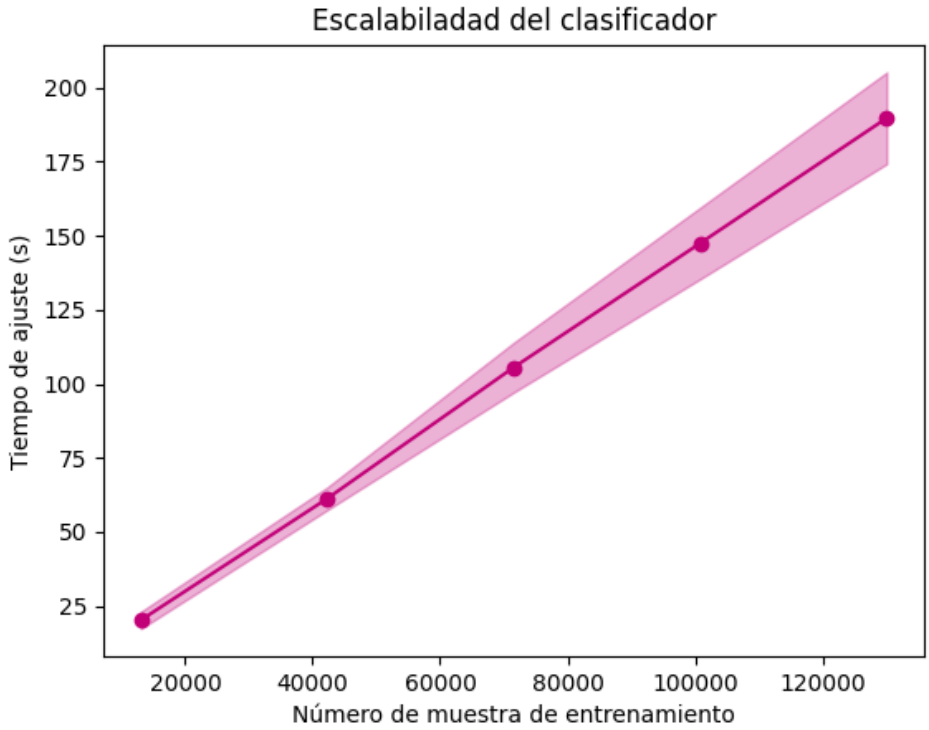
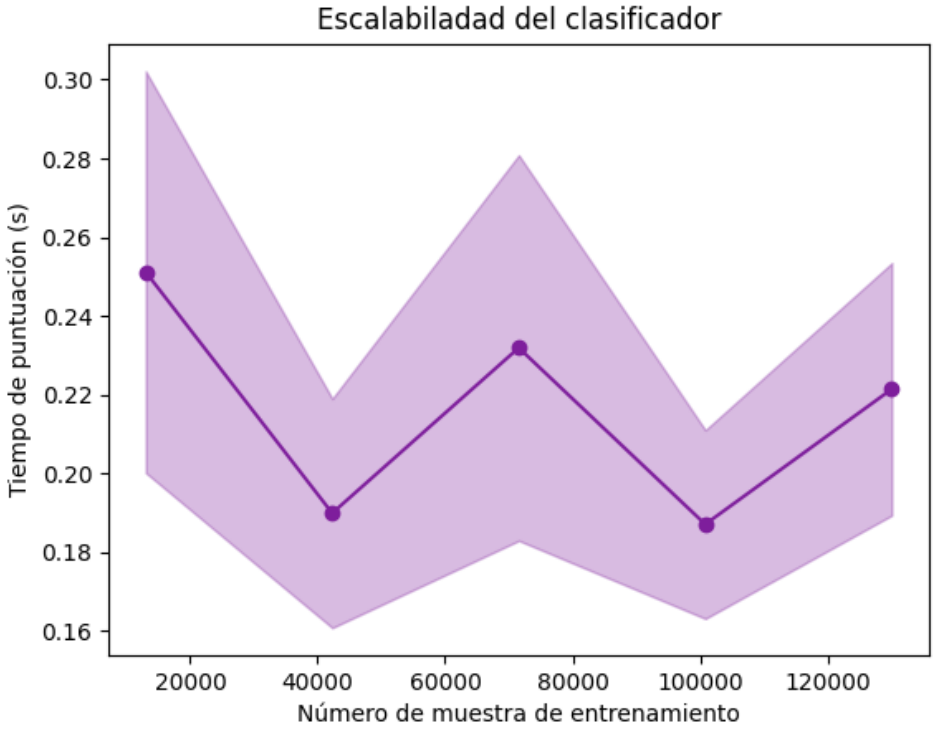


Figura 4.19 Curva de aprendizaje de las características del artículo para la Red clasificadora.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.20 Análisis de complejidad de la Red clasificadora para las columnas del artículo.

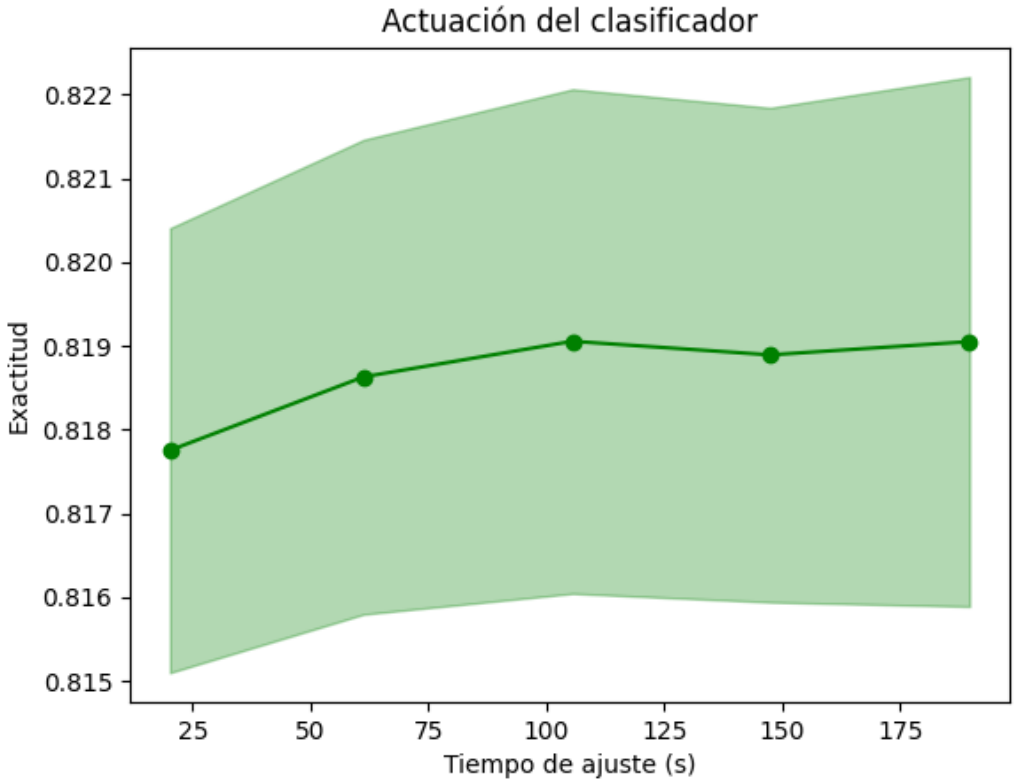


Figura 4.21 Actuación de la Red clasificadora para las características del artículo.

Características de la tabla del artículo

Al aplicar los distintos escaladores se obtienen los aciertos y fallos mostrados en Tabla 4.26. El mayor número de fallos viene dado por los datos sin escalar. Entre el proceso estandarización y el de escalar en un rango no existe una diferencia muy grande en el número de aciertos. El *MaxAbs Scaler* obtiene el mayor porcentaje de resultados verdaderos, debido a la forma de escalar cada característica (columna) de forma independiente.

Tabla 4.26 Aciertos y fallos de los escaladores de la Red clasificadora para las características de la del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	18,04	81,96	4417	20067
StandardScaler	36,30	63,70	8888	15596
MaxAbs Scaler	84,25	15,75	20627	3857
MinMax Scaler (-10,10)	30,22	69,78	7399	17085

Los cambios en los parámetros de la Red clasificadora al entrenarla con las características de la tabla del artículo no obtienen buenas estimaciones a la hora de validar. Los *solver adam* y *sgd* tienen 0 aciertos. Tanto la función hiperbólica como la sigmoide aciertan casi el mismo número de veces. La tasa de actualización *invscaling* es la que mejores resultados produce.

Tabla 4.27 Resultados del cambio de los parámetros de la Red clasificadora para las características de la tabla del artículo.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	0	100	0	24484
	sgd	0	100	0	24484
	lbfgs	3,09	96,91	756	23728
Activation	identity	0	100	0	24484
	logistic	28,10	71,90	6881	17603
	tanh	28,12	71,88	6886	17598
	relu	0	100	0	24484
Learning Rate	constant	0	100	0	24484
	adaptive	0	100	0	24484
	invscaling	28,12	71,88	6884	17600
Hidden Layers	500	0	100	0	24484
	900	0	100	0	24484

Así, la combinación de los mejores parámetros de la Red clasificadora para entrenarla con las características de la tabla del artículo, donde se ha descartado la tasa de actualización por la misma razón que en el caso anterior, es:

Código 4.8 "Mejores parámetros" de la Red Neuronal clasificadora para las columnas de la tabla del artículo.

```
>>> clasificador_opt = MLPClasssifier(
                          solver = "lbfgs",
                          activation = "tanh")
```

Una vez realizado estos análisis previos, se pasa al estudio de distintos casos, cambiando algún parámetro de la red y escalando (Tabla 4.28). Las mejores predicciones se obtienen al escalar los datos y entrenar la red con los parámetros predeterminados. Cuando se trabaja con los datos originales, no se tiene un alto número de aciertos, lo que se debe a que las Redes Neuronales de scikit no están preparadas para ello. El análisis de componentes principales con los datos sin escalar, consigue más fallos que la Red clasificadora predeterminada. En cambio, si se entrena la red con la combinación de los mejores parámetros, los aciertos aumentan de forma significativa.

Tabla 4.28 Aciertos y fallos de las distintas iteraciones de la Red clasificadora para las características de la tabla del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Predeterminado	7,34	92,66	1797	22687
MaxAbs Scaler	83,92	16,08	20548	3936
Mejores Param Sin Escalar	28,08	71,92	6875	17609
Mejores Param Escalado	78,31	21,69	19173	5311
PCA	1,87	98,13	457	24027
Mejores Param PCA	28,97	71,03	7094	17390

Observando la curva de aprendizaje (Figura 4.22), la puntuación del clasificador va aumentando conforme aumenta el número de muestras y seguiría creciendo si se entrenará con más muestras. El análisis de complejidad (Figura 4.23) muestra el tiempo de ajuste frente al número de muestras de entrenamiento. Este tiempo va aumentando de forma muy considerable, alcanzando los ochos minutos para el mayor número de muestras. En cambio el tiempo de puntuación se estabiliza alcanzado el segundo subconjunto de validación en el entrenamiento. La curva de actuación del clasificador (Figura 4.24) muestra como la exactitud de éste va aumentando con el tiempo de ajuste.

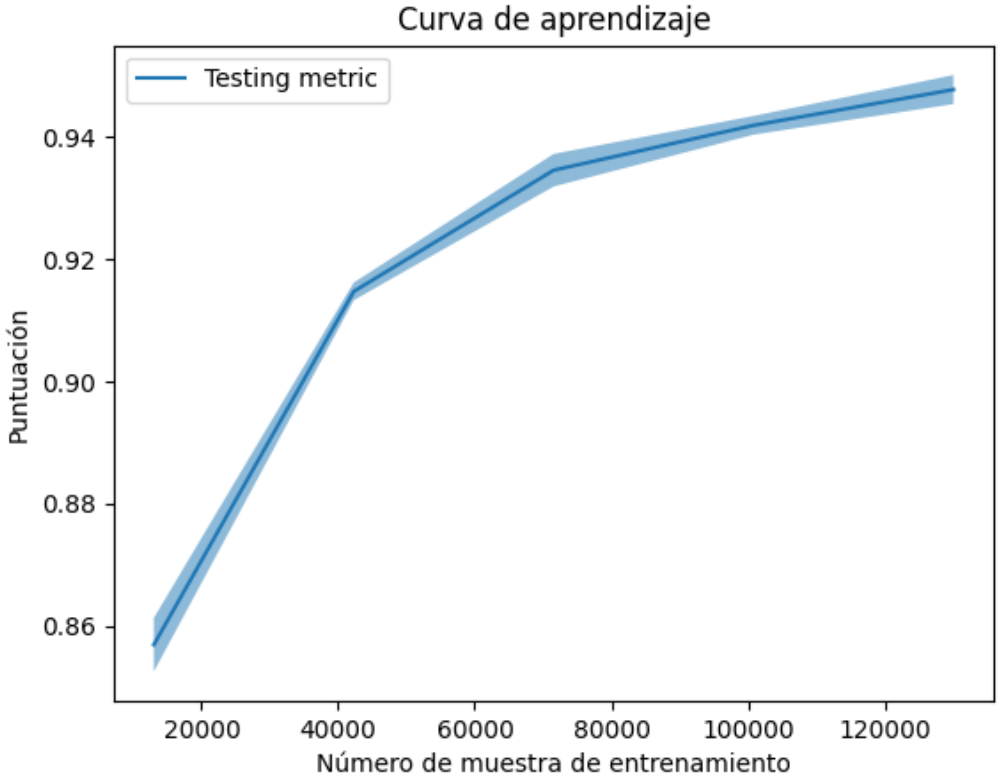
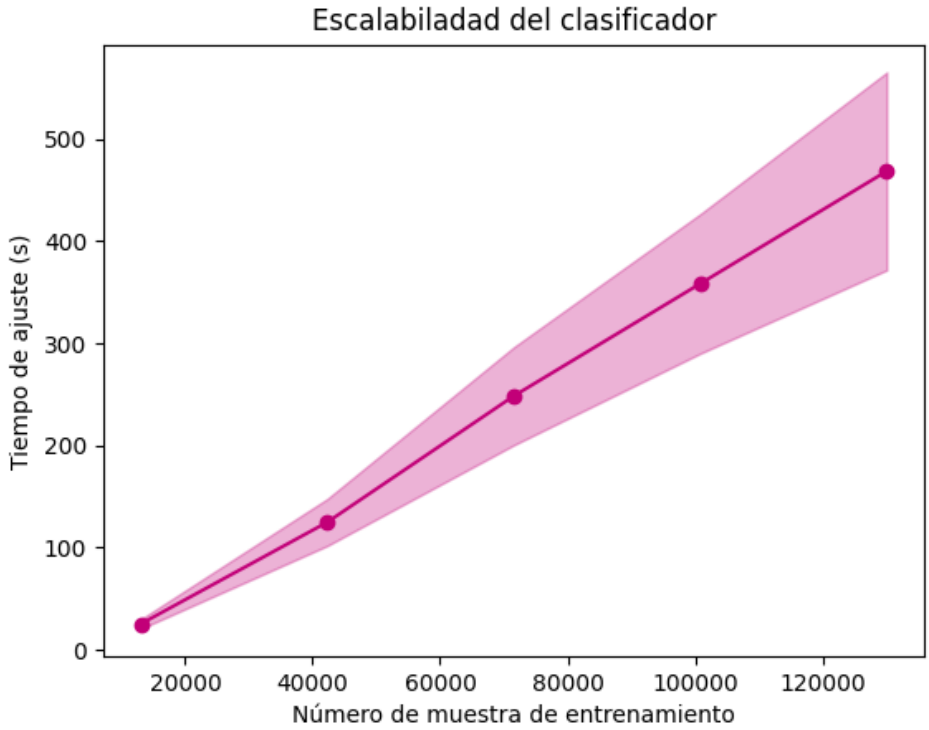
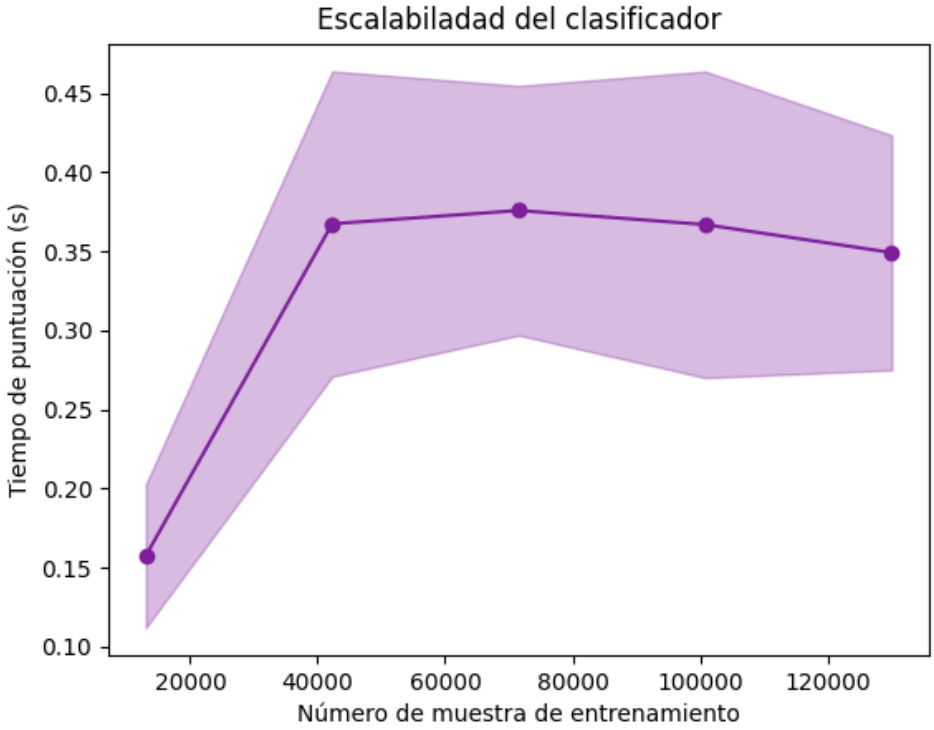


Figura 4.22 Curva de aprendizaje de las características de la tabla del artículo para la Red clasificadora.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.23 Análisis de complejidad de la Red clasificadora para las características de la tabla del artículo.

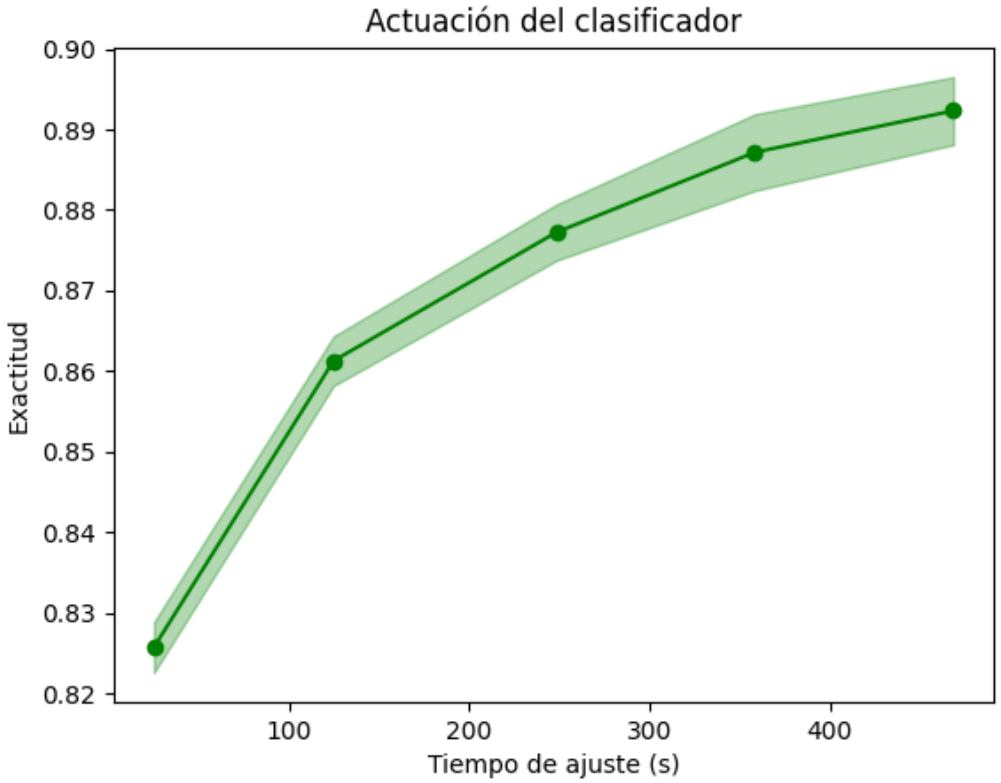


Figura 4.24 Actuación de la Red clasificadora para las características de la tabla del artículo.

Todas las características

Como para todos los casos estudiados, se empieza analizando las estimaciones de los preprocesadores escaladores. Se ve como afectan el hecho de escalar los datos a las redes (Tabla 4.29).

Tabla 4.29 Aciertos y fallos de los escaladores de la Red clasificadora para todas las características.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Sin Escalar	7,43	92,57	1820	22664
StandardScaler	73,07	26,93	17890	6594
MaxAbs Scaler	72,06	27,94	17644	6840
MinMax Scaler (-10,10)	31,41	68,59	7690	16794

A continuación, se completa el estudio de los parámetros internos de la red (Tabla 4.30). En este caso, el *solver adam* tiene el mayor número de aciertos, tal y como explica la librería scikit-learn, al tratarse de un conjunto de datos muy amplio. Para las funciones de activación, existe una mayor diferencia en resultados que para los casos anteriores. La tasa de actualización constantes y "adaptive" solo fallan y el añadir capas intermedias a la red no influye.

Tabla 4.30 Resultados del cambio de los parámetros de la Red clasificadora para todas las características.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	10,44	89,56	2556	21928
	sgd	33,59	66,41	8224	16260
	lbfgs	1,02	98,98	250	24234
Activation	identity	0	100	0	24484
	logistic	28,82	71,18	7057	17427
	tanh	29,28	70,72	7168	17316
	relu	0,02	99,98	6	24478
Learning Rate	constant	0	100	0	24484
	adaptive	0	100	0	24484
	invscaling	28,64	71,36	7011	17473
Hidden Layers	500	0	100	0	24484
	900	0	100	0	24484

Por tanto, la combinación de los mejores parámetros cuando se entrena con todas las columnas es:

Código 4.9 "Mejores parámetros" de la Red Neuronal clasificadora para todas las columnas.

```
>>> clasificador_opt = MLPClasssifier(
                        solver = "adam",
                        activation = "tanh")
```

Como se está analizando la red cuando se le da de entrada todas las características, se van a considerar dos iteraciones más que en los apartados anteriores: el análisis de las componentes principales y la selección de las características con mayor peso (Tabla 4.31). Los casos en los que se ha escalado los datos son los que dan el mayor número de aciertos. La red predeterminada y la red habiendo aplicado el análisis de componentes principales dan las peores estimaciones a la hora de validar el modelo. La aplicación de la red con la combinación de los mejores parámetros mejoran los aciertos frente a las redes predeterminadas. El porcentaje más alto de aciertos se dan para la red entrenada habiendo escalado los datos con el *Standard Scaler*.

Tabla 4.31 Aciertos y fallos de las distintas iteraciones de la Red clasificadora para todas las características.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Predeterminado	7,33	92,67	1795	22689
Standard Scaler	73,72	26,28	18049	6435
Mejores Param Sin Escalar	27,46	72,54	6724	17760
Mejores Param Escalado	70,34	29,66	17221	7263
PCA	3,33	96,67	815	23669
Mejores Param PCA	30,13	69,87	7378	17106
Best K feature (k=15)	4,84	4,84	1185	23299

El *SelectKBest* escoge las siguientes característica como las más importantes: *time_to_tca*, *mission_id*, *c_time_lastob_start* , *mahalanobis_distance*, *t_position_covariance_det*, *c_position_covariance_det*, *t_sigma_r*, *c_sigma_r*, *t_sigma_t*, *c_sigma_t*, *t_sigma_n*, *c_sigma_n*, *c_sigma_rdot*, *c_sigma_tdot* y *t_sigma_ndot*. Muchas de éstas coinciden con las del artículo o la tabla del artículo.

La curva de aprendizaje (Figura 4.25) muestra como la puntuación aumenta con el número de muestras y sigue una tendencia más o menos lineal. El análisis de complejidad (Figura 4.26) pone en evidencia como el tiempo de ajuste del modelo es extenso hasta para el menor número de muestras, el tiempo máximo alcanza los 11 minutos. El tiempo de puntuación es alto, para lo que es el rango normal de éste, pero es mas o menos estable a lo largo de las muestras. La actitud del clasificador (Figura 4.27) es mayor cuanto más grande es el tiempo de ajuste, lo que tiene sentido pues la red cuenta con más tiempo para entrenar y adaptarse de forma correcta a los datos.

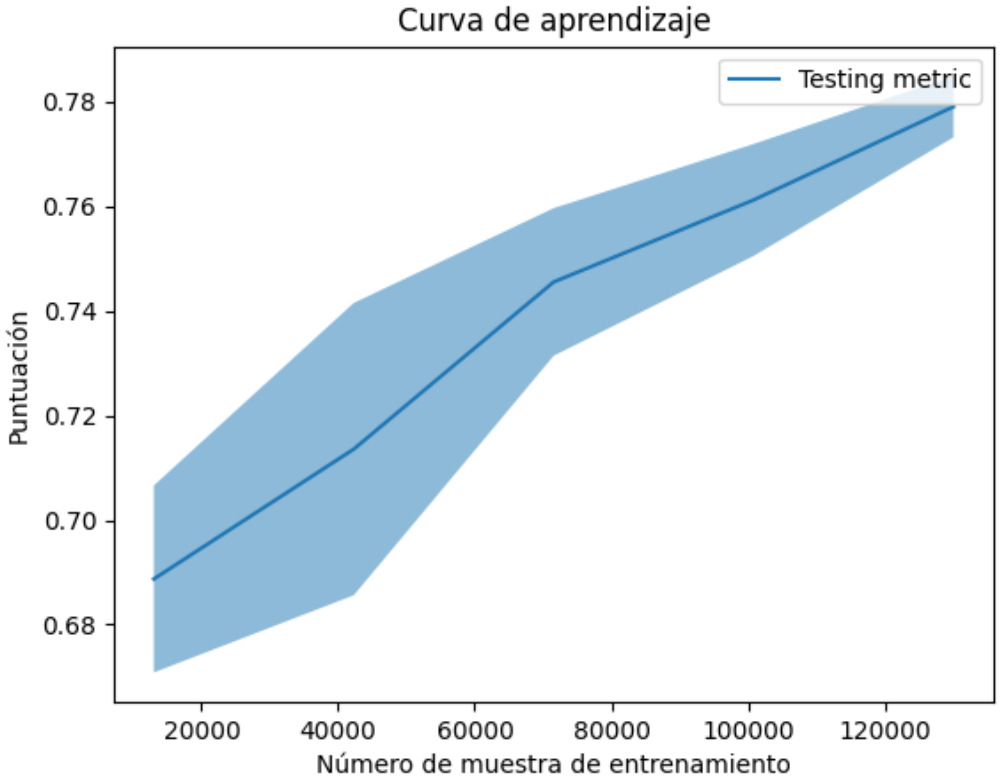
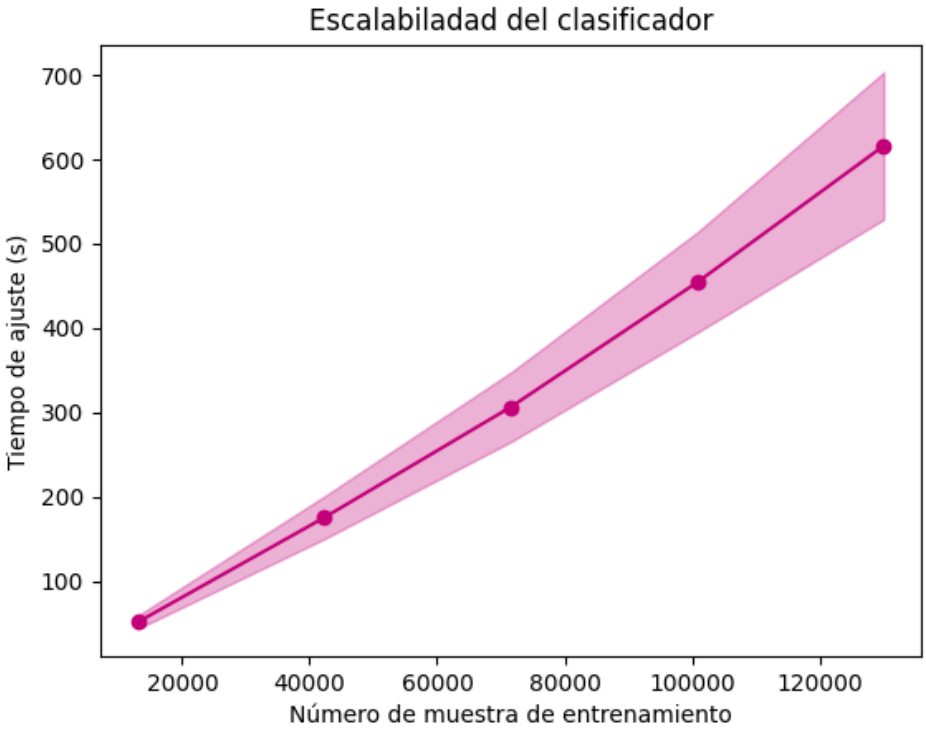
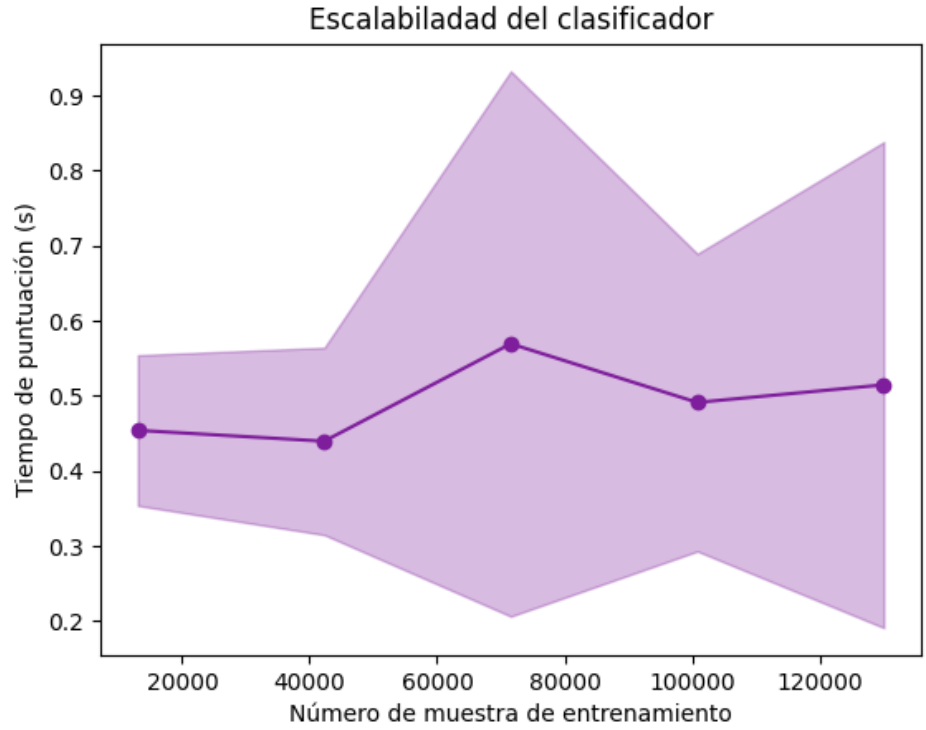


Figura 4.25 Curva de aprendizaje de todas las características para la Red clasificadora.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.26 Análisis de complejidad de la Red clasificadora para todas las características.

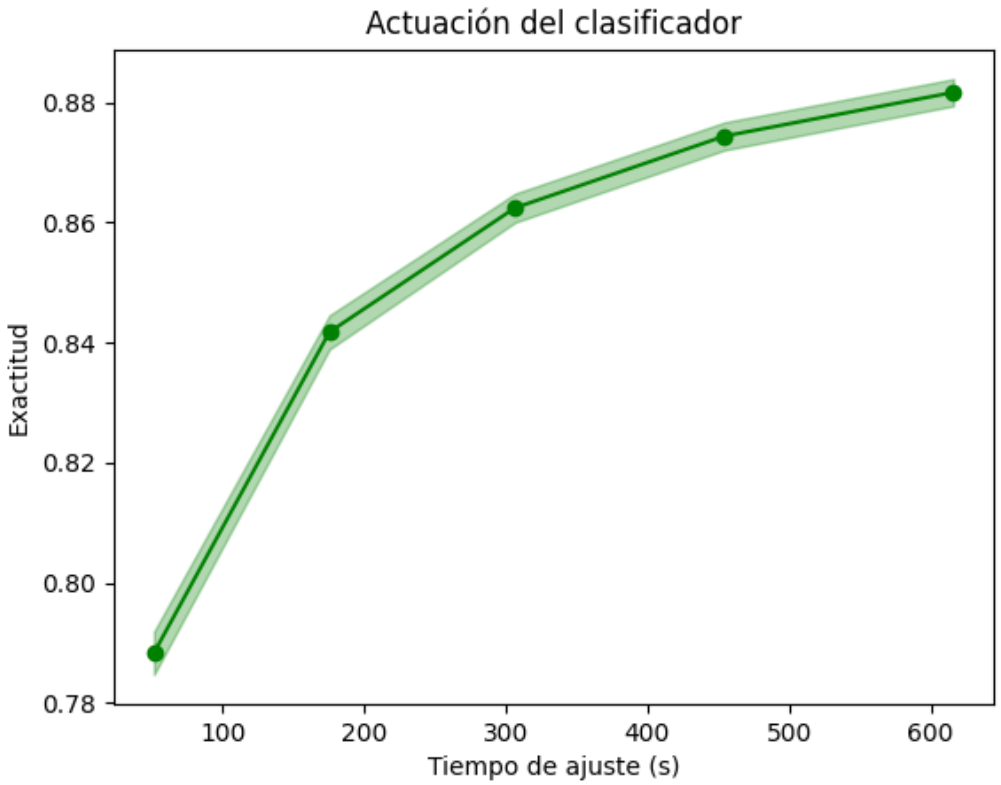


Figura 4.27 Actuación de la Red clasificadora para todas las características.

Comparación de las Redes Neuronales clasificadoras

Para los tres conjuntos de datos entrenados, se observa como los parámetros predeterminados no consiguen buenos resultados. En general, los modelo para ese caso aprenden una categoria y es el que ofrecen como salida para todas las filas de la validación.

Si se escalan los datos, los resultados mejoran de forma considerable. Al igual que al aplicar la combinación de los mejores parámetros. Si se juntan las dos opciones anteriores, se puede alcanzar el mejor porcentaje de aciertos.

El análisis de los componentes principales no es fructifero. Si se aplica sin preprocesar, se consiguen resultados muy malos. Si se escalan los datos primero, el número de fallos disminuye pero sigue siendo un porcentaje de fallos alto.

4.2.2 Regresión

Para las Redes Neuronales de regresión no es necesario modificar los datos de forma previa.

Características del artículo

Se comienza estudiando cuál es el mejor escalador para esta regresión. Dos de las opciones son aceptables, *Standard Scaler* y *MaxAbs Scaler*, teniendo la primera una peor actuación.

Tabla 4.32 Aciertos y fallos de los escaladores de la Red de regresión para las características del artículo.

	Porcentaje		Contador	
	Aciertos	Fallos	Acierto	Fallo
Sin Escalar	0,11	99,89	27	24457
Standard Scaler	18,68	81,32	4573	19911
MaxAbs Scaler	21,30	78,70	5216	19268
MinMax Scaler(-10,10)	1,34	98,66	329	24155

Se continua con el cambio de los parámetros de la red. El *solver* que mejores resultados obtiene es *adam*. La función de activación rectificadora *relu* acierta en más ocasiones que cualquier otra. En cuanto, a la tasa de actualización *adaptive*, la cual se mantiene constante siempre y cuando no existan perdidas, tiene el porcentaje verdadero más alto. El cambio de las capas intermedias de la red, tiene el mismo problema que en el caso de la clasificación. La combinación de los mejores parámetros se muestra en el Código 4.10.

Tabla 4.33 Resultados del cambio de los parámetros de la regresión de la Red Neuronal para las características del artículo.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	4,93	95,07	1208	23276
	sgd	0,53	99,47	129	24355
	lbfgs	0,05	99,95	12	24472
Activation	identity	0,04	99,96	11	24473
	logistic	0	100	0	24484
	tanh	1,08	98,92	264	24220
	relu	2,87	97,13	703	23781
Learning Rate	constant	2,77	97,23	677	23807
	adaptive	6,66	93,34	1631	22853
	invscaling	0	100	0	24484
Hidden Layers	500	0	100	0	24484
	900	0,004	99,996	1	24483

Código 4.10 "Mejores parámetros" de la Red Neuronal regresión para las características del artículo.

```
>>> regresor_opt = MLPRegressor(
                          solver = "adam",
                          activation = "adaptive", )
```

Tras esto, se analizan distintos casos de la aplicación de la Red de regresión para las características del artículo (Tabla 4.34). La iteración predeterminada no ofrece buenas estimaciones del riesgo porque se esta trabajando con muchas filas. Al escalar los datos se obtienen los mejores resultados, siendo la forma predeterminada la mejor, pero no aceptable. El análisis de componentes principales tiene un valor muy alto de fallos.

Tabla 4.34 Aciertos y fallos de los escaladores de la Red de regresión para las características del artículo.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Predeterminado	0	100	0	24484
MaxAbs Scaler	21,30	78,70	5216	19268
Mejores Param Sin Escalar	1,11	98,89	271	24213
Mejores Param Escalado	11,51	88,49	2819	21665
PCA	0,004	99,996	1	24483
Mejores Param PCA	2,26	97,74	554	23930

Además, se pueden analizar las variables características de la regresión para ver la calidad de las iteraciones. Los únicos valores de R^2 que tienen sentido son en los casos en los que el porcentaje de aciertos es alto. En los demás, se puede ver como el error absoluto es muy grande, siendo, así, la regresión no buena.

Tabla 4.35 Parámetros de la regresión de las distintas iteraciones de la Red Neuronal regresión para las características del artículo.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	-1,01E+81	3,42E+40	9,83E+82
MaxAbs Scaler	0,2506	0,0807	0,2137
Mejores Param Sin Escalar	-0,06764	9,52082	103,51452
Mejores Param Escalado	0,3135	0,0740	0,2294
PCA	-1,09E+77	3,55E+38	1,06E+79
Mejores Param PCA	-0,08211132	9,550601051	104,9172022

En la Figura 4.28 se ve como el modelo entrenado con los datos escalados no consigue una buena puntuación, pero se aprecia una tendencia creciente, por lo que si se tuviese más muestras de entrenamiento la puntuación podría llegar a ser mayor. En la Figura 4.29 se observa como el tiempo de ajuste aumenta de forma razonable con el número de muestras, que es lo que debe de pasar. El tiempo de puntuación es prácticamente estable para todos los subconjuntos. La actuación del regresor (Figura 4.30) muestra como la exactitud se mantiene constante, pese al aumento del tiempo de ajuste, y tiene bajo valor.

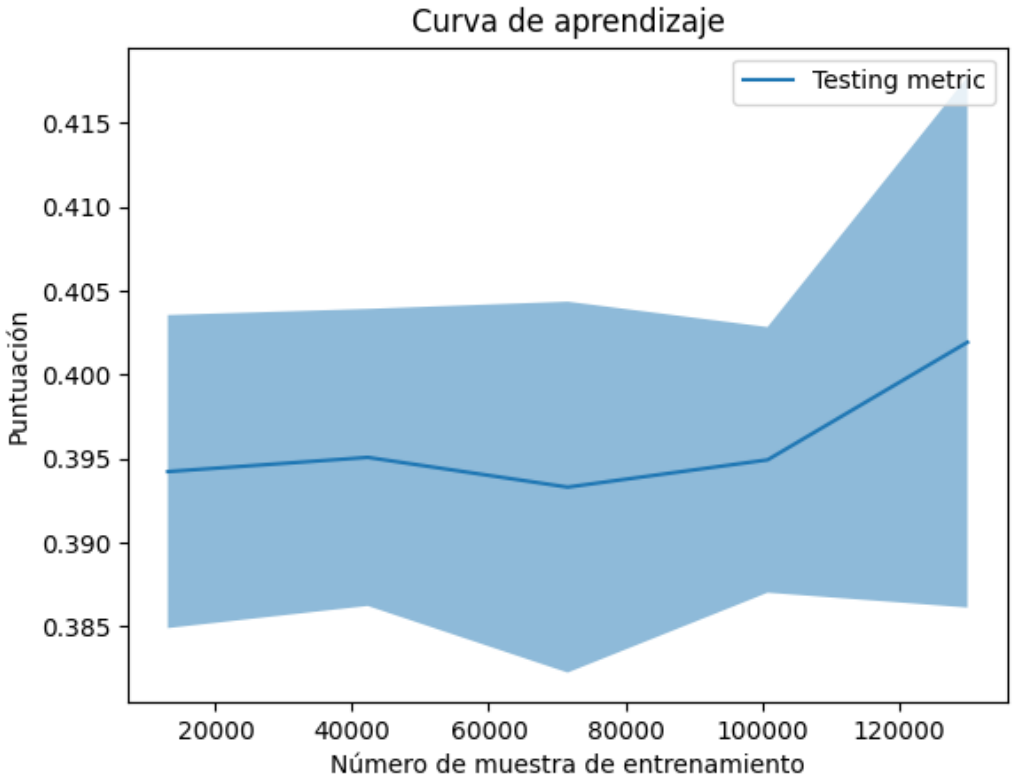
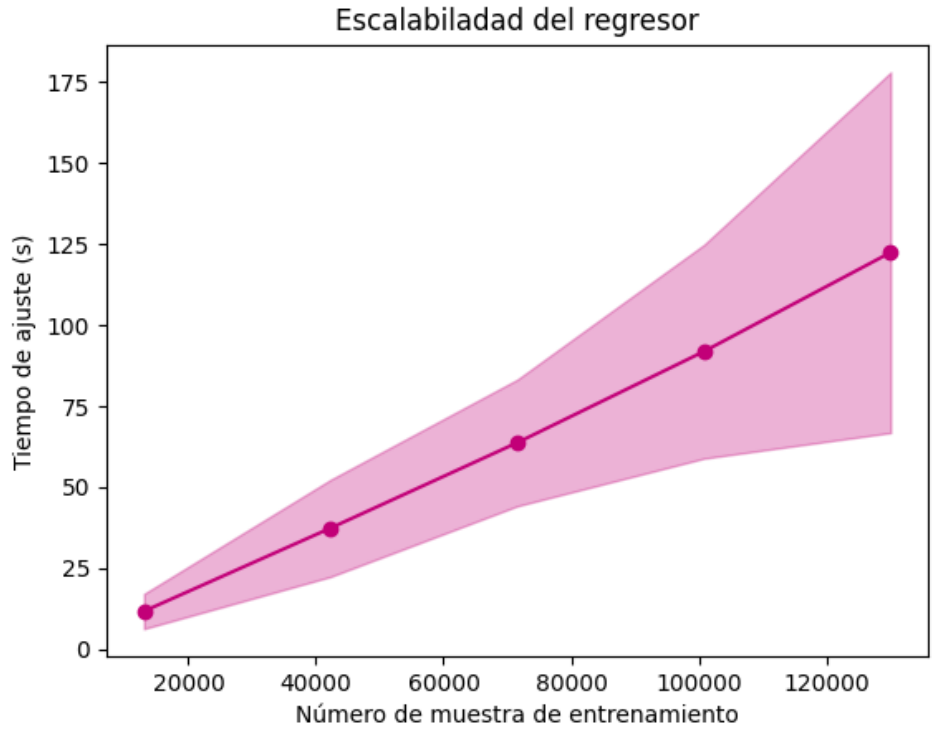
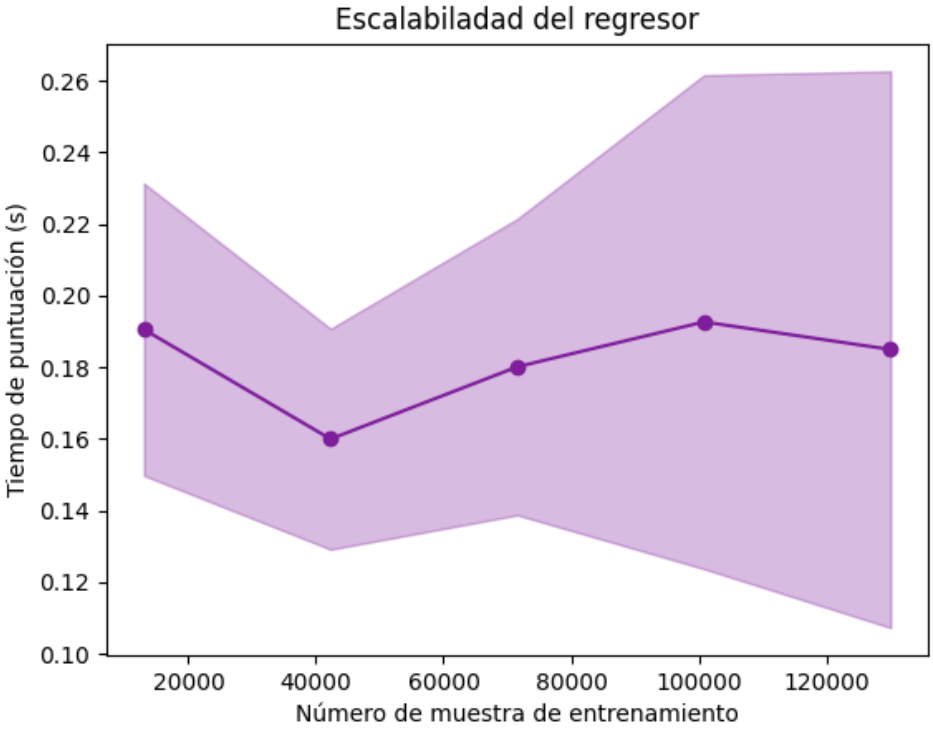


Figura 4.28 Curva de aprendizaje de las características del artículo para la Red de regresión.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.29 Análisis de complejidad de la Red de regresión para las columnas del artículo.

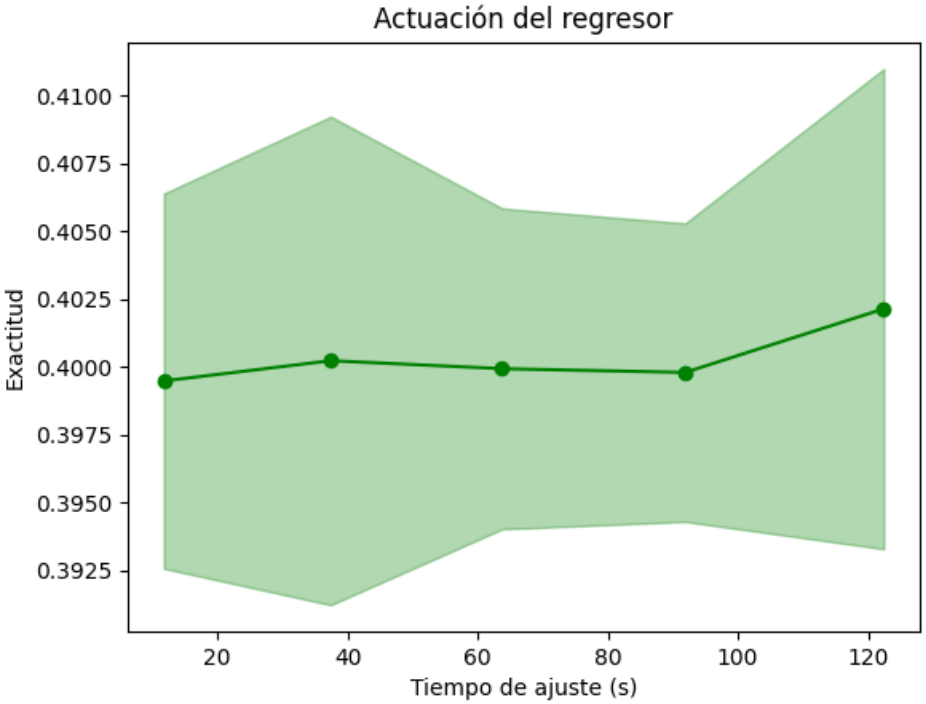


Figura 4.30 Actuación de la Red de regresión para las características del artículo.

Características de la tabla del artículo

Como para los casos anteriores, se comienza aplicando escaladores para ver como afectan en el número de aciertos y fallos. El *MinMax Scaler* consigue muy pocos aciertos y cuando no se escala, la red falla siempre. El *MaxAbs Scalrt* obtiene un buen porcentaje verdadero de lo estimado respecto a lo real.

Tabla 4.36 Aciertos y fallos de los escaladores de la Red de regresión para las características de la tabla del artículo.

	Porcentaje		Contador	
	Aciertos	Fallos	Acierto	Fallo
Sin Escalar	0	100	0	24484
Standard Scaler	6,33	93,67	15450	22934
MaxAbs Scaler	20,32	79,68	4974	19510
MinMax Scaler(-10,10)	0,93	99,07	229	24255

Hecho esto, se pasa al análisis de los distintos parámetros internos del *MLPRegressor* (Tabla 4.33). El método de resolución que mejor porcentaje verdadero tiene es *adam*. La función de activación con peores resultados es la logística, mientras que la función tangente hiperbólica da el menor número de fallos. Para la tasa de actualización de los pesos, la actualización constante es la que más aciertos obtiene. Para el número de capas intermedias, un cambio de valor al que no es el predeterminado no proporciona buenos resultados.

Tabla 4.37 Resultados del cambio de los parámetros de la regresión de la Red Neuronal para las características de la tabla del artículo.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	7,041333116	92,95866688	1724	22760
	sgd	1,008822088	98,99117791	247	24237
	lbfgs	0,0245058	99,9754942	6	24478
Activation	identity	0,065348799	99,9346512	0	24468
	logistic	0	100	0	24484
	tanh	1,127266786	98,87273321	276	24208
	relu	0,906714589	99,09328541	222	24262
Learning Rate	constant	5,905897729	94,09410227	1446	23038
	Adaptive	4,876654141	95,12334586	1194	23290
	invscaling	0	100	0	24484
Hidden Layers	500	0	100	0	24484
	900	0,004	99,996	1	24483

Código 4.11 "Mejores parámetros" de la red regresión para las características del articulo.

```
>>> regresor_opt = MLPRegressor(
                          solver = "adam",
                          activation = "tanh")
```

Ya definidos la combinación de los parámetros internos, se pasa al estudio de distintas iteraciones (Tabla 4.38). Cuando se entrena la red con los datos originales sin escalar, al validar el modelo se obtienen muchos fallos, es indiferente si se utilizan la combinación de los parámetros o si se aplica algún tipo de pre-descomposición matricial. Al escalar los datos, se obtienen buenos resultados. La red predeterminada escalada consigue un porcentaje verdadero bastante alto. Pero, el porcentaje verdadero más cercano a 100 % es el de la Red de regresión cuando se entrena con los datos escaldos y con la combinación de los mejores parámetros.

Tabla 4.38 Aciertos y fallos de las distintas iteraciones para las características de la tabla del artículo de la Red de regresión.

	Porcentaje		Contador	
	Acierto	Fallo	Acierto	Fallo
Predeterminado	0,00	100,00	0	24484
MaxAbs Scaler	20,32	79,68	4974	19510
Mejores Param Sin Escalar	1,12	98,88	275	24209
Mejores Param Escalado	16,20	83,80	3966	10518
PCA	0	100	0	24484
Mejores Param PCA	2,26	97,74	554	23930

Los atributos de la regresión (Tabla 4.39) muestran como las redes que obtienen malos resultados son las que mayor valor de error tienen y tiene valores de R^2 que no entran en el rango válido de éste. En el caso del *Mejores Param Escalado*, pese a tener un valor de R^2 no muy cercano a 1, tiene un error pequeño.

Tabla 4.39 Parámetros de la regresión de las distintas iteraciones de la Red Neuronal regresión para las características de la tabla del artículo.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	-6E+78	2,62E+39	5,77E+80
MaxAbs Scaler	0,4832	0,1797	0,0557
Mejores Param Sin Escalar	-0,0695	9,5298	103,6988
Mejores Param Escalado	0,2764	0,2215	0,0780
PCA	-4,6E+65	7,28E+32	4,46E+67
Mejores Param PCA	-0,0821	9,5506	104,9173

Ahora, se analizan las gráficas. La puntuación recibida por la red en el mejor de los casos anteriores, no es demasiado alta, lo cual corresponde al bajo valor del R^2 (Figura 4.31). Observando el análisis de complejidad (Figura 4.32), el tiempo de ajuste crece bastante con el número de muestras, llega a superar los tres minutos, mientras que el tiempo de puntuación crece lentamente. La actuación del regresor mostrada en la Figura 4.33 aumenta según lo hace el tiempo de ajuste, como debe ser pues la red tiene más tiempo para aprender.

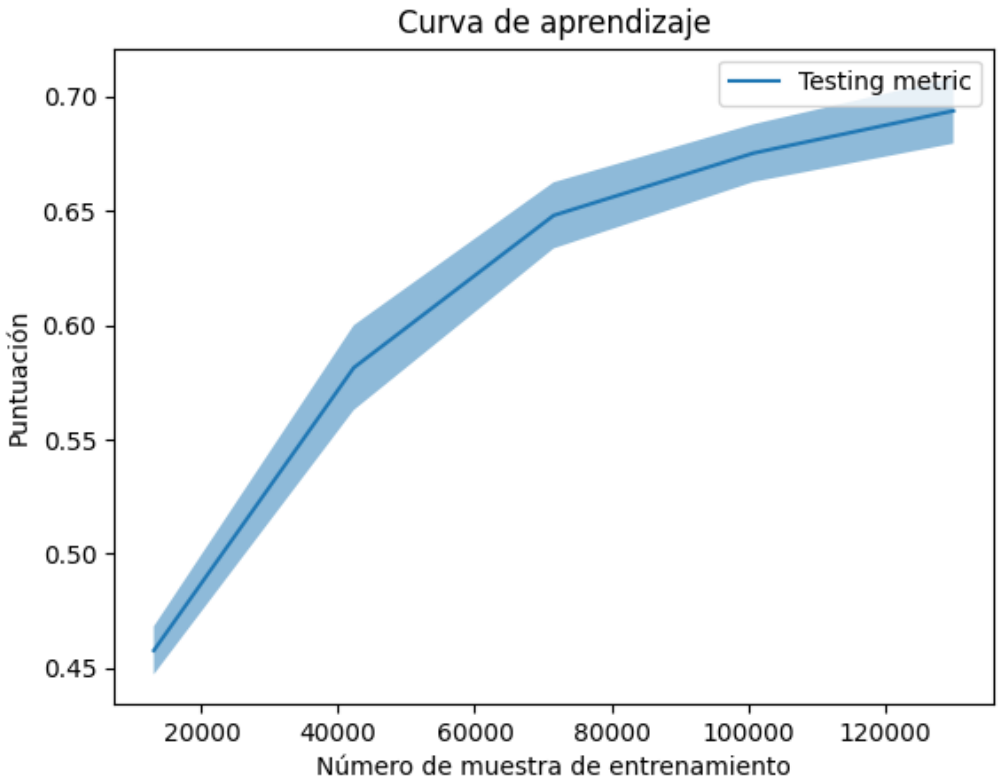
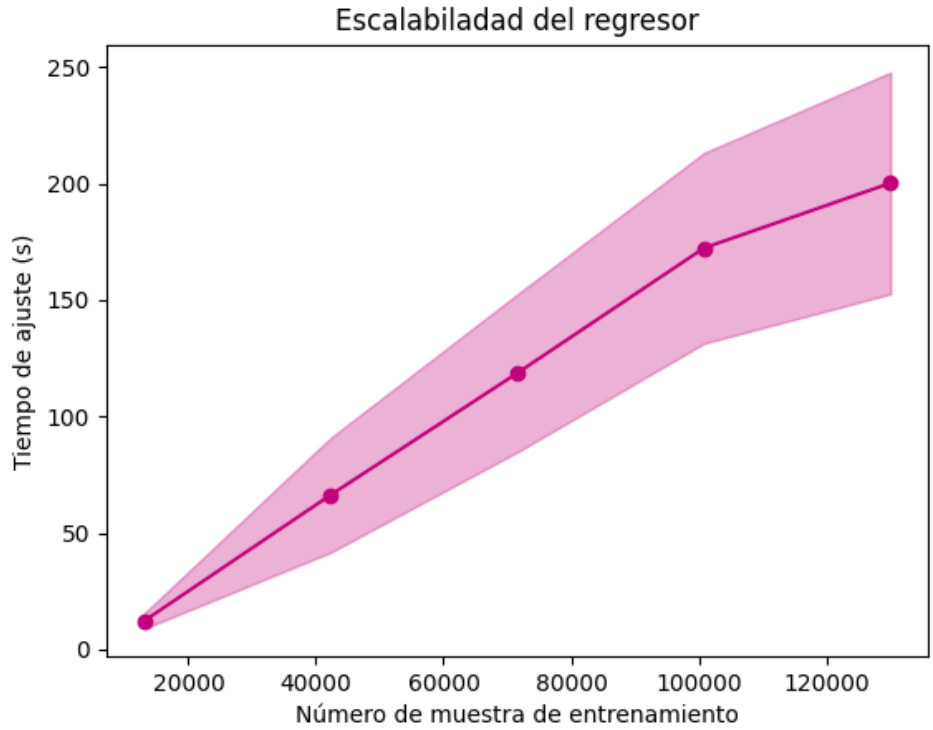
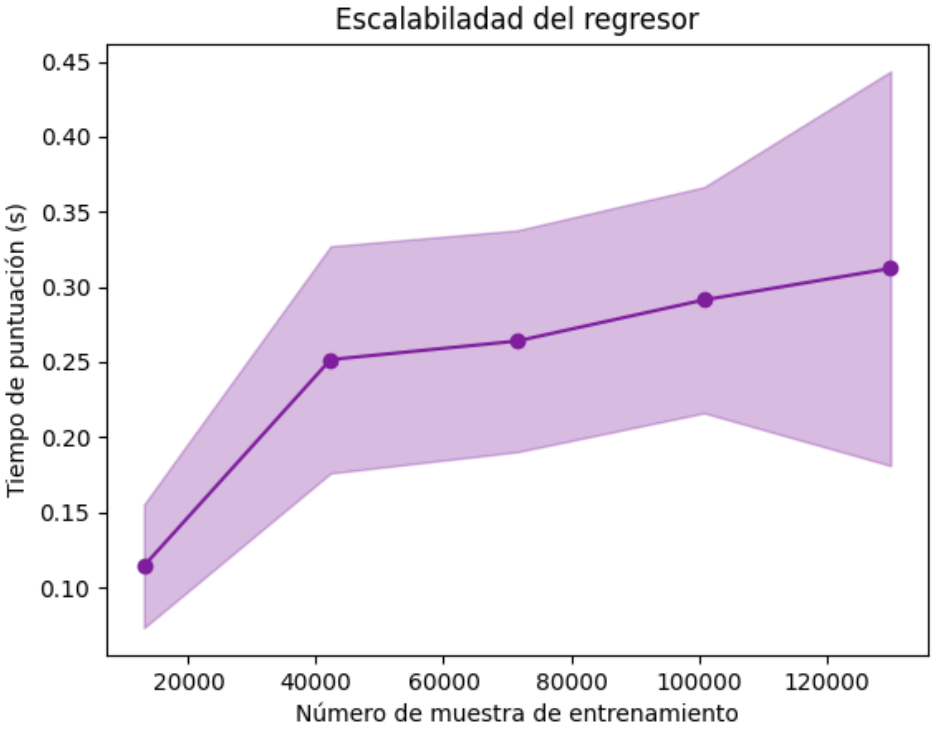


Figura 4.31 Curva de aprendizaje de las características de la tabla del artículo para la Red de regresión.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.32 Análisis de complejidad de la Red de regresión para las columnas de la tabla del artículo.

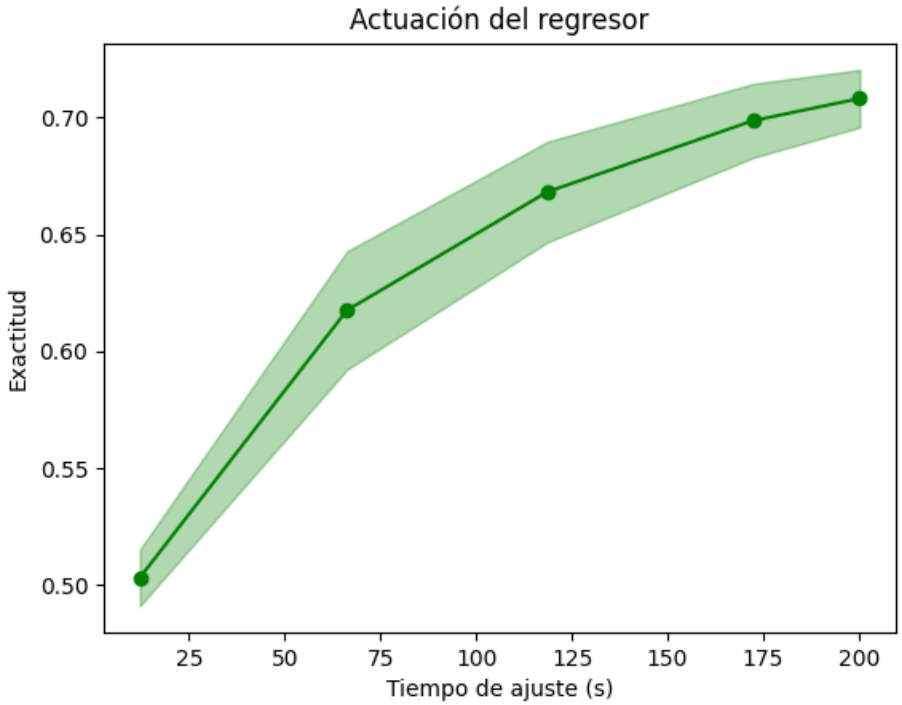


Figura 4.33 Actuación de la Red de regresión para las características de la tabla del artículo.

Todas las características

Se aplican los escaladores obteniendo los resultados mostrados en la Tabla 4.40. El mayor porcentaje de aciertos se tiene para el *Standard Scaler*, por lo que la estandarización para la red regresora cuando se trabaja con todas las características funciona bien. Los otros dos escaladores dan muchos fallos.

Tabla 4.40 Aciertos y fallos de los escaladores de la Red de regresión para todas las características del artículo.

	Porcentaje		Contador	
	Aciertos	Fallos	Acierto	Fallo
Sin Escalar	0	100	0	24484
Standard Scaler	6,83	93,17	1672	22812
MaxAbs Scaler	0,64	99,36	158	24326
MinMax Scaler(-10,10)	3,10	96,90	72	24412

A continuación, se analizan los efectos de los cambios en los parámetros internos de la Red de regresión (Tabla 4.41). En general, ninguno de los cambios individuales dejando los demás parámetros como vienen predeterminados por la librería no genera buenos resultados. Los porcentajes más altos de verdadero solo aciertan unas cientas de veces, lo que es muy poco como se ve al ser menor del 10 % en todas las iteraciones. La combinación de las mejores parámetros internos son los mostrados en Código 4.12.

Tabla 4.41 Resultados del cambio de los parámetros de la regresión de la Red Neuronal para todas las características.

		Porcentaje		Contador	
		Acierto	Fallo	Acierto	Fallo
Solver	adam	0,42	99,58	103	24381
	sgd	0,89	99,11	218	24266
	lbfgs	0	100	0	24484
Activation	identity	1,193	98,807	292	24192
	logistic	0	100	0	24484
	tanh	1,12	98,88	273	24211
	relu	3,47	96,53	850	23634
Learning Rate	constant	0	100	0	24484
	adaptive	1,75	98,25	428	24056
	invscaling	0	100	0	24484
Hidden Layers	500	0	100	0	24484
	900	0	100	0	24484

Código 4.12 "Mejores parámetros" de la red regresión para todas las características.

```
>>> regresor_opt = MLPRegressor(
                          solver = "sgd",
                          activation = "relu",
                          learning_rate = "adaptive")
```

Una vez realizado el análisis previo, se estudia los resultados de entrenamiento de la Red de regresión para distintos casos. Cuando no se escalan los datos, la Red de regresión al trabajar con todas las características solo falla. Cuando los datos se escalan, se obtienen algunas predicciones iguales a las verdaderas, pero no se supera el 10 % de aciertos. Ni el análisis de componentes principales ni la selección de características por el comando *SelectKBest*, las cuales son: *relative_velocity_n*, *mahalanobis_distance*, *t_position_covariance_det*, *c_position_covariance_det*, *t_sigma_r*, *c_sigma_r*, *t_sigma_t*, *c_sigma_t*, *t_sigma_n*, *c_sigma_n*, *t_sigma_rdot*, *c_sigma_rdot*, *t_sigma_tdot*, *c_sigma_tdot* y *t_sigma_ndot*.

Tabla 4.42 Resultados del cambio de los parámetros de la Red de regresión para todas las características.

	Porcentaje		Contador	
	Aciertos	Fallos	Acierto	Fallo
Predeterminado	0	100	0	24484
Standard Scaler	6,83	93,17	1672	22812
Mejores Param Escalado	4,96	95,04	1215	23269
PCA	0	100	0	24484
Mejores Param PCA	0	100	0	24484
Best K feature	0	100	0	24484

En la tabla a continuación, se ve como las iteraciones sin ningún acierto tienen un valor de R^2 que se salen del rango normal de éste. Además, los errores de esto son muy grandes. En

cambio, los valores de estos parámetros para *Standard Sclaer* y los mejores parámetros con los datos escalados son aceptables.

Tabla 4.43 Parámetros de la regresión de las distintas iteraciones de la Red Neuronal regresión para todas las características.

	R^2	Error absoluto medio	Error cuadrático medio
Predeterminado	-2,42E+95	2,35E+98	2,49E+56
MaxAbs Scaler	0,6089	0,4700	0,3911
Mejores Param Escalado	0,4966	0,5780	0,5034
PCA	-8,66E+100	8,39E+87	3,65E+58
Mejores Param PCA	-0,3013	0.9248923319133057	1,3013
Best K feature	-6,66E+98	6,45E+99	9,31E+55

En la curva de aprendizaje (Figura 4.34) se ve la baja puntuación que tiene la Red de regresión cuando los parámetros están escalados. Se ve una tendencia creciente pero con menor pendiente cuando mayores son el número de muestras de entrenamiento. El tiempo de ajuste ((a) Figura 4.35) es bastante grande, ya que al trabajar con muchas características, la red necesita más tiempo para adaptarse. El tiempo de puntuación ((b) Figura 4.35)es estable frente al aumento de muestras y tiene un valor bajo. La actuación del regresor (Figura 4.36) va mejorando con el tiempo de ajuste.

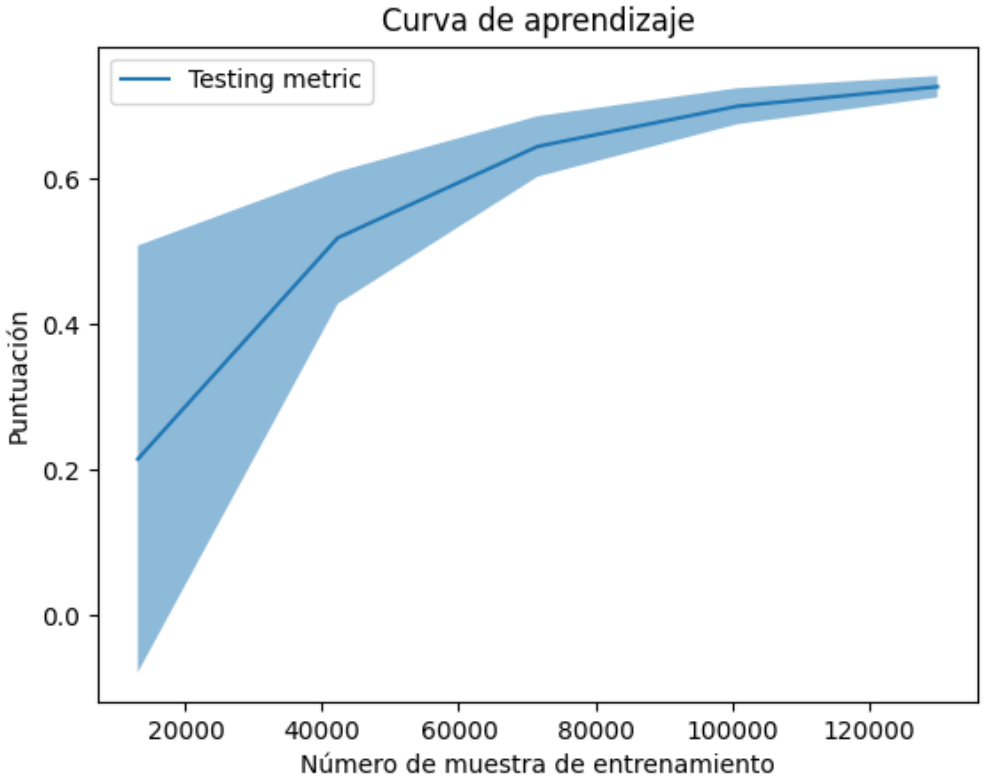
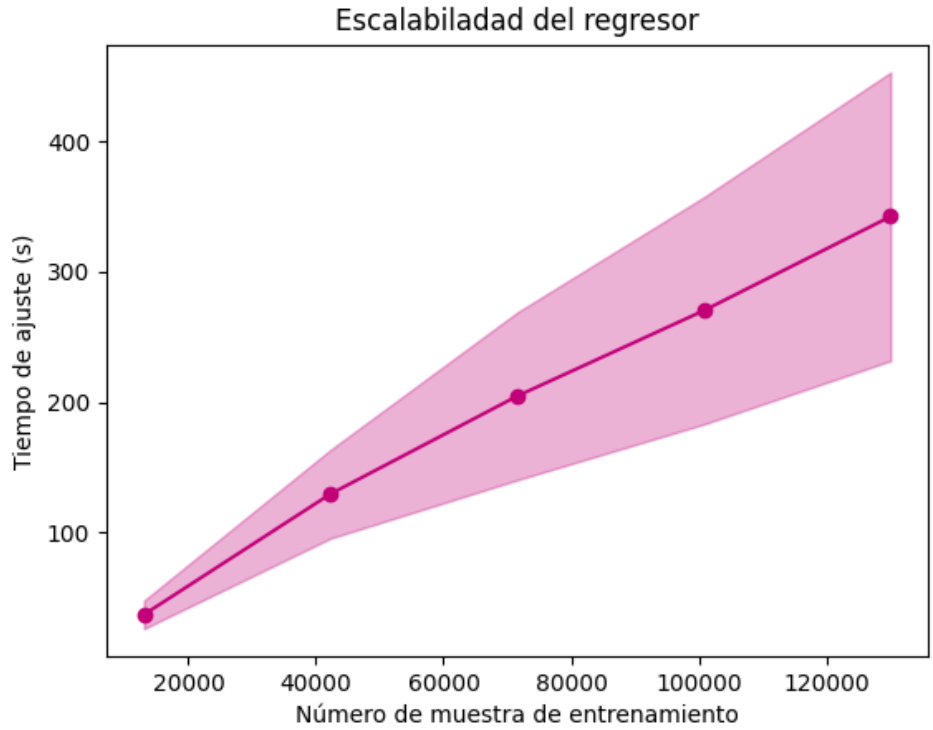
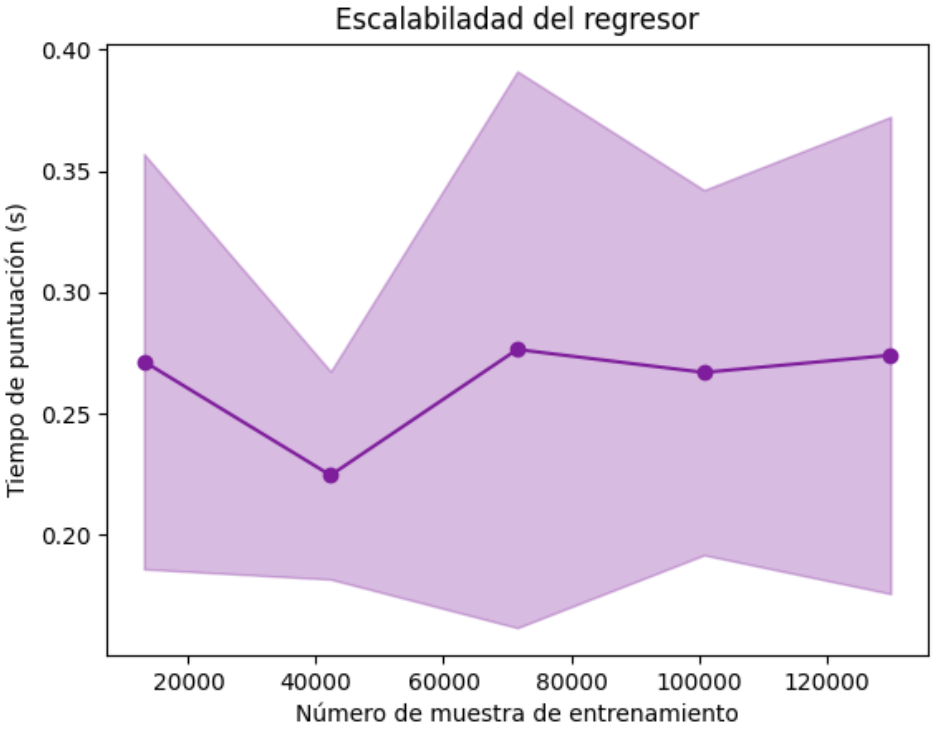


Figura 4.34 Curva de aprendizaje de todas las características para la Red de regresión.



(a) Tiempo de ajuste frente al número de muestras



(b) Tiempo de puntuación frente a al número de muestras

Figura 4.35 Análisis de complejidad de la Red Neuronal de regresión para todas las características.

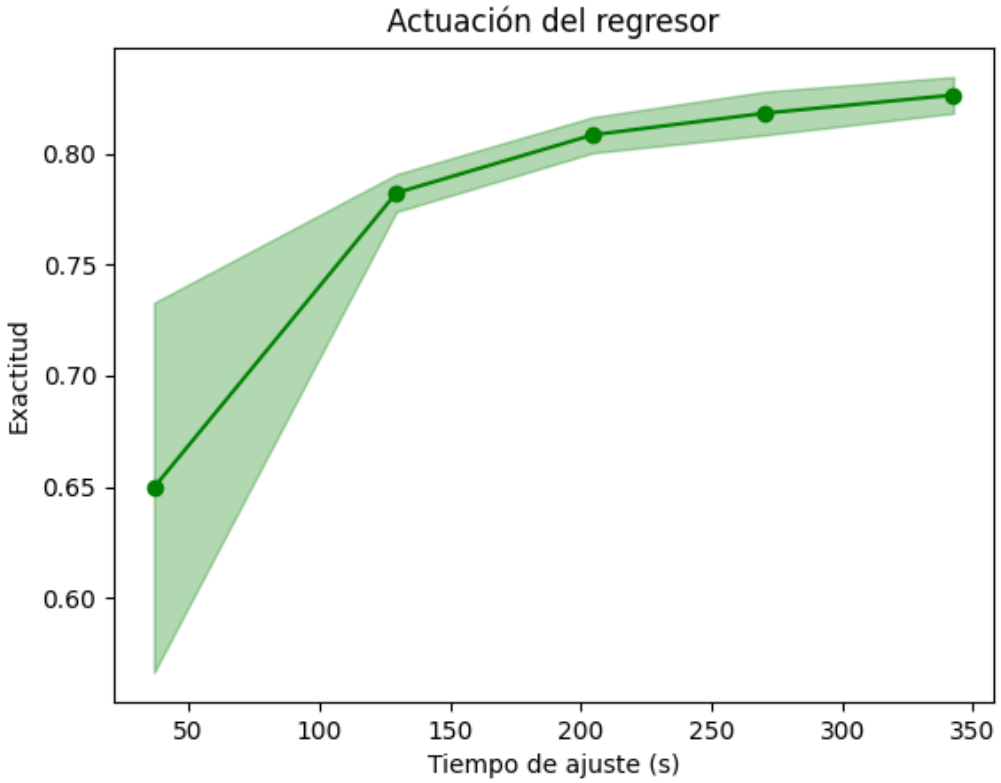


Figura 4.36 Actuación de la Red Neuronal de regresión para todas las características.

Comparación de las Redes Neuronales de regresión

En general, las redes de regresión obtienen un número aceptable de aciertos si se entrenan habiendo escalado de forma previa los datos. Ejecutando la red de la forma predeterminada no se ven aciertos. La combinación de los mejores parámetros al combinarla con los datos escalados, consiguen un porcentaje de aciertos bastante alto. El análisis de componentes principales no da buenos resultados.

En cuanto al tiempo de ajuste, éste crece de forma considerable cuantas mayor sea el número de muestras (filas) y características (columnas) con las que se entrena. Sin embargo, el tiempo de puntuación se mantiene estable con el número de muestras y aumenta poco si se añaden más características.

5 Análisis de resultados

5.1 Árboles de decisión

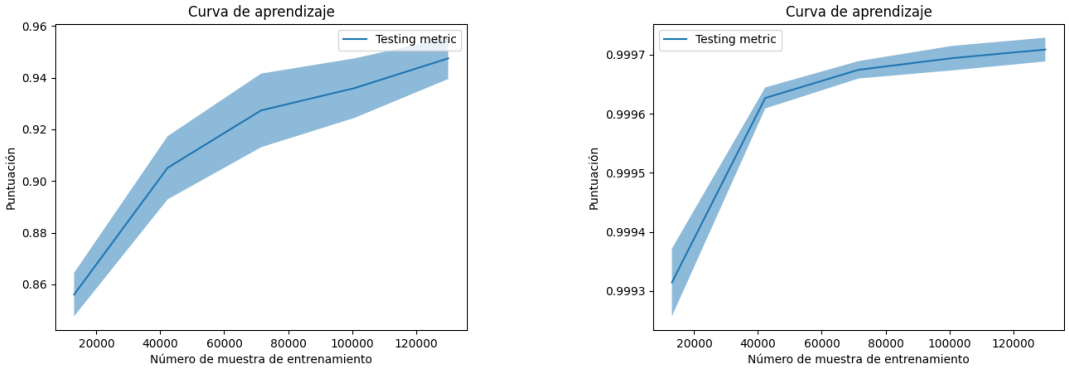
Los Árboles de Decisión son un método de aprendizaje supervisado no paramétrico, el cual sigue una estrategia de "divide-y-vencerás" y tiene estructura jerárquica compuesta por nodos y ramas. Se pueden entrenar para clasificación y regresión.

El Árbol de Decisión clasificador definido por scikit es muy exacto. Los porcentajes de aciertos son cercanos a 100 % en los tres casos de estudio presentados. El preprocesar los datos escalándolos, hace que las estimaciones empeoren. Utilizando el *MaxAbs Scaler* se pueden llegar a conseguir las mejores predicciones dentro de los escaladores, pero no llega el 50 % de aciertos. El Árbol clasificador acierta en más ocasiones cuando el modelo se entrena con las columnas del artículo, debido a que son el menor número de características a tener en cuenta y los Árboles trabajan mejor cuando más simples sean. Además, cualquier tipo de preprocesamiento que se realice antes del entrenamiento afecta negativamente a las predicciones a hacer en el set de validación del modelo.

Los Árboles de Decisión de regresión realizan una muy buena estimación del riesgo, mejor que el clasificador. Los escaladores tienen el mismo efecto y razón que para el modelo de clasificación anterior. Para los casos del entrenamiento con el conjunto de datos del artículo y de la tabla del artículo, la combinación de los "mejores parámetros" respectivos hacen que el regresor sea lo más exacto posible.

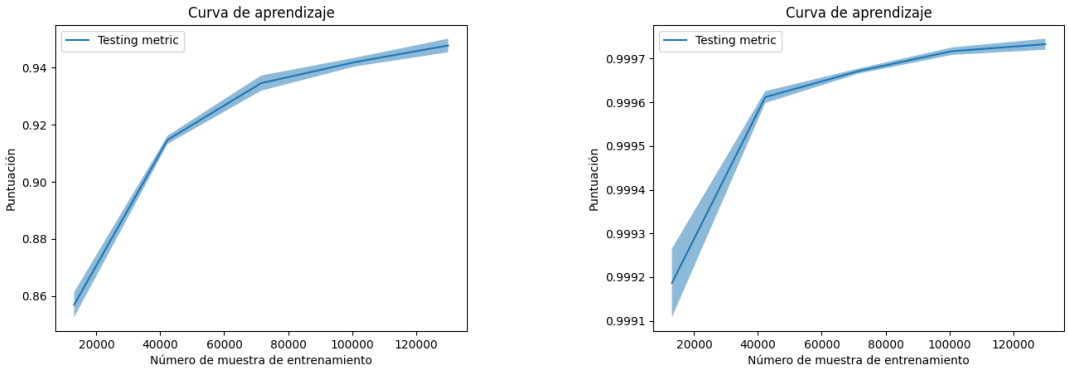
Para ambos tipo de Árboles, en el entrenamiento con todas las características se aplica el preprocesador *SelectKBest*, utilizado para identificar las columnas con el mayor peso, no obtiene buenas estimaciones del riesgo, pese a que algunas de las características seleccionadas coinciden con las del artículo o las de la tabla del artículo. También, se hace que el Árbol seleccione 15 características y se entrene. Esto no hace que el número de fallos en la validación se bajo.

En general, los Árboles de regresión tienen mayor puntuación durante el entrenamiento, como se ve en las curvas de aprendizaje (Figura 5.1, Figura 5.2 y Figura 5.3). Los Árboles clasificadores mejoran la puntuación de forma considerable conforme el número de muestras aumenta, en cambio en los Árboles de regresión el cambio se produce en el cuarto decimal por lo que es casi imperceptible. Además, conforme se entrenan ambos modelos con mayor número de características la puntuación tiene menores valores.



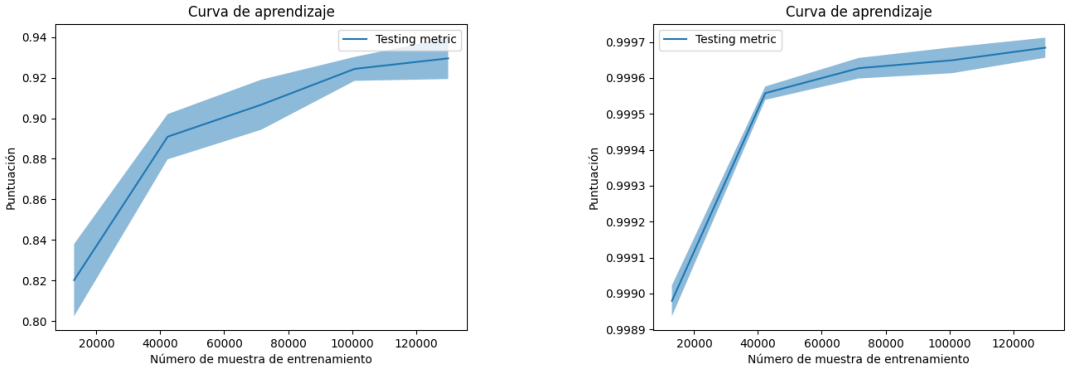
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.1 Curva de aprendizaje de los Árboles de Decisión para las características del artículo.



(a) Árbol clasificador (b) Árbol de regresión

Figura 5.2 Curva de aprendizaje de los Árboles de Decisión para las características de la tabla del artículo.



(a) Árbol clasificador (b) Árbol de regresión

Figura 5.3 Curva de aprendizaje de los Árboles de Decisión para todas las características.

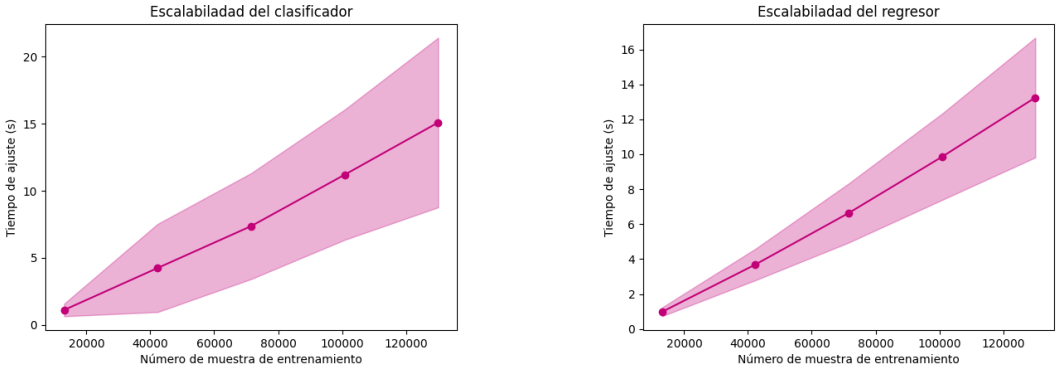
El tiempo de ajuste del árbol (clasificador y regresor) tiene tendencia creciente con el número de muestras y de características, lo cual se ve en la evolución de las gráficas de análisis de complejidad. Además, comparando éstos tiempos para cada caso (Figura 5.4, Figura 5.5 y Figura 5.6), se concluye que el Árbol de regresión tarda ligeramente menos en realizar el ajuste para los subconjuntos de muestras. También, se observa como los tiempos de ajuste en ambos casos crece conforme más características tiene el modelo para entrenar.

Respecto al tiempo de puntuación, éste es prácticamente estable, ya que las variaciones que se ven en las gráficas son muy pequeñas. Cuando las características de entrada son las del artículo (Figura 5.10) o las de la tabla del artículo (Figura 5.11), el Árbol de regresión es menor que el clasificador. El tiempo de puntuación para todas las características es prácticamente igual para ambos tipos de Árboles (Figura 5.9). Además, hay que destacar que los tiempos de puntuación son muy pequeños (menores de 1 segundo).

En cuanto a la actuación de los Árboles (Figura 5.10, Figura 5.11 y Figura 5.12), a partir de las gráficas anteriores se ve como la exactitud aumenta con el tiempo de ajuste, debido a que el modelo cuenta con más tiempo para adaptarse y aprender de forma correcta. Los Árboles clasificadores son menos exactos que los de regresión para los tres casos estudiados.

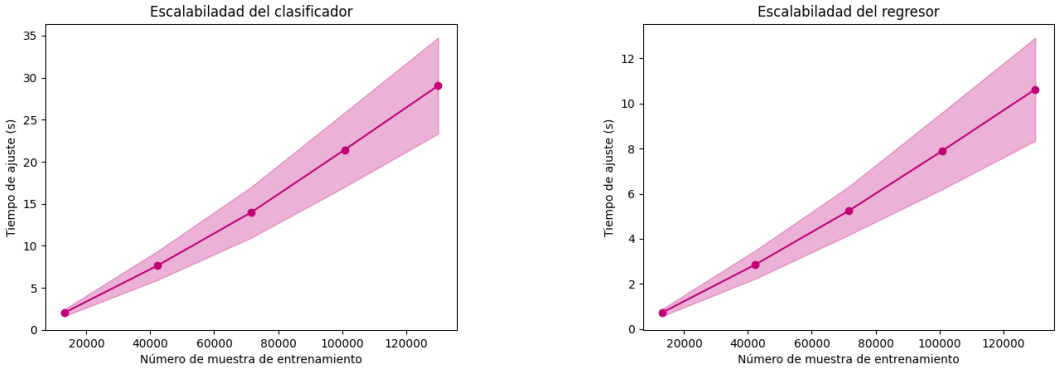
Por tanto, los modelos predeterminados de Árboles por scikit son los mejores, ya que éstos vienen optimizados. Los Árboles de regresión obtienen un mayor número de predicciones acertadas.

Es importante destacar que en el desafío se pide clasificar en dos categorías: "Riesgo" o "No Riesgo", siendo el límite de éstas el valor -5. Si se escogen los mejores modelos de entrenamiento y se categorizan, las predicciones y lo verdadero coincide al 100 %.



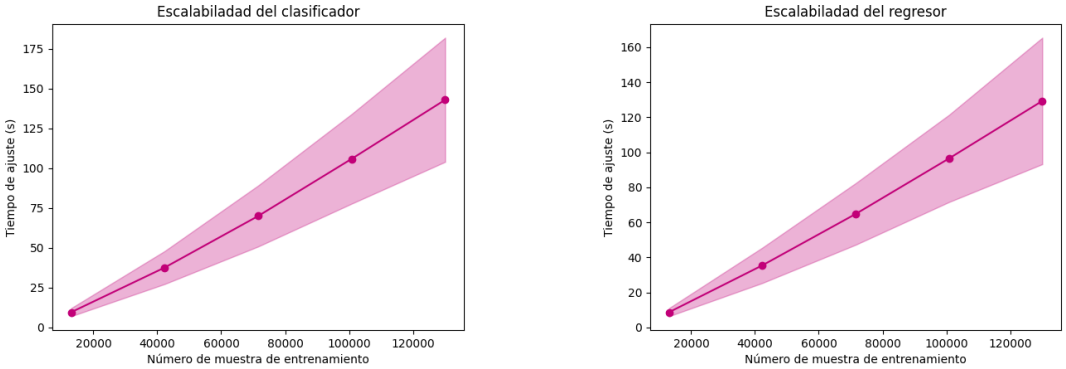
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.4 Comparación de los tiempos de ajuste frente al número de las muestras de los Árboles de Decisión para las características del artículo.



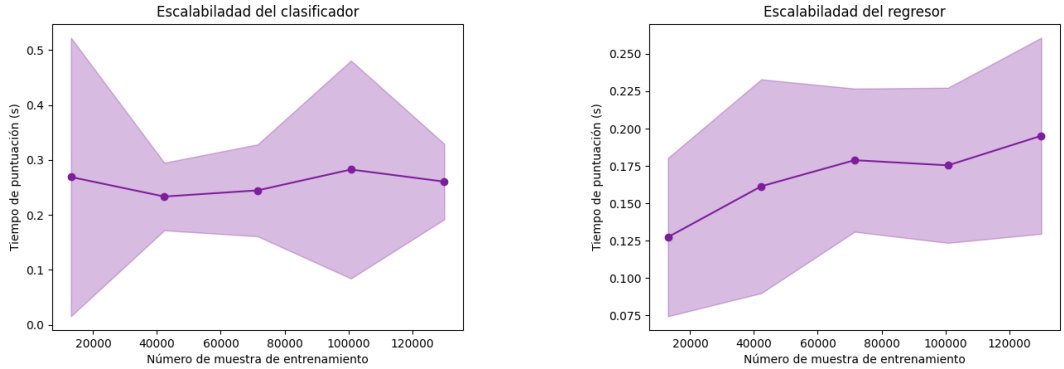
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.5 Comparación de los tiempos de ajuste frente al número de las muestras de los Árboles de Decisión para las características de la tabla del artículo.



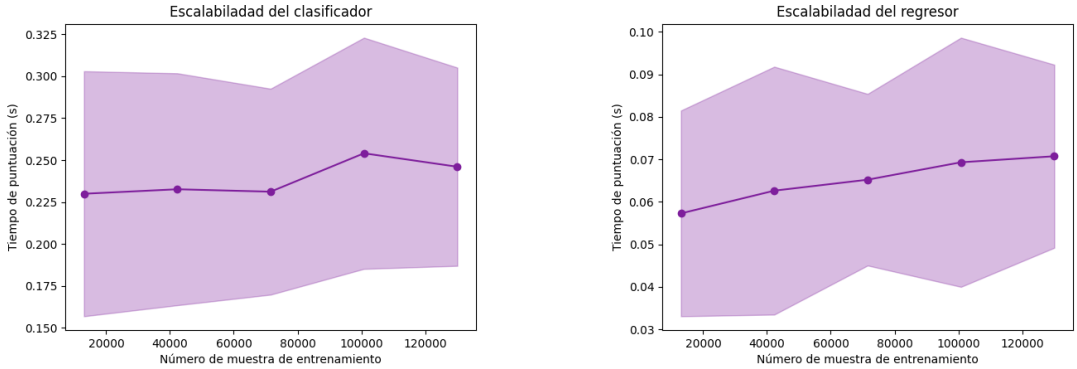
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.6 Comparación de los tiempos de ajuste frente al número de las muestras de los Árboles de Decisión para todas las características.



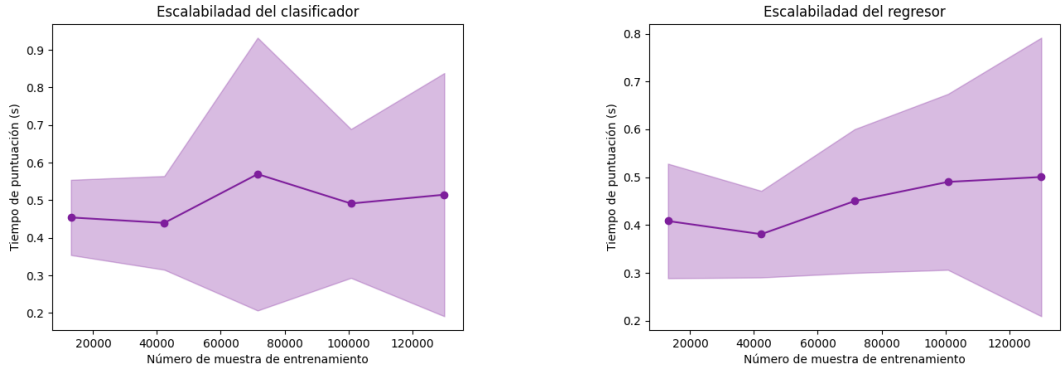
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.7 Comparación de los tiempos de puntuación frente al número de las muestras de los Árboles de Decisión para las características del artículo.



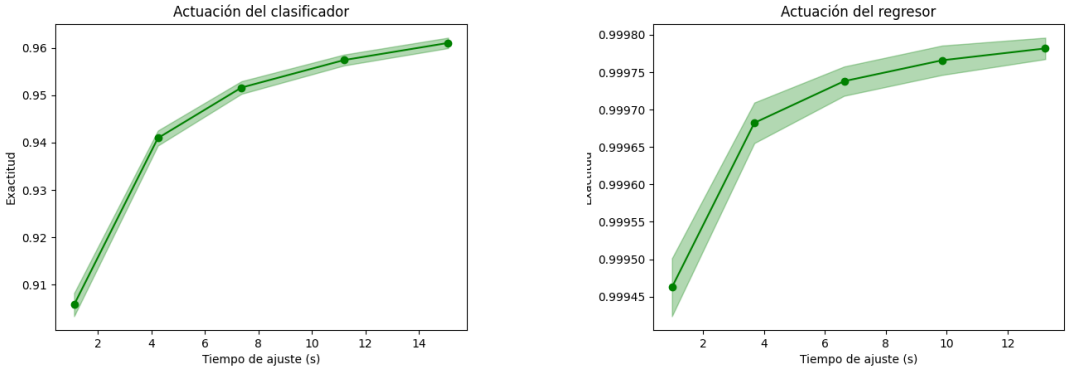
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.8 Comparación de los tiempos de puntuación frente al número de las muestras de los Árboles de Decisión para las características de la tabla del artículo.



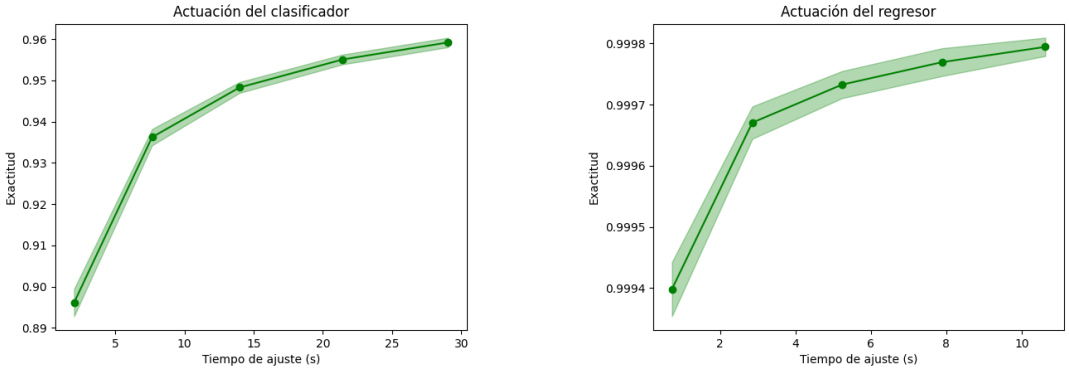
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.9 Comparación de los tiempos de puntuación frente al número de las muestras de los Árboles de Decisión para todas las características.



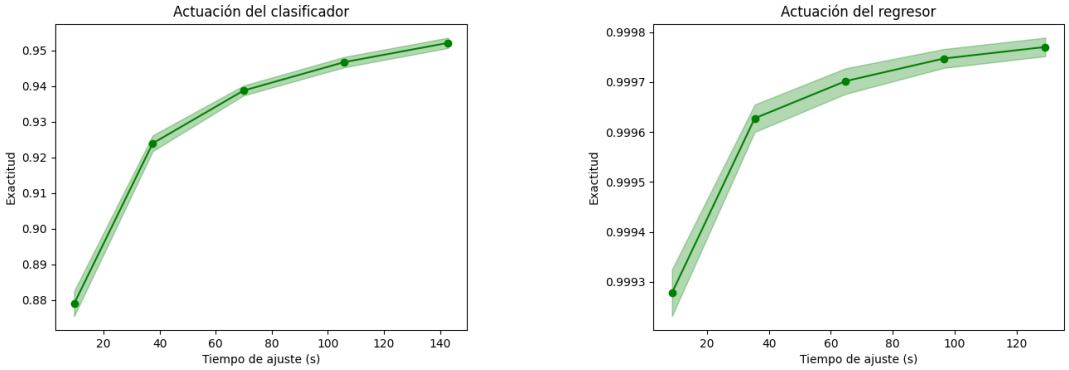
(a) Árbol clasificador (b) Árbol de regresión

Figura 5.10 Comparación de la actitud de los Árboles de Decisión para las características del artículo.



(a) Árbol clasificador (b) Árbol de regresión

Figura 5.11 Comparación de la actitud de los Árboles de Decisión para las características de la tabla del artículo.



(a) Árbol clasificador (b) Árbol de regresión

Figura 5.12 Comparación de la actitud de los Árboles de Decisión para todas las características.

5.2 Redes Neuronales

Las Redes Neuronales son algoritmos de aprendizaje supervisado que pretenden imitar el cerebro humano.

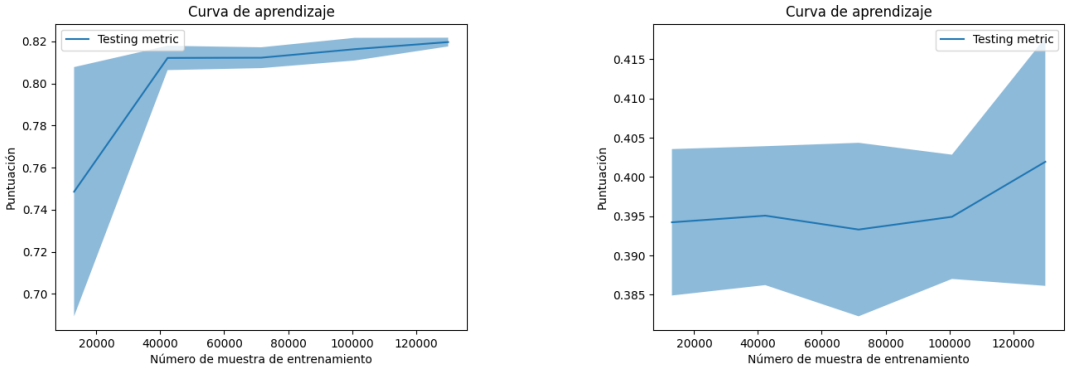
A grandes rasgos, las Redes Neuronales predeterminadas no consiguen buenos resultados. Esto se debe a que las redes aprendizaje supervisado de scikit-learn no están preparadas para trabajar con un conjunto de datos tan amplio. De hecho, analizando las predicciones que realiza cada modelo, se ve que la red aprende una clases (para el clasificador) o números muy grandes (para el regresor) por el hecho de que las redes no están preparadas para trabajar con tantos datos.Por ello, al escalar los datos se mejoran considerablemente los resultados.

Las Redes Neuronales clasificadoras al ser escaladas pueden llegar a tener porcentajes de aciertos aceptables. Las mejores opciones son la estandarización y el *MaxAbs Scaler*. El primero centra los resultados y hace que la red se ajuste mejor a los datos. El segundo escala las columnas de forma independiente. El mayor número de estimaciones verdaderas se dan cuando los datos están escalados y, en ocasiones, cuando los parámetros internos de la Red clasificadora son la combinación definida anteriormente. El entrenamiento con los las columnas de la tabla obtiene las mejores predicciones.

Las Redes Neuronales de regresión no llegan a tener un número alto de aciertos. Pese a que observando las gráficas se podría deducir que se van a obtener unas estimaciones aceptables, a la hora de validar el modelo esa exactitud que presentan los regresores no es así. Al escalar se consigue mejorar estas predicciones, pero los fallos siempre superan el 60 % de fallos.

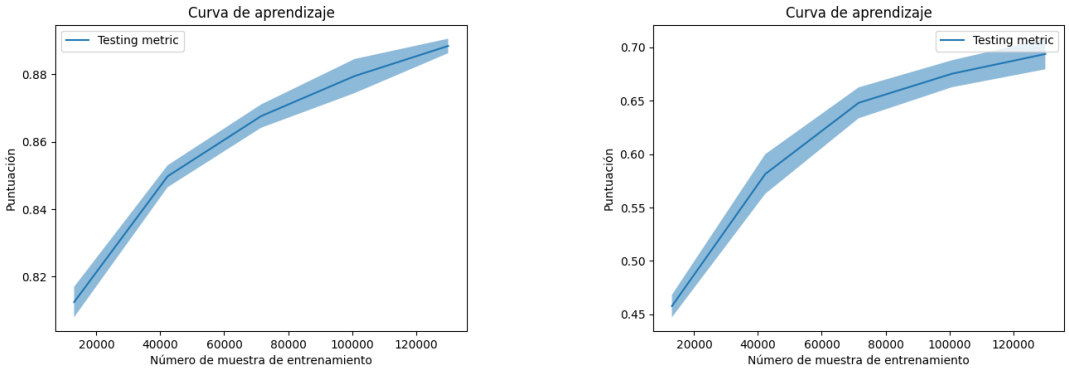
A ambos tipos de Redes se les aplica un preprocesamiento de análisis de componentes principales. Al emplearlo aumenta el número de aciertos frente a las Redes predefinidas, sobretodo si se modifican los parámetros a la combinación definidas en cada caso. En el caso del entrenamiento con todas las características de los datos, se aplica, también, el preprocesador *SelectKBest*. Los resultados para la Red clasificadora se ven empeoradas frente a la Red predeterminada, pese a que muchas de las características seleccionadas coinciden con las del artículo, y con los de la Red de regresión pasa lo mismo.

La puntuación de las Redes clasificadoras (Figura 5.13, Figura 5.14 y Figura 5.15), en general, es mejor que las de las de regresión, ya que el tiempo de ajuste es mayor para las primeras. En los análisis de complejidad (Figura 5.16, Figura 5.17, Figura 5.18, Figura 5.19,Figura 5.20 y Figura 5.21), se ve la diferencia entre esos tiempos de ajuste y como los tiempos de puntuación son prácticamente los mismos. Comparando las actuaciones de las Redes (Figura 5.22, Figura 5.23 y Figura 5.24), se ve que las clasificadores tienen mayor exactitud en los entrenamientos que las regresoras, lo que se ve reflejado en las predicciones.



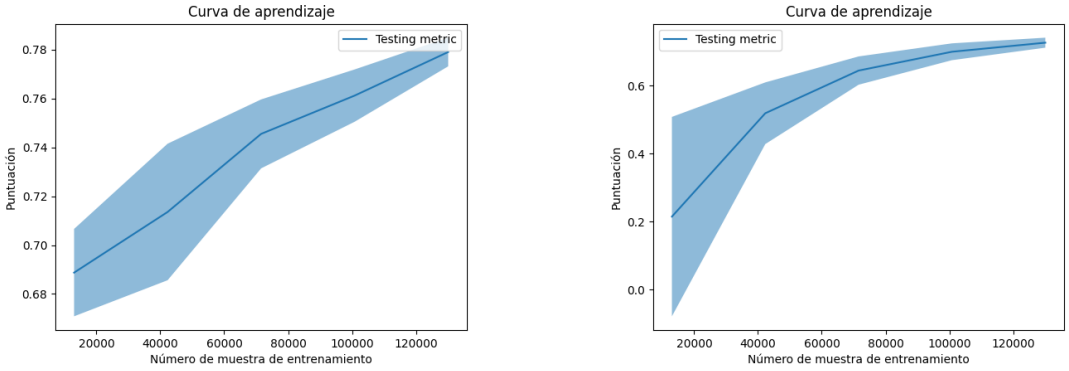
(a) Red clasificadora (b) Red de regresión

Figura 5.13 Curva de aprendizaje de las Redes Neuronales para las características del artículo.



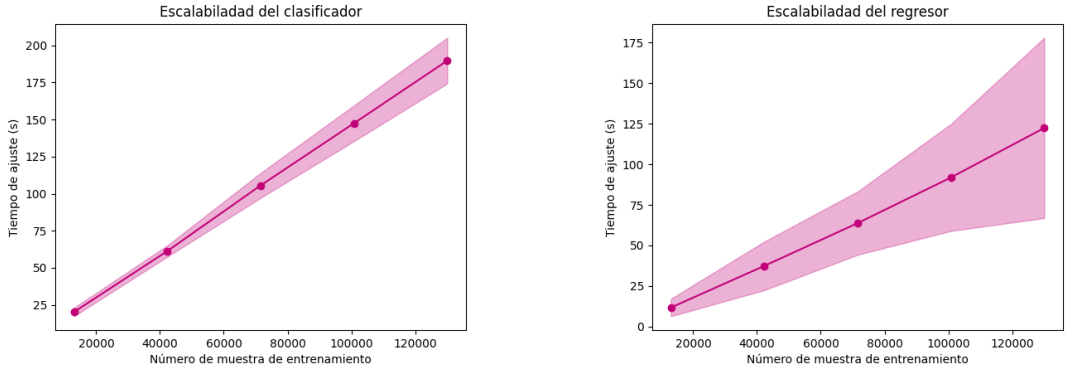
(a) Red clasificadora (b) Red de regresión

Figura 5.14 Curva de aprendizaje de las Redes Neuronales para las características de la tabla del artículo.



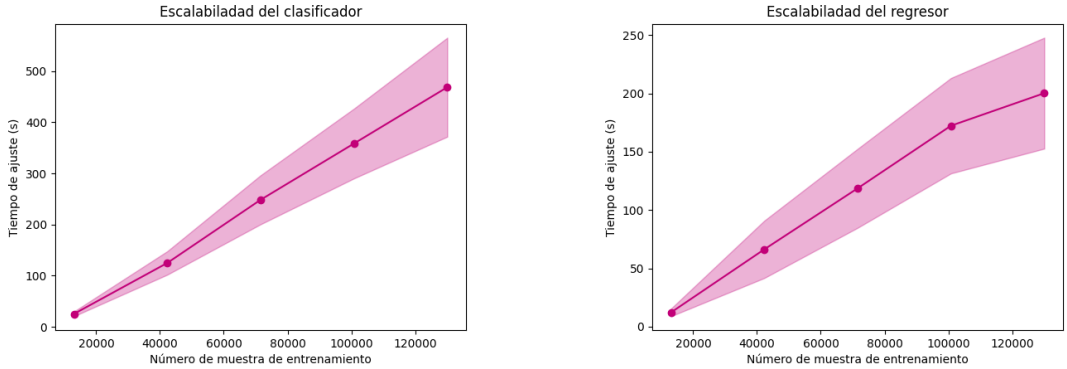
(a) Red clasificadora (b) Red de regresión

Figura 5.15 Curva de aprendizaje de las Redes Neuronales para todas las características.



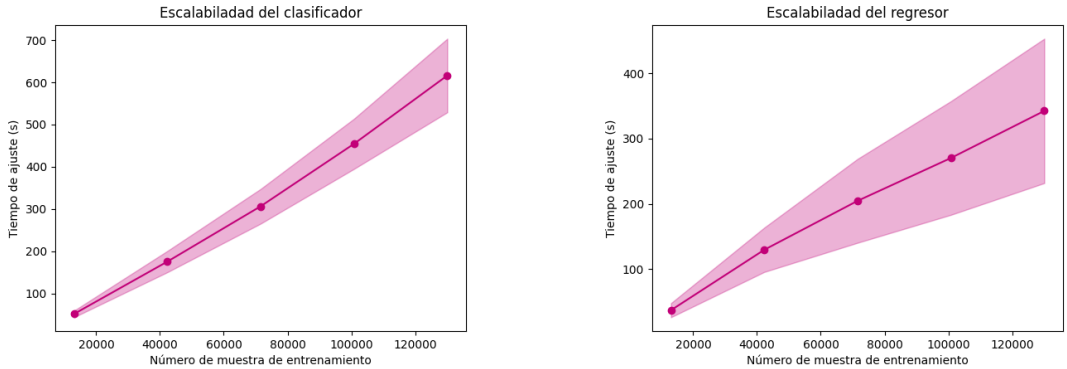
(a) Red clasificadora (b) Red de regresión

Figura 5.16 Comparación de los tiempos de ajuste frente al número de las muestras de las Redes Neuronales para las características del artículo.



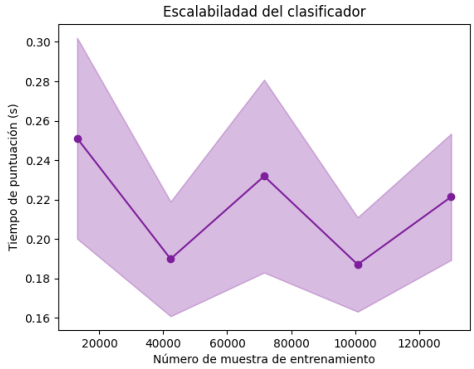
(a) Red clasificadora (b) Red de regresión

Figura 5.17 Comparación de los tiempos de ajuste frente al número de las muestras de las Redes Neuronales para las características de la tabla del artículo.

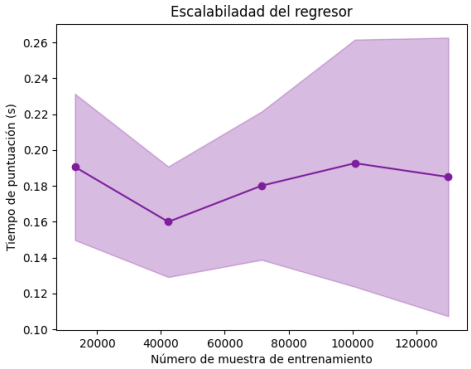


(a) Red clasificadora (b) Red de regresión

Figura 5.18 Comparación de los tiempos de ajuste frente al número de las muestras de las Redes Neuronales para todas las características.

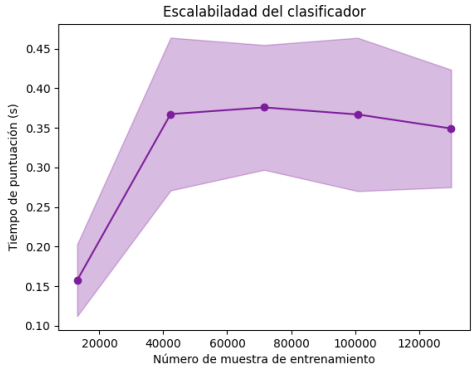


(a) Red clasificadora

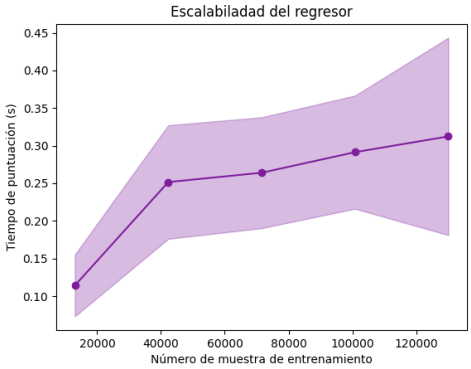


(b) Red de regresión

Figura 5.19 Comparación de los tiempos de puntuación frente al número de las muestras de las Redes Neuronales para las características del artículo.

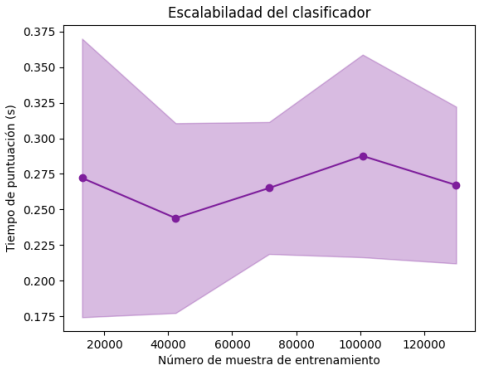


(a) Red clasificadora

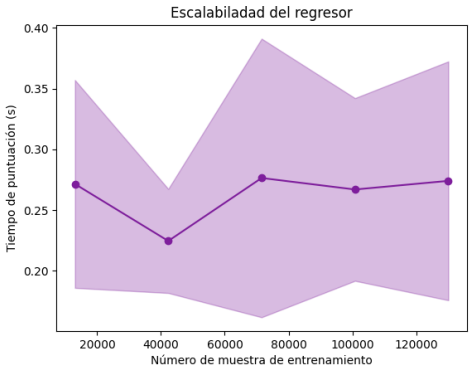


(b) Red de regresión

Figura 5.20 Comparación de los tiempos de puntuación frente al número de las muestras de las Redes Neuronales para las características de la tabla del artículo.

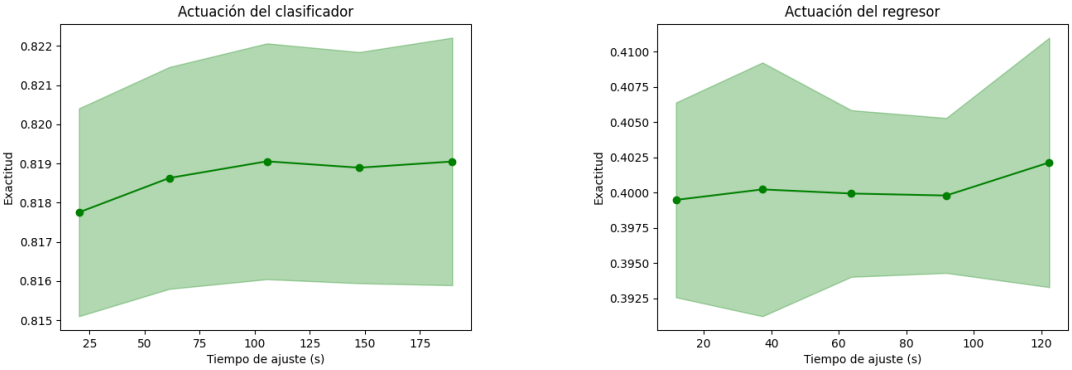


(a) Red clasificadora



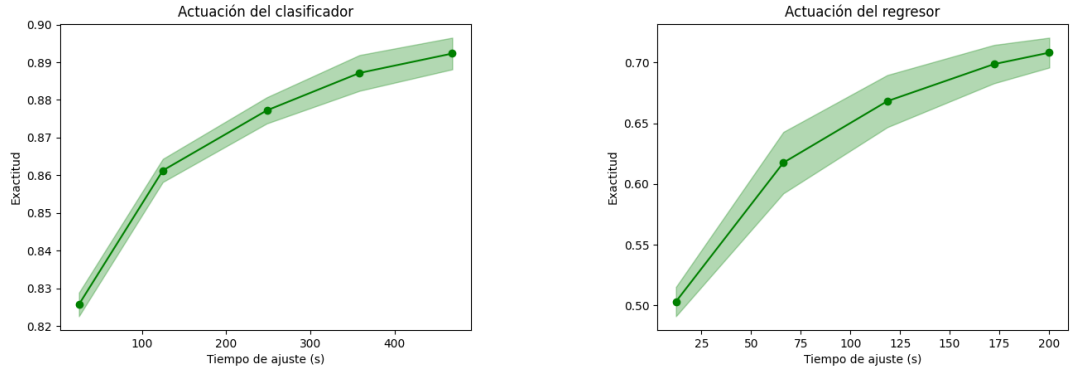
(b) Red de regresión

Figura 5.21 Comparación de los tiempos de puntuación frente al número de las muestras de las Redes Neuronales para todas las características.



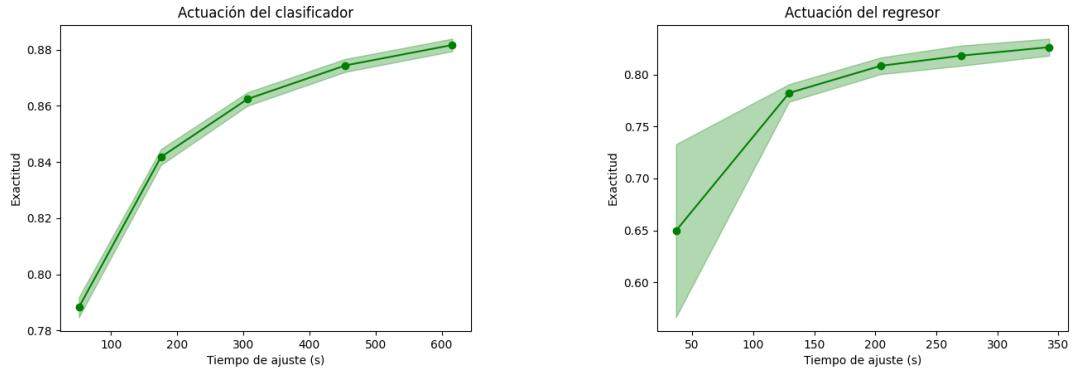
(a) Red clasificadora (b) Red de regresión

Figura 5.22 Comparación de la actitud de las Redes Neuronales para las características del artículo.



(a) Red clasificadora (b) Red de regresión

Figura 5.23 Comparación de la actitud de las Redes Neuronales para las características de la tabla del artículo.



(a) Red clasificadora (b) Red de regresión

Figura 5.24 Comparación de la actitud de las Redes Neuronales para todas las características.

5.3 Comparación de los métodos de aprendizaje supervisado

Los Árboles de Decisión obtienen mejores resultados que las Redes Neuronales, porque éstas en la librería scikit no están preparadas para trabajar con un amplio conjunto de datos.

Como se ha explicado antes, la descomposición de los datos usando el análisis de componentes principales aplicados en la red no obtiene porcentajes verdaderos mayores del 50 %. Los resultados del empleo de esto en los Árboles no se muestran pues el porcentaje de fallo era aún mayor que en el caso de las redes.

Para los dos métodos, cuando se trabaja con todas las características de los datos, se ha empleado un método de selección (*SelectKBest*), el cual guarda las k características con los mayores pesos. Pese a que la mayoría de las columnas elegidas por el comando coinciden con las del artículo y las de la tabla, pero cuando se aplican no se consiguen buenas predicciones.

Con los clasificadores de ambos métodos de aprendizaje, se obtienen resultados muy buenos con el entrenamiento con las características del artículo o de la tabla. Hay que recordar que el tiempo hasta la aproximación más cercana (*time_to_tca*) del conjunto de datos de validación del modelo (*tets_data.csv*) no son menores de dos días, que es cuando se empiezan a barajar las posibles maniobras para evitar la colisión, por lo que algunos CDMs categorizados como "no riesgo" podían cambiar a "riesgo". Pero como primera estimación, considerar solo algunas características de los CDMs es una buena solución. El Árbol de regresión estima muy bien el riesgo. Sin embargo, las Redes de regresión no sirven para realizar predicciones de riesgo.

6 Conclusión y líneas futuras

Evitar la basura espacial es el día a día para las misiones espaciales orbitando alrededor de la Tierra. La Agencia Europea Espacial monitoriza todos estos objetos a través de los CDMs (Conjuction Data Message), mensajes sobre la información de un posible evento de colisión. Entre la información aportada por estos mensajes, está el tiempo hasta la posible colisión. Cuando éste es igual a dos días o menor, se comienzan a barajar estrategias para evitar las colisiones.

A partir de esto, la ESA propone el desafío de crear un modelo capaz de estimar el riesgo de colisión a partir de los CDMs. Para ello, la Agencia proporciona dos conjuntos de datos: uno de entrenamiento y otro de validación, el cual contiene la columna de riesgo para poder comprobar la validez del modelo. Además, en un artículo [5] la ESA analiza los CDMs y determina las características más importantes en la estimación.

Para poder completar el desafío se utiliza técnicas de Machine Learning, un tipo de Inteligencia Artificial. En concreto, se hace uso de métodos de aprendizaje supervisado, a partir de un set de entrenamiento la máquina es capaz de aprender y adaptarse a la solución. Dos tipos de ML supervisado son Árboles de decisión y las Redes Neuronales. Los Árboles de decisión son un modelo con estructura jerárquica compuesta por nodos y ramas. Las Redes Neuronales intentan emular el cerebro humano. Están compuestas de neuronas, las cuales cuando van aprendiendo van haciendo más fuertes o más débiles las conexiones entre las características.

Los modelos se realizan en lenguaje de programación Python y se utilizan librerías externas, como Pandas, Numpy o scikit-learn, librería para la IA.

Tanto los Árboles de decisión como las Redes Neuronales pueden ser de clasificación o de regresión. Además, de estos dos métodos, existen preprocesadores como escaladores, descomposiciones de matrices o seleccionadores de características. En concreto, uno de los preprocesadores de descomposición de matrices es el Análisis de Componentes Principales (PCA) (proyección de los datos según la cual los datos queden mejor representados en términos de mínimos cuadrados). La aplicación del PCA en los Árboles no tiene ningún efecto positivo, simplemente hace que los Árboles solo fallen. En cambio, aumentan el número de aciertos para las redes.

En general, las predicciones de riesgo de los Árboles de Decisión se parecen más a las estimaciones de riesgo verdaderas cuando los datos son los originales. Sin embargo, para las redes ocurre lo contrario. Se debe de preprocesar de alguna manera para obtener unas estimaciones aceptables. Las estimaciones de riesgo de la Red clasificadora son mejores que las de la red de regresión, siendo éstas no aceptables en ningún caso. En los Árboles de decisión ocurre lo contrario, aunque las predicciones del clasificador y regresor son muy buenas. Además, si se categorizan las predicciones entre "Riesgo" o "No Riesgo" el número de aciertos aumenta considerablemente, pero no se haría una validación correcta de los modelos de aprensizaje.

En el caso de entrenamiento con todas las características de los CDMS, se preprocesa los datos con *SelectKBest*, el cual se queda con las k columnas con mayores pesos. Las características que determina para todos los métodos son buenas, en muchas ocasiones coinciden con las mencionadas por la ESA. Ya que con el comando *SelectKBest* no se consiguen buenos resultados, se podría probar con otros métodos de selección de características como *SelectPercentile*, donde se selecciona aquellas con percentil más alto, o *VarianceThreshold*, el cual elimina funciones de baja variación.

Más adelante, como las Redes Neuronales de scikit no están preparadas para trabajr con datos tan amplíos, se puede probar otra librería de Python más centrada en Redes Neuronales como por ejemplo *Keras*. Además, dado que con los Árboles de decisión funcionan, se puede llegar a optimizar e intentar visualizar los Árboles utilizando *export_graphviz*.

Apéndice A
Mecánica Orbital

Apéndice para explicar términos relacionados con la Mecánica Órbital o con el espacio

A.1 Época

La época es el instante de tiempo que se usa como referencia para todos los datos tomados.

A.2 Órbitas

Una órbita de un cuerpo está determinada por un conjunto mínimo de datos, denominados los elementos orbitales, en una época.

Los elementos orbitales se pueden ver en la Figura A.1 y son los que siguen:

- Semieje Mayor (a): semieje mayor de la cónica que define la órbita.
- Excentricidad (e): Mide el grado de desviación con respecto a una circunferencia de la cónica de la órbita.
- Ascensión Recta del Nodo Ascendente (RAAN, Ω): ángulo entre el punto de Aries (punto de la eclíptica a partir del cual el Sol pasa del hemisferio sur celeste al hemisferio norte) y la línea de nodos (intersección entre el plano orbital y el de referencia), en sentido del nodo ascendente. Se mide en sentido contrario a las agujas del reloj.
- Argumento de periapsis (w): ángulo entre la linea de nodos y el vector excentricidad, medido en dirección del movimiento y en el plano de la órbita.
- Inclinación (i): ángulo entre l plano de referencia y el plano orbital, con el sentido indicado la línea de nodos.
- Ley horaria (θ): indica la posición de un cuerpo, en una época en concreto.

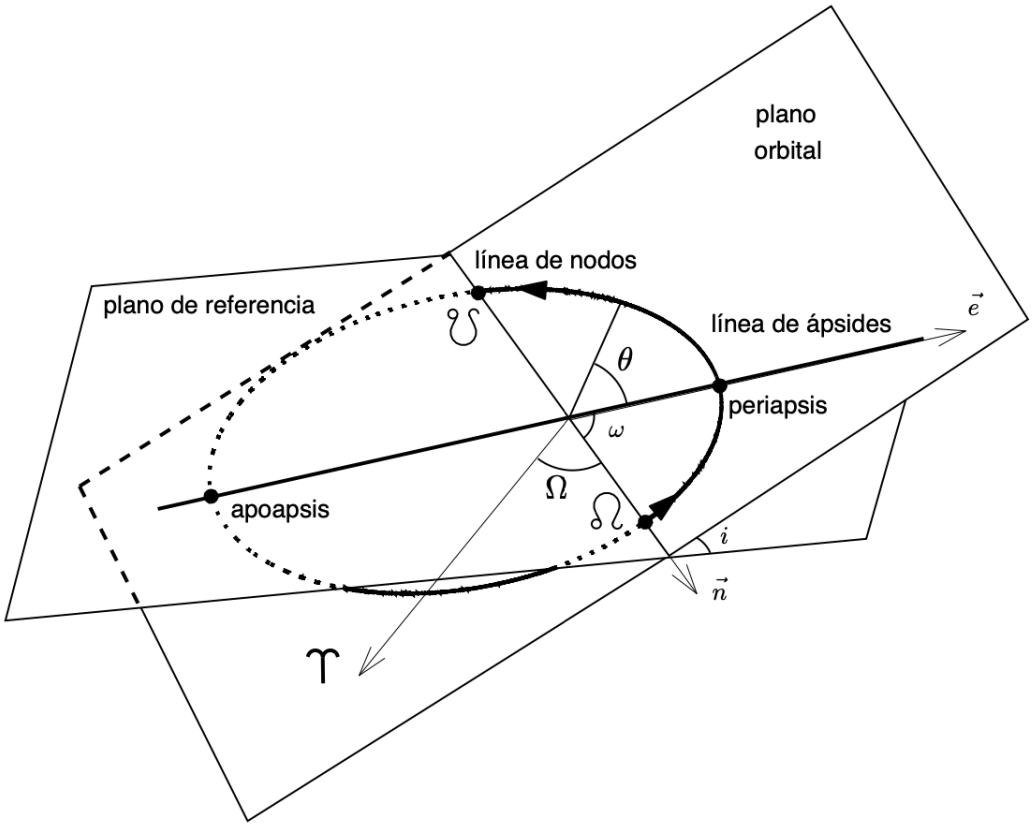


Figura A.1 Elementos orbitales.

A.3 Azimut y elevación

El sistema de referencia topocéntrico está centrado al observador ligado a la tierra. El eje *x* coincide con el Este, el *y* con Norte y el eje *z* es perpendicular al plano *xy* hacia arriba. El azimut es el ángulo entre el Norte y la dirección del observador, en el plano *xy*. La elevación es el ángulo, medido hacia arriba, entre la dirección del objeto y el plano *xy*.

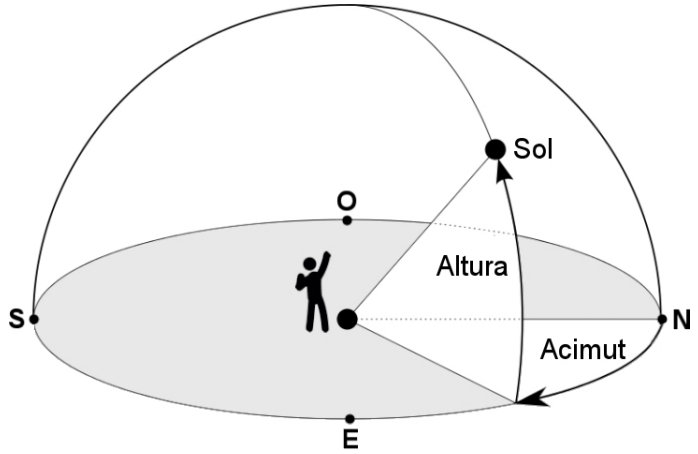


Figura A.2 Sistema topocéntrico.

A.4 Apogeo y perigeo

En una órbita elíptica, el apoapsis y el periapsis son los puntos de intersección órbita elíptica con el semieje mayor. El apoapsis es el punto más alejado del centro de la órbita y el periapsis es el punto más cercano.

Para las órbitas de los satélites. el apogeo es el punto más alejado de la Tierra y el perigeo el más cercano.

A.5 Emisiones de radio F10,7 cm

El flujo de radio solar a 10,7 cm (índice F10,7) es un indicador de actividad solar, medible desde la Tierra día a día. Las emisiones de radio se originan en lo alto de la cromosfera y en lo bajo de la corona de la atmósfera solar. Además, se trata de un largo registro a través de los años, por lo que informa sobre los climatología de los ciclos solares.

A.6 Índice geomagnético

El índice geomagnético cuantifica las fluctuaciones del campo magnético terrestre, en un punto en concreto, en un rango de 0 a 9. El índice AP es el índice de fluctuación K (alteraciones horizontales del campo magnético) transformado a un período de 24 horas.

A.7 Número de Wolf

El número de Wolf o de Zúrich mide el tamaño y el número de manchas solares. La idea, creada por Rudolf Wolf en Zurich, combina manchas solares y grupos, que compensan las variaciones para pequeñas variaciones solares.

Apéndice B
Estadísticas

B.1 Varianza

En la teoría de probabilidad se define la varianza de una variable aleatoria como

$$Var[f(x)] = E[f(x)^2] - [Ef(x)]^2 = \sum g(x_i)^2 p(x_i) - \sum g(x_i) 2 p(x_i)^2 \tag{B.1}$$

donde E[f(x)] es la esperanza matemática y $p(x_i)$ es la probabilidad.

B.2 Desviación estándar

La desviación estándar o típica es la raíz cuadrada de la varianza.

$$\sigma = \sqrt{Var[f(x)]} \tag{B.2}$$

B.3 Covarianza

Para una variable aleatoria bidimensional, la covarianza es el momento de segundo orden centrada en las media. Mide el grado de asociación entre las variables.

$$\sigma_{xy} = E[(X - \mu_x)(Y - \mu_y)] \tag{B.3}$$

donde μ es el momento central de grado 1.

B.4 Parámetros de regresión R^2

R^2 es el coeficiente de determinación. Se usa en modelos de predicción y determina la calidad del modelo. Para la regresión lineal tiene la siguiente expresión matemática:

$$R^2 = \frac{\sigma_{xy}^2}{\sigma_x^2 \sigma_y^2} \tag{B.4}$$

B.5 Error absoluto medio

Este error es la diferencia entre el valor predecido y el valor real en cada punto o instante de tiempo.

B.6 Error cuadrático medio

Para la regresión, es la estimación de la varianza del error, es decir, la suma residual de los cuadrados entre los grados de libertad.

B.7 Distancia de Mahalanobis

La distancia de Mahalanobis es una medida entre un punto de muestra y una distribución. Su utilidad radica en que es una forma de determinar la similitud entre dos variables aleatorias multidimensionales, teniendo en cuenta la correlación entre las variables aleatorias.

B.8 Estadística descriptiva univariante

Los datos univariantes son los que provienen de una única variable. En algunos casos, los datos pueden proceder de dos o más variables y, entonces, se usa la expresión bivariante (si se trata de dos variables) o multivariante (si se conside- ran más de dos).

Apéndice C
Anexo de Machine Learning

C.1 Términos

Validación cruzada

En scikit-learn, el *cv* determina la estrategia de división para validación cruzada, técnica de evaluación estadística de los resultados para un modelo. La validación cruzada divide el conjunto de datos en varios subconjuntos de entrenamiento y predicción y evalúa el rendimiento de cada subconjunto.

Tiempo de ajuste

El tiempo para ajustar el estimador en el conjunto de entrenamiento para cada división de cv (validación cruzada).

Tiempo de puntuación

El tiempo para puntuar el estimador en el conjunto de entrenamiento para cada división de cv.

C.2 Función de calidad del árbol de decisión clasificador

El árbol divide de forma recursiva el espacio entre las característica (columnas) de modo que los valores o etiquetas similares de las muestras se agrupen.

Siendo m el nodo representa por Q_m para n_m muestras, se tiene que para cada división candidata $\theta = (j,t_m)$ del atributo j los subconjuntos $Q_m^{izquierda}$ y $Q_m^{derecha}$. La calidad de cada candidato se calcula a través de la función de impureza o de pérdida $H()$, la cual depende del criterio a utilizar en cada caso. De esta forma, se tiene la siguiente función de calidad

$$G(Q_m,\theta) = \frac{n_m^{izquierda}}{n_m} H(Q_m^{izquierda}(\theta)) + \frac{n_m^{derecha}}{n_m} H(Q_m^{derecha}(\theta)) \tag{C.1}$$

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] Hervé Abdi and Lynne J. Williams, *Principal component analysis*, 7 2010, pp. 433–459.

[2] European Space Agency, *Collision avoidance challenge*, 2021.

[3] Ethem Alpaydin, *Introduction to machine learning*, third edition. ed., MIT Press, 2014.

[4] Howard D. Curtis, *Orbital mechanics for engineering students*, fourth edition ed., Elsevier, BH, Butterworth-Heinemann is an imprint of Elsevier, 2019.

[5] Rasit Abay Nils Einecke Sven Rebhan Jose Martinez-Heras Francesca Letizia Jan Siminski Klaus Merz Thomas Uriot Dario Izzo, Luís F. Sim{õ}es, *Spacecraft collision avoidance challenge: design and results of a machine learning competition*.

[6] ESA, *About space debris*.

[7] Fabian Pedregosa FABIANPEDREGOSA, Vincent Michel, Olivier Grisel OLIVIER-GRISEL, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, David Cournapeau, Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Alexandre Passos, Matthieu Brucher, Matthieu Perrot andÉdouardand, andÉdouard Duchesnay, and FRÉdouard Duchesnay EDOUARDDU-CHESNAY, *Scikit-learn: Machine learning in python gaël varoquaux bertrand thirion vincent dubourg alexandre passos pedregosa, varoquaux, gramfort et al. matthieu perrot*, 2011, pp. 2825–2830.

[8] National Centers for Environmental Information, *Wolf sunspo number*.

[9] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant, *Array programming with numpy*, Nature **585** (2020), 357–362.

[10] Simon Haykin, *Neural networks : a comprehensive foundation*, Macmillan College [etc.], 1994.

[11] J D Hunter, *Matplotlib: A 2d graphics environment*, Computing in Science Engineering **9** (2007), 90–95.

[12] IBM, *What is artificial intelligence?*

[13] IBM, *¿qué son las redes neuronales?*

[14] IBM., *Árboles de decisión*.

[15] MathWorks, *Distancia de mahalanobis*.

[16] John Mccarthy, *What is artificial intelligence?*, 2007.

[17] Wes Mckinney, *Data structures for statistical computing in python*, 2010.

[18] K Merz, B Bastida Virgili, V Braun, T Flohrer, Q Funke, H Krag, S Lemmens, and J Siminski, *Current collision avoidance service by esa's space debris office*.

[19] Klaus Merz, Jan Siminski, Benjamin Bastida Virgili, Vitali Braun, Sven Flegel, Tim Flohrer, Quirin Funke, Andre Horstmann, Stijn Lemmens, Francesca Letizia, Frazer Mclean, Silvia Sanvido, and Volker Schaus, *Esa's collision avoidance service: Current status and special cases*.

[20] NATIONAL OCEANIC and ATMOSPHERIC ADMINISTRATION, *F10.7 cm radio emissions*.

[21] NATIONAL OCEANIC and ATMOSPHERIC ADMINISTRATION, *Station k and a indices*.

[22] The pandas development team, *pandas-dev/pandas: Pandas*, February 2020.

[23] David. Paper, *Hands-on scikit-learn for machine learning applications data science fundamentals with python*, 1st ed. 2020. ed., Apress, 2020.

[24] Jorge Salas Plata and María Portillo, *P. ch. mahalanobis y las aplicaciones de su distancia estadística*, CULCyT: Cultura Científica y Tecnológica, ISSN 2007-0411, Nº. 27, 2008, pags. 13-20 **5** (2008).

[25] John E. Prussing, *Orbital mechanics*, 2nd ed. ed., Oxford University Press, 2013.

[26] V. K. Rohatgi, *An introduction to probability and statistics*, third edition. ed., Wiley, 2015.

[27] IEEE Electron Devices Society, Institute of Electrical, Electronics Engineers, and Vaigai College of Engineering, *Proceeding of the 2018 international conference on intelligent computing and control systems (iciccs) : June 14-15, 2018*.

[28] A M Turing, *Computing machinery and intelligence*, 1950, pp. 433–460.

[29] Jorge Santiago. Nolasco Valenzuela, *Python.*, RA-MA Editorial, 2018.

[30] Rafael Vazquez Valenzuela, *Apuntes de mecánica orbital y vehículos espaciales- gia*, 2021.

[31] Yagang Zhang, *Machine learning*, IntechOpen, 2010.