UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

COMPARATIVA DE MODELOS DE MACHINE LEARNING PARA LA ESTIMACIÓN DE PARÁMETROS DE INTERÉS EMPLEANDO DATOS DE LA EUROPEAN SOIL DATABASE

Autora:

Dña. Lidia Martínez Martínez

Tutor:

Dr. D. Jaime Gómez Gil

Valladolid, 28 de septiembre de 2021

TÍTULO:	**Comparativa de modelos de Machine Learning para la estimación de parámetros de interés empleando datos de la European Soil Database**
AUTORA:	**Dña. Lidia Martínez Martínez**
TUTOR:	**Dr. D. Jaime Gómez Gil**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Javier Manuel Aguiar Pérez**
VOCAL:	**Dra. Dña. Miriam Antón Rodríguez**
SECRETARIO:	**Dr. D. David González Ortega**

FECHA:	**28 de septiembre de 2021**
CALIFICACIÓN:	

Resumen

En este Trabajo Fin de Máster se han propuesto distintos métodos de *Machine Learning* para la predicción de características de suelo de difícil adquisición y de importancia en el ámbito de la agricultura. Esta predicción se realizará a partir de otras características cuya obtención es directa o más sencilla y con las que guardan una cierta correlación. Para ello, se ha utilizado la base de datos European Soil Database v2.0 y, como métodos de predicción, la regresión lineal, la regresión normal, la regresión de Poisson, la regresión de los k vecinos más cercanos, *Decision Tree* y *Random Forest*. En primer lugar, se ha realizado un preprocesado y análisis de los datos para, posteriormente, predecir las características seleccionadas con los métodos propuestos. Todo ello se ha desarrollado mediante el lenguaje de programación Python y las bibliotecas de scikit-learn para realizar las predicciones y Sweetviz para visualizar los datos. Las características que se escogieron como variables a estimar fueron la relación carbono-nitrógeno (C_N), el peso equivalente de carbonato de calcio (CACO3_TOT) y la retención de agua del suelo a capacidad de campo (WC_FC); mientras que las características que se emplearon para predecir estas variables fueron el pH, la materia orgánica, distintas clases de texturas y las bases intercambiables de calcio y potasio. Para evaluar el rendimiento de los métodos de predicción elegidos se calculó la raíz del error cuadrático medio (RMSE) y el coeficiente de correlación. Con ello, se ha podido comprobar que la combinación de características de baja correlación con la variable estimada y el empleo de modelos de predicción consiguen mejorar la correlación. Además, tras analizar los resultados obtenidos se ha podido concluir que *Random Forest* es el mejor método entre los evaluados para predecir las características.

Palabras clave:

Machine Learning, método de predicción, suelo, *Random Forest*, características, correlación, scikit-learn, Sweetviz.

Abstract

In this Master's Dissertation, different Machine Learning methods have been proposed for the prediction of soil characteristics that are difficult to acquire and important in the agricultural sector. This prediction will be obtained by combining other characteristics whose acquisition is direct or simpler and with some correlation with the predicted characteristic. To do so, the European Soil Database v2.0 has been employed, using the linear regression, normal regression, Poisson regression, k nearest neighbor regression, Decision Tree and Random Forest as prediction methods. In the first place, data was pre-processed and analysed in order to predict the selected characteristics with the proposed methods. This processing and analysis work has been developed using the Python programming language and the scikit-learn and Sweetviz libraries to make the predictions and the visualization tasks, respectively. The characteristics that were chosen as variables to be estimated were the carbon-nitrogen ratio (C_N), the calcium carbonate equivalent weight (CACO3_TOT) and the soil water retention at field capacity (WC_FC); while the characteristics that were used to predict these variables were pH, organic matter, different kinds of textures and exchangeable calcium and potassium bases. The root mean square error (RMSE) and the correlation coefficient were calculated to evaluate the performance of the chosen prediction methods. Finally, it was possible to improve the correlation of the resulting estimation by combining characteristics that are lowly correlated with the estimated variable using the evaluated prediction models. In addition, the analysis of the results obtained lead to the conclusion that Random Forest is the best method to predict the characteristics among those evaluated in this dissertation.

Keywords:

Machine Learning, prediction method, soil, Random Forest, characteristics, correlation, scikit-learn, Sweetviz.

Agradecimientos

Simplemente me gustaría agradecer a todos los que, de una forma u otra, me habéis acompañado durante estos dos años. Ha habido muchos momentos adversos en tan corto periodo de tiempo, pero aun así habéis permanecido a mi lado. Sé que, pase lo que pase, siempre voy a poder contar con vosotros.

Gracias de todo corazón.

Índice abreviado

Índice general

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1.1 Ámbito del proyecto

En la actualidad la gestión sostenible de los suelos es una necesidad puesto que, por un lado, son la base para la producción de alimentos y, por otro lado, son el principal depósito natural de carbono existente. Por tanto, la conservación de los suelos es importante ya que con ello se conseguiría aumentar y diversificar el cultivo de los alimentos y mitigar el cambio climático.

Para conseguir este propósito es fundamental conocer los parámetros que describen las propiedades del suelo. De entre todos ellos, algunos como el pH o las distintas clases de texturas son fácilmente medibles. Sin embargo, otros como la relación carbono-nitrógeno, el peso equivalente de carbonato de calcio o la retención de agua del suelo no tienen una adquisición sencilla. Esto se debe a que requieren de un análisis en el laboratorio que es tedioso o costoso por la instrumentación que se necesita para realizar la medida.

Por otra parte, el aprendizaje automático o *Machine Learning* está en boga hoy en día debido, entre otras cosas, a su capacidad para buscar relaciones en conjuntos de datos. Por ello, en este Trabajo Fin de Máster se ha decidido aplicar distintas técnicas de Machine *Learning* y análisis de datos para predecir características de suelos de difícil adquisición.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Máster es proponer métodos para la predicción de características de suelo de difícil adquisición y de interés en ámbitos como la agricultura. Esta predicción se realizará a partir de otras características más sencillas de medir y cuya correlación con las primeras no haga posible su predicción directa.

Adicionalmente, se han añadido los siguientes objetivos secundarios con los que complementar el objetivo principal:

- Realizar un análisis de los datos disponibles en la base de datos European Soil Database v2.0 para determinar las características de trabajo que se emplearán en las predicciones.
- Estudiar e implementar diferentes métodos de *Machine Learning* que permitan resolver problemas de predicción.
- Analizar y comparar los distintos métodos propuestos, así como algunas de las posibles configuraciones de las que disponen.

1.3 Fases y métodos

Con el fin de facilitar el desarrollo del proyecto se decidió organizar la realización del TFM en las siguientes fases:

- Búsqueda de una base de datos que contenga características de suelo.
- Análisis de la información adjunta a la base de datos.
- Estudio teórico de distintos métodos de *Machine Learning* para predecir datos.
- Aprendizaje del manejo de las librerías de Python que se van a utilizar para programar.
- Programación en Python del procesado y análisis de datos.
- Elección de las características que se aplicarán a los métodos de predicción propuestos.
- Obtención y discusión de los resultados.
- Identificación de las líneas futuras de mejora.
- Elaboración de la memoria del Trabajo Fin de Máster.

1.4 Organización de la memoria

La memoria de este Trabajo Fin de Máster se ha estructurado de la siguiente forma:

- En el Capítulo 1 se plantea el ámbito del proyecto, así como los objetivos que se pretenden alcanzar a lo largo de este TFM. También se enumeran las fases y métodos que se han seguido para conseguir lograr los objetivos propuestos y se muestra la organización de la memoria.
- El Capítulo 2 establece el marco teórico necesario para llevar a cabo el trabajo propuesto. En primer lugar, se introduce tanto la inteligencia artificial como el aprendizaje automático. Posteriormente se explica el funcionamiento de cada uno de los métodos de predicción implementados: regresión lineal, modelo lineal generalizado empleando la distribución normal y de Poisson, regresión de los k vecinos más cercanos, árbol de decisión y bosques aleatorios.
- En el Capítulo 3 se describen los materiales y métodos utilizados. Para empezar, se especifica qué base de datos se ha utilizado y cómo se ha obtenido. A continuación, se describen las bibliotecas de Python scikit-learn y Sweetviz ya que son las fundamentales para implementar los métodos de predicción propuestos y visualizar los datos. En último lugar se detalla el proceso que se ha llevado a cabo para realizar el análisis de los datos, así como las medidas elegidas para evaluar el rendimiento de los estimadores.

- El Capítulo 4 muestra los gráficos y resultados obtenidos tras realizar el análisis de la base de datos. Asimismo, para la estimación de las características escogidas para predecir se ha calculado el RMSE y la correlación alcanzada con cada uno de los métodos de *Machine Learning* implementados.
- Finalmente, en el Capítulo 5 se exponen las conclusiones obtenidas y las posibles líneas futuras que permitirían complementar y continuar con este trabajo.
- Adicionalmente, se han añadido dos anexos. El Anexo I incluye una tabla con las siglas de las características que aparecen en la base de datos y su significado, mientras que en el Anexo II se muestra el código desarrollado en el TFM.

Capítulo 2: Marco teórico

En el presente capítulo se realizará una introducción a la inteligencia artificial, así como a uno de los campos que la constituyen, el aprendizaje automático. A continuación, se explicará el funcionamiento de la regresión lineal, el modelo lineal generalizado, los k vecinos más cercanos (KNN), los árboles de decisión y los bosques aleatorios, que son las técnicas de aprendizaje supervisado que se han empleado en este Trabajo Fin de Máster.

2.1 Inteligencia artificial

La inteligencia artificial (*Artificial Intelligence*, AI) se refiere a la ciencia y la ingeniería que permite a las máquinas simular el cerebro humano para aprender y pensar. Además, se ocupa de desarrollar sistemas informáticos que puedan almacenar conocimientos y utilizarlos de forma eficaz para ayudar a resolver problemas y realizar tareas. En los últimos años, la AI se ha desarrollado rápidamente y ha cambiado el estilo de vida de las personas [1], [2].

La inteligencia artificial es un método de ingeniería que se encarga de simular las estructuras y los principios operativos del cerebro humano. Por tanto, el *software* de inteligencia artificial serían los programas de ordenador con capacidad para realizar operaciones análogas al aprendizaje y la toma de decisiones en los seres humanos. En la década de 1950 fue propuesta por primera vez por Jon McCarty. Una década más tarde, Rosenblatt desarrolló un modelo que era capaz de procesar muestras de datos experimentales. En la década de 1980, la investigación de algoritmos basados en redes neuronales artificiales se desarrolló rápidamente. Ya en el siglo XXI, la inteligencia artificial ha logrado un avance revolucionario gracias al aprendizaje profundo y al desarrollo del Internet móvil, que trajo consigo más escenarios de aplicaciones de AI. Además, el progreso de la AI se ha convertido en una estrategia de desarrollo muy importante para países de todo el mundo, ya que mejora la competitividad nacional y mantiene la seguridad. El uso extensivo de sistemas expertos ha mejorado la eficiencia de la industria y con ello ha conseguido disminuir en gran medida sus costes [1], [3], [4].

En los últimos años se han desarrollado multitud de aplicaciones gracias a una mayor potencia informática, a los algoritmos complejos y al crecimiento exponencial de los datos generados por humanos y máquinas. Algunos ejemplos son la previsión de ventas, el control de procesos, el reconocimiento de imágenes, la traducción, el desarrollo de automóviles sin conductor, la educación y la medicina. Aunque una de las aplicaciones de AI más destacadas es el reconocimiento de voz, donde los modelos desarrollados son capaces de responder correctamente a todo tipo de solicitudes de los usuarios [1], [3].

Un sistema de inteligencia es aquel que comprende el espectro completo de la AI. Es capaz de adquirir conocimiento a través de los datos proporcionados por las máquinas y los ordenadores y utilizar esta información para automatizar y acelerar tareas que antes solo era

posible realizarlas por humanos. Además, son capaces de adaptar su comportamiento analizando cómo el medio ambiente se ve afectado por sus acciones previas. La inteligencia artificial está formada por tres pilares principales: estructura de dominio, generación de datos y aprendizaje automático. Para la inteligencia artificial es vital tener una estructura de tareas bien definida con la que poder diseñar un sistema, así como un banco de datos masivo para poner en marcha el sistema. Además, es necesaria una estrategia con la que seguir generando datos para que el sistema pueda responder y aprender. Por último, también se requieren de rutinas de aprendizaje automático capaces de detectar patrones y de hacer predicciones a partir de los datos no estructurados [1], [4].

El primer pilar que figura en el sistema de inteligencia de la AI es la estructura del dominio. Esta estructura facilita dividir un problema complejo en tareas compuestas que se pueden resolver mediante el aprendizaje automático. El segundo pilar es la generación de datos. Una estrategia activa de recopilación de datos es vital para mantener un flujo constante de nueva información útil hacia los algoritmos de aprendizaje compuestos. En la mayoría de las aplicaciones de AI hay dos clases generales de datos: valores de datos de tamaño fijo que se pueden usar para entrenar los modelos y datos que el sistema genera activamente a medida que experimenta y mejora el rendimiento. El último pilar del sistema de inteligencia es el aprendizaje automático que es una herramienta para el reconocimiento de patrones. Los sistemas de AI dividen los problemas complejos en múltiples tareas de predicción simples y para resolverlas se valen de los algoritmos de aprendizaje automático. A partir de la estructura definida por el conocimiento del dominio y los datos adquiridos para entrenar, los algoritmos de aprendizaje automático pueden predecir un nuevo dato a partir de los datos anteriores [1].

El análisis de grandes conjuntos de datos o datos no estructurados a menudo se realiza mediante el aprendizaje automático. Los conjuntos de datos actuales pueden provenir de diversas fuentes como son Internet (publicaciones en redes sociales, reseñas de productos, tendencias de búsqueda, etc.), datos generados por sensores (imágenes de satélite, tráfico peatonal y de automóviles, etc.), así como datos generados por procesos comerciales (transacciones comerciales, tarjetas de crédito, etc.). A diferencia de los conjuntos de datos tradicionales, los nuevos conjuntos de datos se caracterizan por ser más grandes en volumen, velocidad y variabilidad, por lo que se requiere de un análisis antes de que estos datos puedan usarse. Esto incluye un análisis de los distintos tipos de datos y la evaluación de su relevancia. Para ello se emplean técnicas de ML como el aprendizaje supervisado y no supervisado, así como técnicas de aprendizaje profundo y reforzado para analizar datos no estructurados [1].

En el ámbito de la educación la inteligencia artificial permite la selección del contenido más apropiado para cada alumno, de acuerdo con sus habilidades y requisitos individuales. Esto supone un apoyo más eficiente y centrado en el alumno. Por otro lado, en el ámbito de la medicina se emplea tanto con fines de diagnóstico como para seguimiento de enfermedades y análisis de supervivencia. Los tratamientos personalizados se ven reforzados por la creciente aplicación de la AI en el campo de la medicina de precisión. Además, también

se emplea para el análisis de imágenes en diversos campos como radiología, oftalmología, dermatología o cardiología. Con ello se consigue una mayor ayuda para el diagnóstico de enfermedades como pueden ser el cáncer de piel o problemas cardíacos [1].

2.1.1 Aprendizaje automático

El aprendizaje automático (*Machine Learning*, ML) es uno de los campos que constituyen la inteligencia artificial. ML combina múltiples disciplinas como son la estadística, la lógica, la robótica, la informática, la inteligencia computacional, el reconocimiento de patrones y la minería de datos entre otros [2], [5].

El aprendizaje automático ha estado presente en la inteligencia artificial desde sus inicios en la década de 1950. Posteriormente, en 1980 el ML se estableció como un campo separado de la AI y en 1983 Michalski *et al.* publicó el primer libro sobre aprendizaje automático. Desde entonces, esta ciencia ha continuado creciendo y se han creado varios subcampos [5].

El ML es un método de análisis de datos que se encarga de automatizar la construcción de modelos analíticos. Con el rápido desarrollo de la tecnología informática, el aprendizaje automático ha ganado popularidad en la ciencia de datos, ya que puede manejar fácilmente conjuntos de datos grandes y complejos. En consecuencia, sus aplicaciones son numerosas y en diversas áreas como son el reconocimiento de imágenes y de voz, la predicción del tráfico, las recomendaciones de productos, los vehículos autónomos, la detección de correo electrónico no deseado, los asistentes personales virtuales y las fábricas inteligentes, entre otras muchas [6].

El aprendizaje automático es una disciplina científica que se encarga del desarrollo de capacidades de aprendizaje en sistemas informáticos. Se basa en la idea de que los sistemas o máquinas pueden identificar patrones por sí mismos, aprender de las muestras de datos y tomar decisiones automáticamente. Un sistema informático aprende si su función o conocimientos mejoran debido a la experiencia o si es capaz de adaptarse a un entorno cambiante. La experiencia puede ser debida a que el sistema aprende por sí mimo, pero también se puede proporcionar desde el exterior en forma de datos de los que el sistema aprende [5].

El aprendizaje automático también es capaz de reemplazar las tediosas tareas manuales, por lo que los tiempos de procesamiento y los costes económicos se reducen. Además, su tasa de progreso tiene un crecimiento exponencial, mientras que competidores como los algoritmos de búsqueda y planificación se desarrollan a una velocidad lineal. Los resultados del ML se encuentran en forma de modelos o funciones que representan el conocimiento adquirido y son lo que se utilizan con mayor frecuencia para realizar predicciones ante situaciones desconocidas [2], [5].

Los métodos de aprendizaje automático se pueden clasificar según el uso que se haga del maestro, de las formas en el que el sistema aprende, así como del modo de adquirir conocimiento y los datos empleados. Por lo general, los métodos de ML se clasifican como aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo [2], [5]. En la Figura 1 se puede un esquema en el que se muestra esta clasificación de los métodos de aprendizaje automático.

En el aprendizaje supervisado los datos que se proporcionan para entrenar constan tanto de variables de entrada (independientes) como de variables de salida (dependientes). Debido a esto también se conoce como aprendizaje con maestro. Dada una muestra de par de conjuntos de datos de entrada-salida, el fin es encontrar una función que asigne las entradas de ese tipo a salidas de ese tipo minimizando los errores tanto como sea posible. Es decir, el objetivo sería establecer una relación entre dos conjuntos de datos y utilizar uno de ellos para pronosticar el otro. Suelen ser modelos de regresión mejorados para adaptarse a regímenes cambiantes, datos atípicos y variables correlacionadas. Este tipo de aprendizaje se utiliza, por ejemplo, para el reconocimiento de letras y dígitos escritos a mano y la predicción de índices bursátiles. Los dos tipos principales de aprendizaje supervisado son la clasificación y la regresión. En el aprendizaje por clasificación los datos de salida son discretos, mientras que en aprendizaje por regresión son continuos [1], [5], [7], [8].

En el aprendizaje no supervisado el objetivo es identificar una estructura subyacente en los datos proporcionados como entrada, ya que estos no tienen una salida definida explícitamente. Por ello, también se conoce como aprendizaje sin maestro. Este tipo de aprendizaje se utiliza, por ejemplo, en problemas de segmentación tanto de imágenes como de texto y para detectar novedades en el control de procesos. Los dos tipos principales de aprendizaje no supervisado son el agrupamiento (*clustering*) y la reducción de la dimensionalidad. El agrupamiento se encarga de asociar los datos de entrada a diferentes grupos. Dado un número fijo de grupos, el fin es encontrar un agrupamiento de tal manera que las similitudes de los objetos en un grupo sean mucho mayores que las similitudes entre objetos de distintos grupos. De esta forma, se ha identificado la estructura para que todo el grupo pueda ser representado por un punto de datos representativo. En cambio, la reducción de la dimensionalidad trata de encontrar aquellas variables de entrada que conservan la mayor parte de la información de los datos. Esta técnica es especialmente importante puesto que el uso de más dimensiones que las estrictamente necesarias conlleva un aumento del especio necesario para almacenar los datos. Además, la velocidad de los algoritmos que utilizan los datos depende de la dimensión de los vectores, por lo que una reducción de la dimensión supone una disminución del tiempo de cálculo [5], [7], [8].

En el aprendizaje por refuerzo el objetivo es encontrar una secuencia de operaciones (camino) que conduzcan a una solución que maximice la “recompensa” a lo largo del tiempo. A los caminos que parecen beneficiosos se les asigna una puntuación de recompensa positiva, mientras que a los caminos que se consideran perjudiciales se les establece una puntuación

de recompensa negativa. De esta forma, el algoritmo tratará de encontrar el camino con la recompensa general más alta intentando evitar, en la medida de lo posible, aquellos con recompensa negativa. Uno de los mayores desafíos de este tipo de aprendizaje es encontrar un equilibrio entre exploración y explotación. Esto se debe a que para maximizar la recompensa el algoritmo debe elegir aquellas acciones que se han probado en el pasado y que produzcan caminos con recompensas positivas. Es decir, por un lado, debe explotar su conocimiento actual. En cambio, por otro lado, también tiene que explorar nuevas acciones que no se hayan probado en el pasado ya que pueden derivar en caminos con mejores recompensas [5], [7], [8].

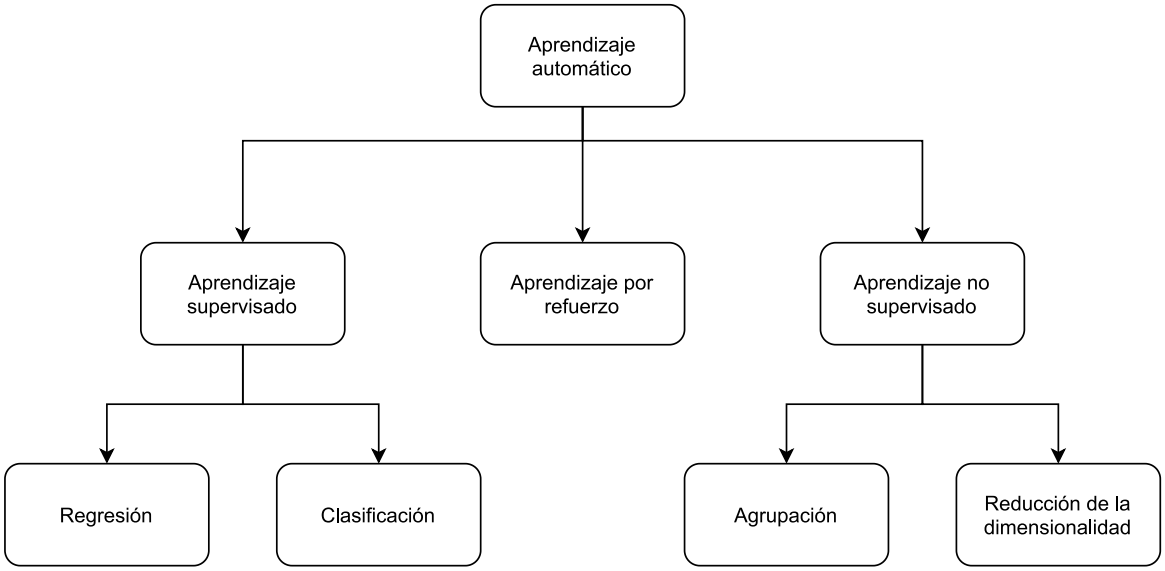


Figura 1: Métodos de aprendizaje automático.

Para construir un modelo ML en primer lugar hay que determinar los factores que influyen en el problema que se quiere solventar, así como las metas que se esperan alcanzar. A continuación, es necesario seguir los siguientes pasos: recopilar el conjunto de datos y preprocesarlos, construir el modelo ML y entrenarlo, evaluar e implementar el modelo y, por último, hacer predicciones en secuencia [2].

Uno de los principales problemas del aprendizaje automático es la interpretabilidad, ya que se suele utilizar como una técnica de "caja negra" mediante la cual la máquina aprende a desarrollar algoritmos predictivos a partir de datos. Es decir, con ML se obtienen resultados pero muchas veces no se comprende cómo se ha llegado a ellos puesto que el algoritmo predictivo desarrollado es demasiado complicado. Otro de los problemas del ML es que los métodos de aprendizaje no están orientados a la cuantificación de la incertidumbre. Es decir, se emplean datos de entrenamiento para maximizar la capacidad predictiva de funciones muy flexibles, tratando de encontrar vectores únicos de parámetros óptimos en lugar de distribuciones de probabilidad posteriores [7].

Aunque hay muchos ejemplos de aprendizaje supervisado, en las siguientes secciones únicamente se van a definir aquellos que se han empleado en este trabajo, que son la regresión normal, la regresión de Poisson, la regresión linear, los *k* vecinos más cercanos, los árboles de decisión y los bosques aleatorios.

2.2 Técnicas de estimación empleadas

Para desarrollar el sistema que se ha propuesto en este TFM se han utilizado las técnicas de estimación de aprendizaje supervisado de regresión lineal, modelo lineal generalizado, k vecinos más cercanos (KNN), árboles de decisión y bosques aleatorios. Por ello, son las que se detallan más en profundidad a continuación.

2.2.1 Análisis de regresión: Regresión lineal

El modelado predictivo trata de encontrar aquellas reglas que puedan predecir los valores de una o más variables en un conjunto de datos (salidas) a partir de los valores de otras variables del conjunto de datos de partida (entradas). Los algoritmos que se han desarrollado con diversas técnicas de modelado predictivo surgen a raíz de la investigación en varias disciplinas como son la estadística, el reconocimiento de patrones y el aprendizaje automático entre otras. Algunas de estas técnicas son los árboles de decisión, las redes neuronales, las máquinas de vector de soporte, los vecinos más cercanos. Además, cabe destacar que una de las técnicas más empleadas para realizar el modelado predictivo es el análisis de regresión [9].

El análisis de regresión es una técnica que permite estudiar y medir la relación entre variables. A partir de los datos registrados en una muestra, el análisis de regresión trata de predecir o de buscar una estimación mediante una relación matemática entre dos o más variables. Por tanto, el objetivo es predecir/estimar el valor de una variable (salida) en función de una o más variables de partida (entradas). El hecho de que el uso de los modelos de regresión sea muy popular puede ser debido tanto a la fácil interpretación de los parámetros del modelo como a su sencillo manejo [9], [10].

Se dice que el análisis de regresión es lineal cuando la relación entre las variables dependientes e independientes es lineal y su objetivo es, por un lado, estimar la función de regresión en relación con el modelo elegido y, por otro lado, probar la fiabilidad de las estimaciones obtenidas [10].

Además, en una regresión lineal, la variable que se quiere predecir es la dependiente y se suele denotar con Y. En cambio, las variables con las que se estimaría el valor de Y serían las independientes y se denotan con X. En el caso de que Y únicamente dependa de una variable independiente, el análisis de regresión es simple. Por el contrario, si Y depende de dos o más variables independientes, el análisis de regresión es múltiple [10].

En el caso estudiado en este TFM se ha tratado de estimar una variable a partir de dos o más variables independientes, por lo que la técnica empleada ha sido la de regresión lineal múltiple. Un modelo de regresión múltiple puede escribirse como [9]:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p x_p + \varepsilon$$

Donde Y es la variable dependiente (salida), X_i las variables independientes (entradas) siendo con $i = (1, 2 \dots, p)$, ε el término de error aleatorio no observable y β_i los parámetros de regresión siendo $i = (0, 1, 2 \dots, p)$. El principal problema del análisis de regresión consiste en estimar los parámetros β_i [9].

Roger Joseph Boscovich fue el pionero en el análisis de regresión lineal, ya que en torno a 1755-1757 encontró un método para determinar los coeficientes de una línea de regresión. Estableció que para obtener una línea que pase lo más cerca posible de todas las observaciones, se deben de cumplir dos condiciones [10]:

1. La suma de las desviaciones tiene que ser nula.
2. La suma de los valores absolutos de las desviaciones debe ser mínima.

Posteriormente, en 1805, Adrien Marie Legendre publicó el método de los mínimos cuadrados que consiste en minimizar la suma de los residuos al cuadrado, siendo los residuos las desviaciones al cuadrado entre los valores estimados y observados. A partir de entonces el método de los mínimos cuadrados ha sido el más utilizado para estimar los parámetros β_i [10].

2.2.2 Modelo lineal generalizado

El modelo lineal generalizado (*Generalized Linear Model*, GLM) es una ampliación del modelo de regresión lineal, puesto que este último únicamente asume que la variable de salida Y (variable dependiente) es igual a una combinación lineal. El modelo de regresión lineal puede escribirse como [11]:

$$E(Y|\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{\beta} \quad \text{ó} \quad Y = \boldsymbol{X}^T\boldsymbol{\beta} + \varepsilon$$

Sin embargo, hay situaciones en las que no se cumple esta condición como cuando los datos no tienen una distribución normal, por ello, el GLM asume que la distribución de los datos pertenece a un miembro de la familia exponencial. Además, permite que el modelo de regresión lineal se relacione con la variable de salida Y mediante una función de enlace [12].

El modelo lineal generalizado surge en 1972 a partir de la investigación de Nelder y Wedderburn, quienes unificaron la teoría del modelado estadístico y, en particular, los modelos de regresión. Por un lado, expusieron que los modelos de regresión lineal más comunes de la estadística clásica pertenecían a una misma familia y, por tanto, podían tratarse de la misma manera. Por otro lado, también mostraron que las estimaciones de máxima verosimilitud para estos modelos se podían obtener empleando un mismo algoritmo llamado mínimos cuadrados ponderados iterativamente [13].

El GLM está formado por los siguientes componentes [14], [15]:

- Componente aleatorio: la salida Y es un vector de n valores aleatorios de la forma $(Y_1, \dots, Y_n)$
- Componente sistemático: es la estructura lineal para el modelo de regresión ($\eta = \boldsymbol{X}^T\boldsymbol{\beta}$) donde $X^T = (1, X_{i1}, X_{i2}, \dots, X_{in})^T$ con $i = 1, \dots, m^T$ representa las variables independientes. Describe cómo la ubicación de la distribución de la respuesta cambia con las variables independientes de entrada.
- Función de enlace (G): función monótona y diferenciable que enlaza el componente aleatorio con el sistemático, relacionando la media (μ) de la variable dependiente Y con la estructura lineal:

  $$\eta = G(\mu) \quad E(Y|\boldsymbol{X}) = G^{-1}(\boldsymbol{X}^T\boldsymbol{\beta})$$

  Donde los coeficientes de regresión $\boldsymbol{\beta}$ representan el vector de parámetros que se quiere estimar.

Las distribuciones de la familia exponencial más comunes empleadas en los GLM son la normal, la binomial, la de Poisson y las funciones gamma [15]. En este TFM únicamente se han empleado la distribución normal y la de Poisson puesto que el rango de la salida Y era el que más se ajustaba a los casos a estudiar, siendo en el caso de la distribución normal $(-\infty, \infty)$ y en la distribución de Poisson $[0, \infty)$.

En la distribución normal la variable aleatoria X tiene una función de densidad de la forma [10]:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \qquad (\sigma > 0)$$

La distribución normal es una distribución de probabilidad continua con media μ y varianza σ^2. Su origen data de 1733 con De Moivre, quien fue el primero en obtener la distribución normal como una aproximación de la distribución binomial. Posteriormente, en 1774, Laplace obtuvo la distribución normal como una aproximación de la distribución hipergeométrica. Más tarde, Gauss con sus trabajos en 1809 y 1816 estableció técnicas basadas en la distribución normal que, durante el siglo XIX, se convirtieron en métodos estándar [10].

Por el contrario, la distribución de Poisson es una distribución de probabilidad discreta. La variable aleatoria X se corresponde con el número de elementos observados por unidad de tiempo o espacio y su función de probabilidad viene dada por [10]:

$$P(X = x) = \frac{\exp(-\lambda) \cdot \lambda^x}{x!}$$

La distribución de Poisson se suele emplear para describir modelos de probabilidad para un número de eventos discretos aleatorios que ocurren durante un periodo fijo de tiempo o espacio [10].

2.2.3 KNN

El algoritmo KNN (*K-Nearest Neighbor*) es un tipo de algoritmo no paramétrico de aprendizaje automático supervisado que se puede emplear para problemas predictivos de regresión y clasificación. Sirve para predecir tanto variables categóricas como numéricas con naturaleza no paramétrica y el único parámetro que hay que definir es el número de vecinos más cercanos. Debido a su simplicidad y eficacia, el algoritmo KNN es ampliamente utilizado en diversos campos como pueden ser la minería de datos, el reconocimiento de patrones, la selección de características o la detección de valores atípicos [16], [17].

Una característica del algoritmo KNN es que clasifica los datos sin procesar por lo que sería equivalente a realizar un preprocesamiento de los datos. Además, al tratarse de un método de clasificación no paramétrico, no necesita un proceso de entrenamiento por lo que se reducirían en gran medida los costes informáticos y así como el tiempo de entrenamiento. Esto quiere decir que no requiere de conocimientos previos sobre las propiedades estadísticas de las instancias de entrenamiento y, por tanto, puede clasificar directamente el nuevo dato en base a la información proporcionada por el conjunto de entrenamiento. Esto supone una clara ventaja respecto al enfoque de aprendizaje profundo (*deep learning*) ya que este tiene una complicada estructura de red de aprendizaje interna que hace que aumente el coste del tiempo [16], [18].

Su funcionamiento se basa en encontrar los k puntos de datos más cercanos en el conjunto de entrenamiento cuando se proporciona un nuevo punto. Si estos k vecinos más cercanos pertenecen todos a la misma categoría, se puede deducir que el nuevo punto también posee sus mismas características y atributos. Un ejemplo de funcionamiento del algoritmo KNN se puede ver en la Figura 2. Para realizar este proceso, el algoritmo KNN calcula la distancia entre los conjuntos de entrenamiento y prueba, el método más utilizado para ello es la métrica Euclidiana que se define como [18], [19]:

$$dist(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Donde p y q son puntos del espacio n-dimensional.

Aunque el algoritmo KNN tiene muchas ventajas, también presenta algunos problemas críticos. Por un lado, la regla de decisión que emplea para clasificar es muy simple ya que ignora las diferencias en la capacidad de clasificación de los vecinos que no son los k más cercanos. Esto significa que dentro del grupo de los k vecinos más cercanos solo los que pertenecen a la clase mayoritaria se utilizan para determinar el resultado de la decisión de

clasificación de la consulta, independientemente del papel del resto de vecinos más cercanos. Si los vecinos de la clase mayoritaria son dominantes es razonable emplear este método, pero si la diferencia entre el número de vecinos de las dos clases mayoritarias no es significativa no es razonable descartar por completo a los vecinos de la segunda clase dominante. Además, este método de decisión únicamente usa la función de distancia para medir la similitud entre el nuevo dato y los de entrenamiento, ignorando por completo la distribución espacial. Por ejemplo, si la distribución espacial de los vecinos de la segunda clase mayoritaria está más relacionada con el nuevo dato, sería más fiable asignar el nuevo dato a la segunda clase mayoritaria en lugar de a la primera [16].

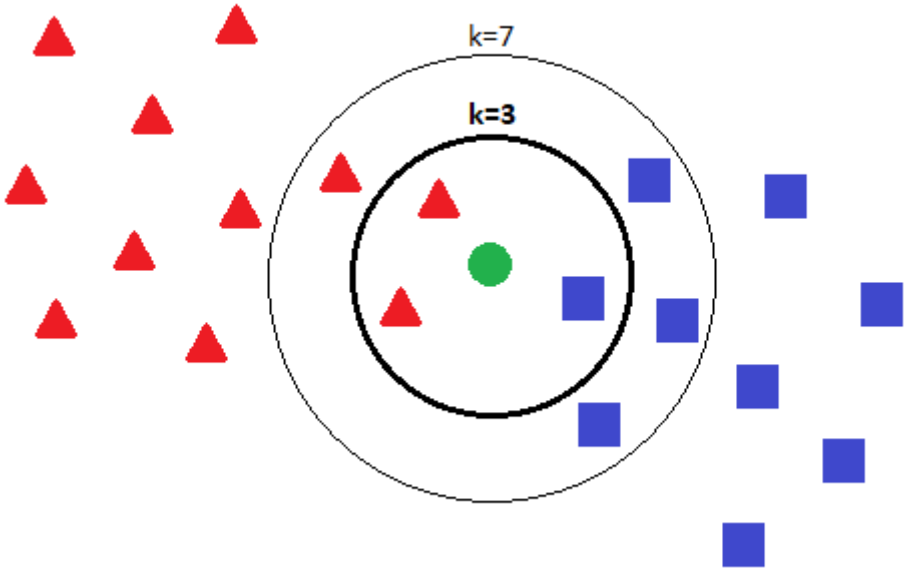


Figura 2: Ejemplo de funcionamiento del algoritmo KNN. Los triángulos rojos y los cuadrados azules son las muestras de dos clases distintas del conjunto de entrenamiento y el nuevo dato que se quiere clasificar es el círculo verde. Para k=3 la clase asignada al círculo verde será la de los triángulos rojos puesto que dentro del círculo interno hay 2 triángulos frente a 1 cuadrado. Sin embargo, para k=7 el círculo externo contiene 4 cuadrados y 3 triángulos por lo que el círculo verde se clasificará como cuadrado azul.

Por otro lado, el algoritmo es sensible a los valores de k. Esto quiere decir que, si k es demasiado pequeño, la precisión de la clasificación disminuirá ya que hay pocos datos con los que comparar. En cambio, si k es demasiado grande y el nuevo dato pertenece a la clase con menos instancias de entrenamiento, se seleccionarán los datos pertenecientes a la clase mayoritaria como los vecinos más cercanos. Esto provocará un peor rendimiento en la clasificación porque la clasificación se haría de acuerdo con la mayoría y no a la similitud de los datos. Además, este valor de k es fijo y único para determinar la clase a la pertenecen todos los nuevos datos. Sin embargo, como la distribución de los datos de entrenamiento es diferente respecto a la ubicación de cada nuevo dato, el número de vecinos más cercanos necesarios para clasificar de forma correcta cada nuevo dato también debería de ser distinto. Es decir, se necesitan diferentes valores de k para clasificar un dato ubicado en el centro de una clase de datos de entrenamiento que para un dato localizado en la superposición entre varias clases [16], [19].

2.2.4 Árboles de decisión

Los árboles de decisión (*Decision Trees*) se clasifican dentro de los métodos de aprendizaje supervisado no paramétrico y se pueden emplear tanto para clasificar como para realizar regresiones. La finalidad es crear un modelo que sea capaz de predecir el valor de una variable objetivo a través de reglas de decisión simples inferidas de las características de los datos [20].

La construcción del árbol de decisión se basa en una partición recursiva que es binaria, la cual es un proceso iterativo que se encarga de dividir los datos aplicando una serie de reglas simples. Inicialmente, todas las muestras que se han seleccionado para el entrenamiento se emplean para determinar la estructura del árbol. A continuación, el algoritmo divide los datos utilizando todas las posibles divisiones binarias y escoge aquella que al dividir los datos minimiza la suma de las desviaciones cuadradas de la media en las nuevas ramas. El proceso de división se sigue aplicando a cada rama creada hasta que cada nodo alcanza un tamaño de nodo mínimo y se convierte en un nodo terminal. Este tamaño mínimo lo especifica el usuario a través del número de muestras de entrenamiento en el nodo [21].

Cuanto más profundo es el árbol, más complejas son las reglas de decisión y más se ajusta el modelo. Sin embargo, cuando el árbol alcanza la estructura completa final, ha podido sufrir un ajuste excesivo (*overfitting*) debido a que se construye a partir de muestras de entrenamiento. Esto puede provocar una menor capacidad de generalización ya que este problema puede deteriorar la precisión de clasificación del árbol cuando se aplica a datos nuevos. Para solucionarlo se emplea un proceso denominado poda, en el que se utiliza un conjunto de datos de validación junto con un factor de complejidad de costos especificado por el usuario. Al realizar la poda se consigue minimizar la suma de la varianza de la variable de salida en los datos de validación y el producto de nodos terminales junto con el factor de complejidad del costo, que representa el costo de complejidad por nodo. Otros mecanismos para evitar este problema son el establecimiento del número mínimo de muestras necesarias en un nodo de la rama o de la profundidad máxima del árbol [20], [21].

Una de las principales ventajas sobre otras técnicas de modelado es que genera un modelo que representa reglas o declaraciones lógicas que se pueden interpretar y entender de manera sencilla. Esto se debe a que los árboles de decisión son visuales y ello supone que los resultados proporcionen una información clara sobre qué factores son importantes a la hora de hacer la predicción. Además, esta técnica se puede emplear tanto para variables continuas como para variables discretas o categóricas, aunque es más adecuada para predecir resultados categóricos. Los datos tampoco requieren una normalización o la eliminación de valores en blanco. También son capaces de manejar problemas con múltiples salidas [9], [20].

Por otro lado, una de las desventajas de este tipo de regresión es que no funciona bien para datos no lineales y es susceptible a datos ruidosos. Además, no es apropiada a la hora de predecir datos continuos a menos que se disponga de tendencias visibles y patrones

secuenciales. No son buenos para extrapolar datos puesto que las predicciones son constantes por partes, es decir, no son uniformes ni continuas. Por último, cabe destacar que si se producen pequeñas variaciones en los datos se pueden generar árboles completamente diferentes, por lo que pueden ser inestables. Para mitigar este problema hay que emplear los árboles de decisión dentro de un grupo de datos en lugar de hacerlo con todo el conjunto de datos disponibles [9], [20].

2.2.5 Bosques aleatorios

Un bosque aleatorio (*Random Forest*) está formado por un conjunto de clasificadores de árboles de decisión aleatorios que hace predicciones mediante una combinación de las predicciones de los árboles individuales. El bosque aleatorio se puede emplear para hacer predicciones sobre atributos de destino tanto nominales (clasificación) como numéricos (regresión). Además, es uno de los modelos predictivos que ofrece un mejor rendimiento, por lo que su uso se ha extendido por todos los ámbitos de investigación. Se han empleado en una amplia variedad de aplicaciones en disciplinas que van desde las ciencias puras hasta las humanidades [6], [22], [23].

El término de bosque aleatorio fue introducido por Breiman en 2001 y se refiere a un conjunto de árboles de decisión en los que cada árbol se construye utilizando un proceso aleatorio. Se trata de un enfoque no lineal cuyo objetivo es mejorar la precisión a través del promediado de múltiples árboles de decisión en el que cada uno de ellos sigue los siguientes pasos. En primer lugar, se emplea una muestra de arranque del conjunto de entrenamiento para construir cada árbol. A continuación, para encontrar la mejor división en los árboles se utiliza un único subconjunto aleatorio de variables en cada nodo interno [6], [23].

El bosque aleatorio es una evolución del modelo de aprendizaje por conjuntos Bagging y su objetivo radica en reducir la varianza de un modelo estadístico. Para ello, simula la variabilidad de los datos mediante la extracción de muestras de un único conjunto de entrenamiento y añade predicciones en un nuevo registro. De esta forma es capaz de mejorar las predicciones de muchos métodos supervisados como los árboles de decisión. Esto se debe a que los árboles se cultivan en profundidad sin emplear una fase de poda, por lo que el resultado que se obtiene es un conjunto de clasificadores caracterizados por una alta varianza y un sesgo bajo. En cambio, el bosque aleatorio, al ser un procedimiento por conjuntos reduce la varianza calculando las predicciones como el promedio de los árboles generados. Además, al ser una evolución de Bagging, el bosque aleatorio puede obtener árboles más diferentes y no relacionados [6].

La principal característica de los modelos de bosques aleatorios es la alta precisión que se obtiene en las predicciones. Esto se debe a que se emplean enfoques no paramétricos basados en algoritmos iterativos para realizar la predicción. Sin embargo, estos algoritmos también son responsables de la mayor desventaja de los bosques aleatorios, ya que generan los modelos conocidos como “caja negra”. Esto significa que los modelos no son interpretables

a través de parámetros y formas funcionales como, en cambio, sí lo es el análisis de regresión clásico [6].

A pesar de esta desventaja los modelos de bosques aleatorios presentan otras muchas ventajas. En primer lugar, cabe destacar que el rendimiento de estos modelos converge a medida que se agregan más árboles, por lo que no hay riesgo de sobreajustar los datos de entrenamiento. También se ha comprobado que estos modelos funcionan muy bien incluso cuando se utiliza una muestra muy pequeña de los atributos en cada nodo. Además, aunque se emplee una gran cantidad de atributos son rápidos de construir puesto que únicamente se prueba en cada nodo una pequeña fracción de ellos. Únicamente dependen de uno o dos parámetros de ajuste pero suelen ofrecer buenos resultados con los parámetros predeterminados. Permiten la detección de valores atípicos, así como la imputación o sustitución de valores perdidos [22], [23].

2.3 Conclusiones del capítulo

En este capítulo se ha realizado una introducción a la inteligencia artificial para posteriormente centrarse en el *Machine Learning* y las técnicas de aprendizaje supervisado de regresión lineal, modelo lineal generalizado, regresión de los k vecinos más cercanos, árboles de decisión y bosques aleatorios. Estos conocimientos se emplearán para hacer la predicción de las características de suelo mediante la implementación de las técnicas anteriormente citadas.

Capítulo 3: Materiales y métodos

En este capítulo se pone en contexto la base de datos utilizada para llevar a cabo la predicción de características de suelo. Además, se describe el proceso seguido para analizar estos datos, explicando tanto los pasos que se han dado como las medidas empleadas para determinar el rendimiento de los métodos usados para la predicción.

3.1 Datos de trabajo

La base de datos que se ha empleado en este trabajo se ha obtenido del Centro Común de Investigación de la Comisión Europea, más concretamente del Centro Europeo de Datos de Suelos (European Soil Data Centre, ESDAC). La ESDAC es el centro de datos relacionados con el suelo en Europa. Su principal objetivo es ser el punto de referencia para albergar todos los datos e información sobre el suelo a nivel europeo. Para ello, contiene una serie de recursos organizados en diferentes secciones: conjuntos de datos, servicios/aplicaciones, mapas, documentos, eventos, proyectos y enlaces externos [24].

En este proyecto la base de datos utilizada ha sido la base de datos analítica del perfil del suelo de Europa (Soil Profile Analytical Database of Europa, SPADBE) en su versión 2.1.0.0, la cual está incluida en la base de datos de suelo europea v2.0 (European Soil Database v2.0). La base de datos analítica del perfil del suelo de Europa consta de los siguientes parámetros cuantitativos para los diferentes horizontes del suelo [25]:

- Textura (y grados de tamaño de partícula).
- Contenido de materia orgánica (carbono, nitrógeno).
- Estructura.
- Contenido de nitrógeno total.
- pH.
- Porcentaje de sodio intercambiable (*Exchangeable Sodium Percentage*, ESP) o relación de absorción del sodio (*Sodium Adsorption Ratio*, SAR).
- Contenido de carbonato de calcio.
- Contenido de sulfato de calcio.
- Conductividad eléctrica.
- Capacidad de intercambio catiónico (*Cation Exchange Capacity*, CEC) y bases intercambiables
- Retención de agua del suelo.
- Densidad a granel.

- Profundidad de la raíz.
- Nivel del agua subterránea.
- Material padre/principal.

Los países analizados en esta base de datos han sido: Albania (AL), Austria (AT), Bélgica (BE), Bulgaria (BG), Suiza (CH), República Checa (CZ), Alemania (DE), Dinamarca (DK), Estonia (EE), España (ES), Finlandia (FI), Francia (FR), Gran Bretaña (GB), Grecia (GR), Hungría (HU), Irlanda (IR), Italia (IT), Letonia (LT), Luxemburgo (LU), Lituania (LV), Países Bajos (NL), Noruega (NO), Polonia (PO), Portugal (PT), Rumanía (RO), Suecia (SE), Eslovenia (SI) y Eslovaquia (SK). Cada uno de estos países tiene dos hojas de cálculo asociadas que son XX-EST_HOR.xls y XX-EST_PROF.xls, siendo XX las letras que identifican a cada país estudiado. Estas hojas de cálculo contienen los atributos o características de perfil medidos en cada país. Concretamente XX-EST_HOR.xls contiene una tabla con los datos referentes a los horizontes de los perfiles medidos y XX-EST_PROF.xls muestra en una tabla los datos referentes a las características generales de los perfiles estimados [25].

En el Anexo I se incluye una tabla con las siglas que aparecen en la base de datos y su significado.

3.2 Métodos empleados

En este TFM el procesado y análisis de los datos se ha realizado mediante el lenguaje de programación Python en su versión 3.8.8. Para ello se ha usado PyCharm, que es un entorno de desarrollo integrado desarrollado por JetBrains. Las bibliotecas de Python utilizadas han sido, por un lado, pandas y NumPy que son de propósito general y, por otro lado, scikit-learn y Sweetviz para realizar las predicciones y visualizar los datos respectivamente. Puesto que estas dos últimas bibliotecas son las más relevantes en la realización de este trabajo son las que se van a describir más detalladamente a continuación.

3.2.1 Scikit-learn

Scikit-learn es una biblioteca de aprendizaje automático de código abierto que permite trabajar tanto con modelos de aprendizaje supervisado como no supervisado. También dispone de múltiples herramientas para realizar el ajuste, la elección y la evaluación de los modelos, así como para el preprocesamiento de datos y otras muchas utilidades [20].

Scikit-learn cuenta con varios algoritmos y modelos de aprendizaje automático integrados denominados estimadores. Cada estimador se puede ajustar (*fit*) a ciertos datos empleando para ello su método de ajuste. Generalmente, el método de ajuste tiene dos entradas [20]:

- Las muestras de matriz o matriz de diseño X. Son las variables de entrada, es decir, las variables independientes. En la matriz X las muestras se suelen representar como filas y las características como columnas [20].
- Los valores objetivo y son las variables de salida. En el caso de tareas de regresión los valores son continuos mientras que para clasificación son discretos. La variable y suele ser un vector de una dimensión en la que la entrada i-ésima se corresponde con la i-ésima muestra de la matriz de diseño, es decir, se corresponde con la i-ésima fila de la matriz X [20].

En general se espera que tanto X como y sean vectores NumPy (numéricos) o vectores de tipos de datos equivalentes. Una vez que el estimador se ha ajustado, éste se puede emplear para predecir (*predict*) los valores objetivo de nuevos datos sin necesidad de tener que volver a entrenar al estimador [20].

Seguidamente, se muestra un ejemplo de ajuste y predicción utilizando un estimador lineal:

```
from sklearn import linear_model
estimator = linear_model.LinearRegression()
X = [[5, 1], [3, 8]] # Matriz de diseño
Y = [9, 4] # Valores objetivo
estimator.fit(X, y) # Ajuste
estimator.predict([[2, 0], [6, 7]]) # Predicción con nuevos datos de entrada
```

Los estimadores de la biblioteca scikit-learn que se han utilizado en este TFM son los siguientes:

- **LinearRegression()**

Este tipo de regresión se ajusta a un modelo lineal para minimizar la suma residual de cuadrados entre los objetivos observados del conjunto de entrenamiento y los objetivos que se han predicho mediante la aproximación lineal. Es decir, si los coeficientes del modelo lineal son $w = (w_1, \dots, w_p)$, matemáticamente este estimador resuelve un problema de la forma $\min_{\mathrm{w}} ||Xw - y||_2^2$ [20]. Para todos los parámetros se han utilizado los valores por defecto.

El código necesario para realizar el ajuste y predicción de los datos con la regresión lineal se muestra a continuación:

```
from sklearn import linear_model
reg_linear = linear_model.LinearRegression().fit(X_train, y_train)
y_estimated_LR = reg_linear.predict(X_test)
```

- **TweedieRegressor()**

Este estimador implementa un modelo lineal generalizado para la distribución Tweedie. Según el valor de los parámetros `power` y `link` se puede modelar una distribución normal, de Poisson, gamma o Gaussiana inversa [20]. Como en este TFM únicamente se han utilizado la

distribución Normal y la de Poisson los valores adecuados para estos parámetros son `power=0` y `link='identity'` en el caso de la distribución normal y para la distribución de Poisson `power=1` y `link='log'`. En el resto de los parámetros se han empleado los valores predeterminados, siendo el número máximo de iteraciones 100.

El código necesario para realizar el ajuste y predicción de los datos con el estimador `TweedieRegressor()` es el siguiente:

```
from sklearn import linear_model
reg_normal = linear_model.TweedieRegressor(power=0,
        link='identity').fit(X_train, y_train)
y_estimated_N = reg_normal.predict(X_test)

reg_poisson = linear_model.TweedieRegressor(power=1,
        link='log').fit(X_train, y_train)
y_estimated_P = reg_poisson.predict(X_test)
```

- **KNeighborsRegressor()**

En este tipo de regresión la etiqueta asociada al nuevo dato de consulta procedente del conjunto de prueba se calcula en función de las etiquetas de sus k vecinos más cercanos [20]. El parámetro que establece el número de vecinos que se utilizarán para determinar la etiqueta del nuevo dato, es decir el valor de k, es `n_neighbors`. En este caso, el valor escogido ha sido de 10, esto quiere decir que la etiqueta de cada dato de prueba se calculará a partir de la distancia entre esta y los 10 vecinos más cercanos procedentes del conjunto de entrenamiento. La razón por la que se eligió este valor fue porque al realizar varias pruebas para distintos valores de k, la que obtuvo los mejores resultados fue aquella en la que se había fijado el parámetro `n_neighbors` a 10. En el resto de los parámetros se han utilizado los valores por defecto, por lo que la métrica utilizada para establecer los vecinos más cercanos será la Euclidiana y el algoritmo empleado será el más apropiado de entre la fuerza bruta, el *K_D Tree* o *Ball Tree* en función de los valores pasados al método de ajuste.

A continuación, se puede ver el de código necesario para realizar el ajuste y predicción de los datos con la técnica de los k vecinos más cercanos:

```
from sklearn import neighbors
reg_knn = neighbors.KNeighborsRegressor(n_neighbors=10).fit(X_train, y_train)
y_estimated_KNN = reg_knn.predict(X_test)
```

- **DecisionTreeRegressor()**

Este estimador crea un modelo que es capaz de predecir una variable objetivo a partir del aprendizaje de reglas de decisión simples [20]. Se ha asignado un valor fijo al parámetro `random_state` para controlar la aleatoriedad, de esta manera se obtienen siempre los mismos resultados. En el resto de los parámetros los valores empleados han sido los predeterminados, siendo el error cuadrático medio el criterio para medir la calidad de una división y la estrategia

utilizada para dividir los nodos aquella que escoge la mejor división de entre todas las que haya.

Seguidamente, se muestra el código necesario para realizar el ajuste y predicción de los datos con árboles de decisión:

```
from sklearn import tree
reg_tree = tree.DecisionTreeRegressor(random_state=0).fit(X_train, y_train)
y_estimated_DT = reg_tree.predict(X_test)
```

- **RandomForestRegressor()**

Los bosques aleatorios están formados por un conjunto de árboles de decisión que hacen predicciones a partir de la combinación de las realizadas por cada árbol de manera individual [23]. De esta manera se logra mejorar la precisión de la predicción y controlar el sobreajuste [20]. Para controlar la aleatoriedad se ha asignado un valor fijo al parámetro `random_state`, con ello lo que se consigue es obtener siempre los mismos resultados. En el resto de los parámetros se han utilizado los valores por defecto, por lo que la cantidad de árboles en el bosque será de 100 y la función para medir la calidad de una división será el error cuadrático medio.

El código necesario para realizar el ajuste y predicción de los datos con bosques aleatorios se muestra a continuación:

```
from sklearn import ensemble
reg_forest = ensemble.RandomForestRegressor(random_state=0).fit(X_train,
       y_train)
y_estimated_RF = reg_forest.predict(X_test)
```

3.2.2 Sweetviz

Sweetviz es una biblioteca de código abierto que sirve para generar visualizaciones tras realizar un análisis exploratorio de datos (*Exploratory Data Analysis*, EDA). Para ello, Sweetviz permite crear un informe HTML de los datos analizados empleando únicamente dos líneas de código [26].

El objetivo del sistema a través de las visualizaciones es ayudar a realizar un análisis rápido de las características de los valores objetivo frente al resto de variables, así como a realizar una comparación entre conjuntos de datos. Esta comparativa puede hacerse entre distintos conjuntos de datos como pueden ser los datos de entrenamiento y los de prueba o entre características dentro de un mismo conjunto [26].

Otra característica de esta biblioteca es que también es capaz de detectar de manera automática el tipo de dato, es decir, si son características numéricas, categóricas o de texto, así como los valores únicos, perdidos o más frecuentes y si hay filas duplicadas. Además, realiza un análisis numérico en el que, por cada conjunto de datos, muestra tanto el valor

mínimo y máximo como el rango, la media, la mediana y la moda, la desviación estándar, la suma, el parámetro de kurtosis y la asimetría y distintos cuartiles [26].

Para crear y visualizar un informe de Sweetviz primero hay que emplear la función `analyze()` para realizar el análisis de los datos y crear el objeto de informe. Posteriormente, para mostrar el informe hay que utilizar la función `show_html()`, la cual creará y guardará un informe HTML en la ruta proporcionada con el nombre del fichero deseado. Una vez que el fichero se ha generado, se abrirá automáticamente a través del navegador predeterminado [26].

El código necesario para realizar estos pasos es el siguiente:

```
import sweetviz as sv
my_report = sv.analyze(data_analyze) # Creación del informe
my_report.show_html('filepath/filename.html') # Visualización del informe
```

3.3 Proceso del análisis de datos

En esta sección se va a describir el proceso que se ha seguido para realizar el análisis de los datos en este Trabajo Fin de Máster. En primer lugar, se van a explicar cada uno de los pasos dados para, a continuación, detallar cómo se ha realizado la evaluación de los métodos empleados.

En el Anexo II se puede ver el código desarrollado para mostrar los datos de EST-PROF y de EST-HOR, junto con el código para procesar y analizar los datos presentes en EST-HOR.

3.3.1 Pasos realizados

En primer lugar, se han unificado las hojas de cálculo individuales de cada país en una sola con el objetivo de facilitar el posterior análisis y estudio de las características de los suelos. La hoja de cálculo que agrupa los datos de los horizontes de los perfiles de todos los países se ha llamado EST-HOR.xlsx, mientras que EST-PROF.xlsx contiene la información de las características generales de los perfiles estimados de todos los países.

Seguidamente, se puede ver el código necesario para realizar la unificación de las hojas de cálculo:

```
import pandas as pd
import glob
import os
juntar = pd.concat(map(pd.read_excel, glob.glob(os.path.join('',
       "./../data/SPADE14/*-EST_HOR.xls"))))
juntar.to_excel('./../data/EST-HOR.xlsx')

juntar = pd.concat(map(pd.read_excel, glob.glob(os.path.join('',
       "./../data/SPADE14/*-EST_PROF.xls"))))
juntar.to_excel('./../data/EST-PROF.xlsx')
```

Debido a que los sensores que se emplean en el campo miden valores numéricos, la hoja de cálculo con la que se ha seguido trabajando es EST-HOR.xlsx ya que cuenta con más características de tipo numérico.

En segundo lugar, se ha llevado a cabo una etapa de preprocesado con el objetivo de identificar y eliminar los datos no válidos. Esto se debe a que la documentación adjunta con la base de datos indicaba que aquellas características con un valor de -999 hacían referencia a un valor desaparecido (*missing value*) y un valor de -998 significaba que no era aplicable ya que se refería a rocas u horizontes orgánicos. Por este motivo, se han localizado estos valores en la hoja de cálculo conjunta de todos los países y se han establecido como NaN (*Not a Number*) para que en posteriores operaciones fuera más fácil su identificación. Posteriormente, en las características que se empleen para realizar el análisis predictivo se eliminarán de los datos todas aquellas muestras catalogadas como NaN para evitar errores.

A continuación, se han seleccionado únicamente las características de tipo numérico ya que son las que contienen los datos que se quieren estudiar y analizar. De esta forma, de las 33 características de las que constaba en un principio EST-HOR, únicamente se analizarán 25 puesto que el resto son categóricas, de texto o irrelevantes.

La obtención de algunas características presentes en la hoja de cálculo requiere un análisis en el laboratorio que es tedioso, costoso o necesita una gran cantidad de instrumentación para realizar la medición. Por este motivo, el siguiente paso ha sido el de realizar un análisis predictivo de una de estas características a partir de otras cuya medición es más directa y con las que están relacionadas. El objetivo de este análisis predictivo, por lo tanto, es proponer un método alternativo para el cálculo de estos parámetros de interés de la base de datos, con el menor error asociado posible y que evite los factores negativos mencionados anteriormente. Por ello, las características que se van a intentar predecir son las siguientes:

- C_N a partir de PH, OM y TEXT
- CACO3_TOT a partir de PH, TEXT, EXCH_K y EXCH_CA
- WC a partir de OM, TEXT y EXCH_CA

Tal y como se ha descrito en la subsección 3.2.1 las técnicas de estimación o estimadores empleados para realizar las predicciones han sido regresión normal, regresión de Poisson, regresión lineal, regresión KNN, bosques aleatorios (R*andom Forest*) y árboles de decisión (*Decision Trees*). Puesto que algunas de estos estimadores se ajustan y funcionan mejor con los datos normalizados se ha realizado la predicción tanto con los datos normalizados como sin normalizar. De esta forma, al comparar los resultados obtenidos se va a comprobar si este paso es necesario o no.

Para ajustar/entrenar estos modelos se ha utilizado el 65% de los datos existentes y el resto de los datos, el 35%, se ha empleado para predecir/testear. Con el objetivo de evitar

problemas o tendencias en los resultados por utilizar datos para entrenar procedentes de unos países y para predecir los restantes, se distribuyeron aleatoriamente todos los datos disponibles en la hoja de cálculo. De esta manera, al mezclar los datos de los países entre sí aumentó la diversidad de estos.

A continuación, se muestra el código necesario para distribuir de manera aleatoria los datos de EST-HOR:

```
import pandas as pd
data_HOR = pd.read_excel('./../data/EST-HOR.xlsx')
data_HOR_random = data_HOR.sample(frac=1)
data_HOR_random.to_excel('./../data/EST-HOR_random.xlsx')
```

Por último, se ha analizado el rendimiento de todos los estimadores implementados utilizando las métricas que se van a presentar y explicar en la siguiente subsección.

3.3.2 Rendimiento de los estimadores

Tal y como se ha comentado en la subsección 3.3.1 para comprobar el rendimiento de los estimadores se han empleado dos tipos de medidas, la raíz del error cuadrático medio y el coeficiente de correlación. En ambas se compara la diferencia entre los parámetros estimados y los reales para determinar qué técnica es la más eficiente de todas.

- **Raíz del error cuadrático medio**

Tanto el error cuadrático medio (*Mean Squared Error*, MSE) como la raíz del error cuadrático medio (*Root Mean Squared Error*, RMSE) son dos de los métodos más utilizados en los modelos de regresión para medir la diferencia entre los parámetros estimados y los parámetros reales [10], [27].

Si $\hat{x}$ es un estimador del parámetro real x, el error cuadrático al estimar x por $\hat{x}$ es: $(x - \hat{x})^2$ [10], [26].

Por lo tanto, el error cuadrático medio se define como la esperanza matemática de este valor, es decir, el valor esperado de la diferencia al cuadrado entre una estimación del parámetro real y el propio parámetro real [10], [27]:

$$MSE(\hat{x}) = E[(x - \hat{x})^2]$$

Si se normaliza según el número de muestras $n_{samples}$ el error cuadrático medio quedaría de la siguiente forma [20]:

$$MSE(\hat{x}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}-1} (x_i - \hat{x}_i)^2$$

También, cabe destacar que la eficiencia de un estimador es inversamente proporcional a su error cuadrático medio [10].

El RMSE no es más que la raíz cuadrada del error cuadrático medio, pero cuenta con la ventaja de que tiene las mismas unidades que el parámetro que evalúa, lo que hace que el RMSE sea más fácilmente interpretable que el MSE en este sentido. A continuación, se muestra la ecuación del RMSE:

$$RMSE(\hat{x}) = \sqrt{\frac{\sum_{i=1}^{n_{samples}-1}(x_i - \hat{x_i})^2}{n_{samples}}}$$

En este TFM la función empleada para calcular la raíz del error cuadrático medio es `mean_squared_error()` que se encuentra en el módulo `metrics` de la biblioteca scikit-learn. Esta función por defecto calcula el error cuadrático medio, pero fijando el parámetro `squared` a `False` devuelve la raíz del error cuadrático medio.

```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_estimated_LR, y_true, squared=False)
```

- **Coeficiente de correlación**

El coeficiente de correlación mide la fuerza de la relación lineal entre dos variables aleatorias. Puede tomar valores en el intervalo [-1, 1], siendo los valores extremos 1 y -1 el indicativo de una relación perfectamente lineal entre las variables. Una relación positiva (+) significa que las dos variables varían en la misma dirección. Por el contrario, una relación negativa (-) significa que las variables varían en direcciones distintas. El valor nulo implica la ausencia de una relación lineal, es decir que las variables que se están analizando son independientes [10].

En este TFM se ha utilizado el coeficiente de correlación de Pearson para evaluar el rendimiento de los estimadores. Si X e Y son variables aleatorias que siguen una distribución conjunta desconocida, entonces el coeficiente de correlación es [10]:

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

Donde σ_X y σ_Y son las desviaciones estándar de X e Y respectivamente y $Cov(X,Y)$ es la covarianza medida entre X e Y [10].

Si se trabaja con matrices, R sería la matriz de coeficientes de correlación y C la matriz de covarianzas, por lo que la expresión anterior se puede reescribir como [28]:

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} \cdot C_{jj}}}$$

En este TFM la función empleada para calcular el coeficiente de correlación es `corrcoef()` disponible en la biblioteca NumPy.

```
import numpy as np
coef_corr = np.corrcoef(y_estimated_LR, y_true)[0,1]
```

3.4 Conclusiones del capítulo

En este capítulo se han presentado los materiales y los métodos que se han utilizado en este estudio, así como el proceso del análisis de datos que se ha seguido. Los resultados derivados del empleo de esta metodología se expondrán en el próximo capítulo.

Capítulo 4: Resultados y discusión

Para ver el tipo de variables (numéricas, categóricas o de texto) y la relación entre ellas se realizó un análisis de los datos incluidos tanto en EST-PROF como en EST-HOR. A continuación, se estimaron los parámetros de difícil adquisición C_N, CACO3_TOT y WC_FC mediante los estimadores de regresión lineal, regresión normal, regresión KNN, regresión de Poisson, *Decision Tree* y *Random Forest*. Por último, se calculó el RMSE y el coeficiente de correlación para determinar el rendimiento de los estimadores empleados.

4.1 Análisis de los datos incluidos en EST-PROF

En primer lugar, tras abrir el informe HTML generado con Sweetviz de los datos incluidos en EST-PROF se puede observar un resumen de estos. En él se indica el número de filas de las que consta el fichero, así como el número de columnas o características que contiene y distingue cuántas de ellas son categóricas, numéricas o de texto. En este caso tal y como se puede ver en la Figura 3, EST-PROF cuenta con 1078 filas y 59 características de las cuales 41 son categóricas, 14 son numéricas y 4 son de texto.

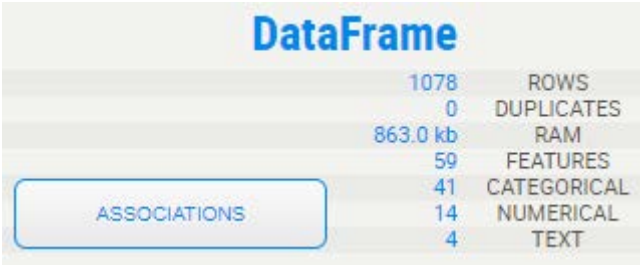


Figura 3: Resumen de las características incluidas en EST-PROF.

Además, si se pulsa el botón "Associations" aparecen en un único gráfico las correlaciones existentes entre las distintas variables. Los círculos representan la correlación numérica, es decir, la correlación entre dos variables de tipo numérico, mientras que los cuadrados representan o bien el coeficiente de incertidumbre o bien la razón de correlación. El coeficiente de incertidumbre se calcula a partir de dos variables categóricas, en cambio, la razón de correlación mide la relación existente entre una variable categórica y otra numérica. En el Anexo I se incluye una tabla con las siglas de la base de datos y su significado.

En la Figura 4 se puede observar que en el gráfico de las correlaciones de EST-PROF hay muy pocos círculos, lo que indica que la mayor parte de las asociaciones se dan entre variables categóricas o categóricas y numéricas.

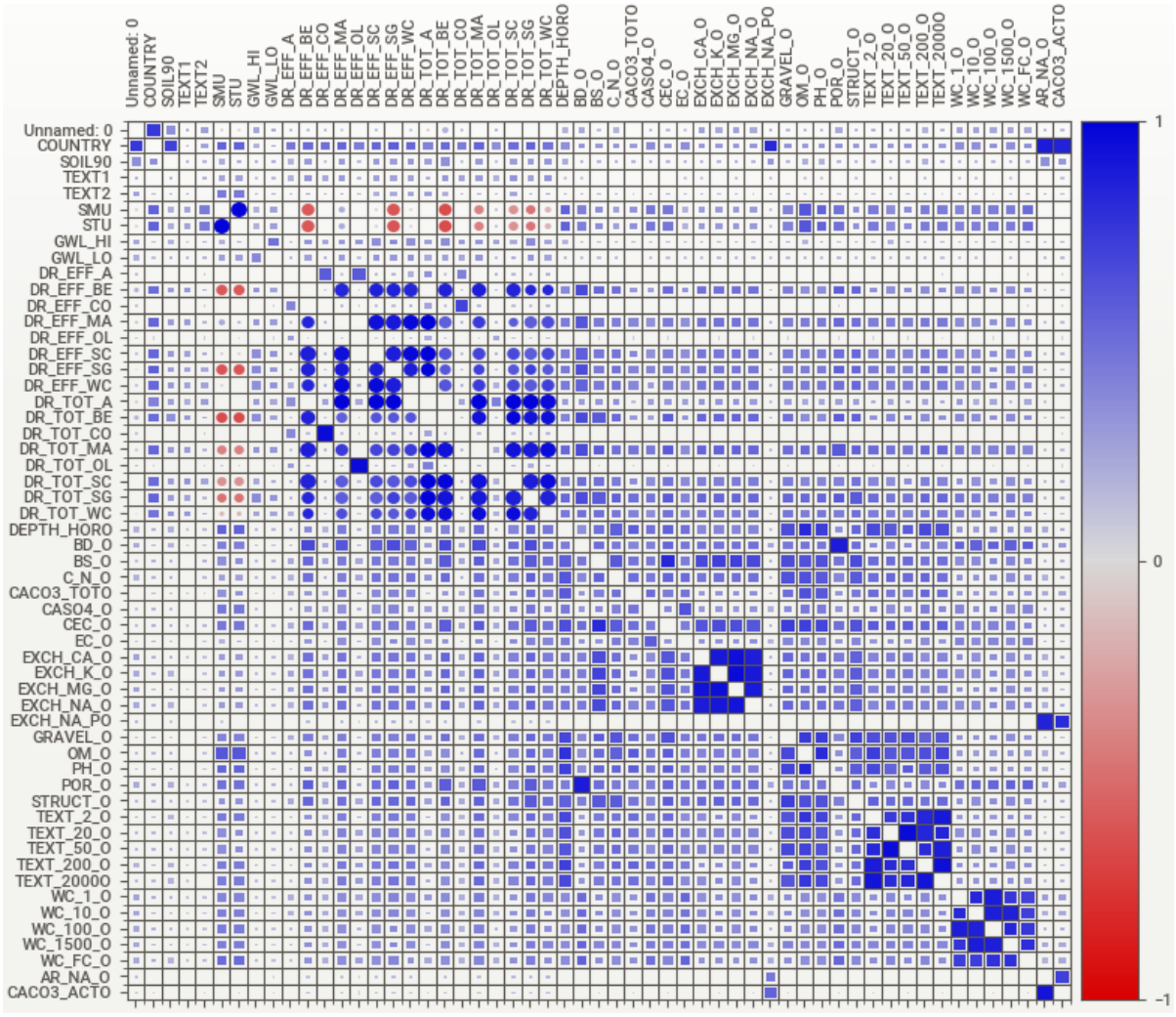


Figura 4: Gráfico de las asociaciones de las características incluidas en EST-PROF. Los cuadrados representan variables relacionadas con características categóricas, es decir, muestran el coeficiente de incertidumbre en el caso de dos variables categóricas y la razón de correlación cuando una variable es categórica y la otra numérica. Los círculos representan la correlación numérica, que es el coeficiente de correlación entre dos variables numéricas. El color azul denota una correlación positiva mientras que el color rojo indica que la correlación es negativa. Además, cuanto más intensos son los colores mayor es la correlación, en valor absoluto, entre las variables.

4.2 Análisis de los datos incluidos en EST-HOR

A continuación, se analizó el informe HTML generado con Sweetviz de EST-HOR. En el resumen se puede observar que hay una mayor cantidad de datos puesto que, tal y como se puede ver en la Figura 5, el número de filas es de 4485. En cambio, ahora el número de características se ve reducido a 33 de las cuales 4 son categóricas, 27 son numéricas y 2 son de texto. Las distintas asociaciones entre las variables de EST-HOR se observan en la Figura 6.

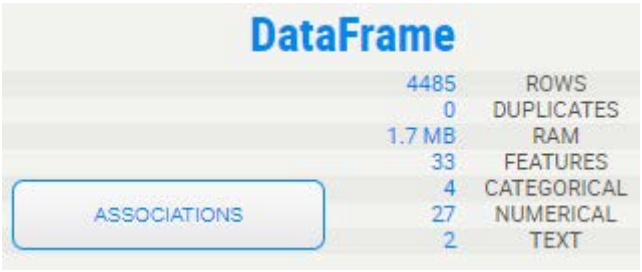


Figura 5: Resumen de las características incluidas en EST-HOR.

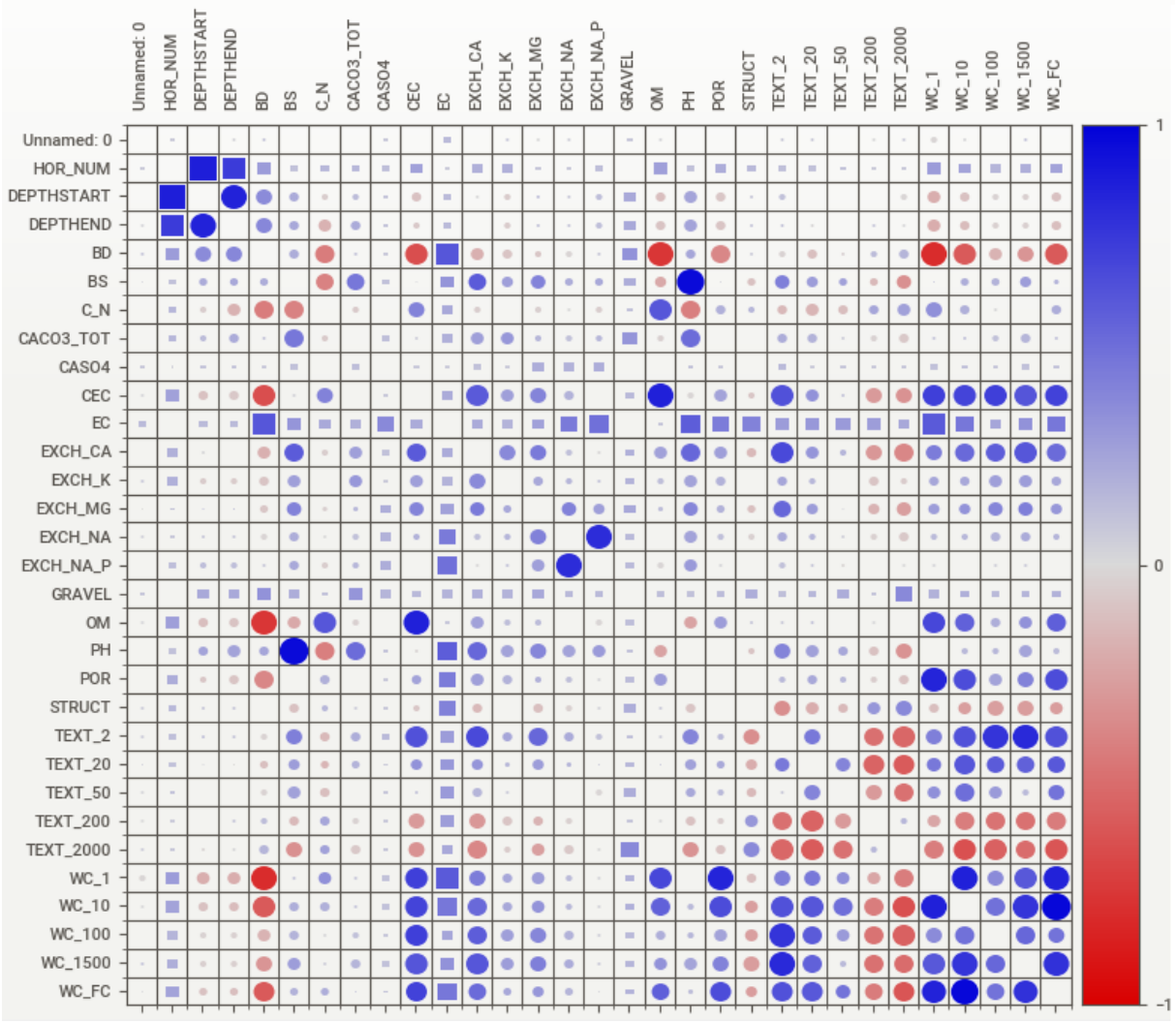


Figura 6: Gráfico de las asociaciones de las características incluidas en EST-HOR. Los cuadrados representan variables relacionadas con características categóricas, es decir, muestran el coeficiente de incertidumbre en el caso de dos variables categóricas y la razón de correlación cuando una variable es categórica y la otra numérica. Los círculos representan la correlación numérica, que es el coeficiente de correlación entre dos variables numéricas. El color azul denota una correlación positiva mientras que el color rojo indica que la correlación es negativa. Además, cuanto más intensos son los colores mayor es la correlación, en valor absoluto, entre las variables.

Como los sensores que se emplean en el campo miden valores numéricos, se ha decidido seguir trabajando con esta hoja de cálculo ya que cuenta con más características de este tipo. Además, trabajar tanto con características categóricas como numéricas implicaría un análisis y trabajo mucho más complejo.

El siguiente paso fue eliminar las características categóricas y de texto y también aquellas características numéricas que no aportaban información relevante. Tras realizar estos cambios, el gráfico obtenido es el que se puede observar en la Figura 7.

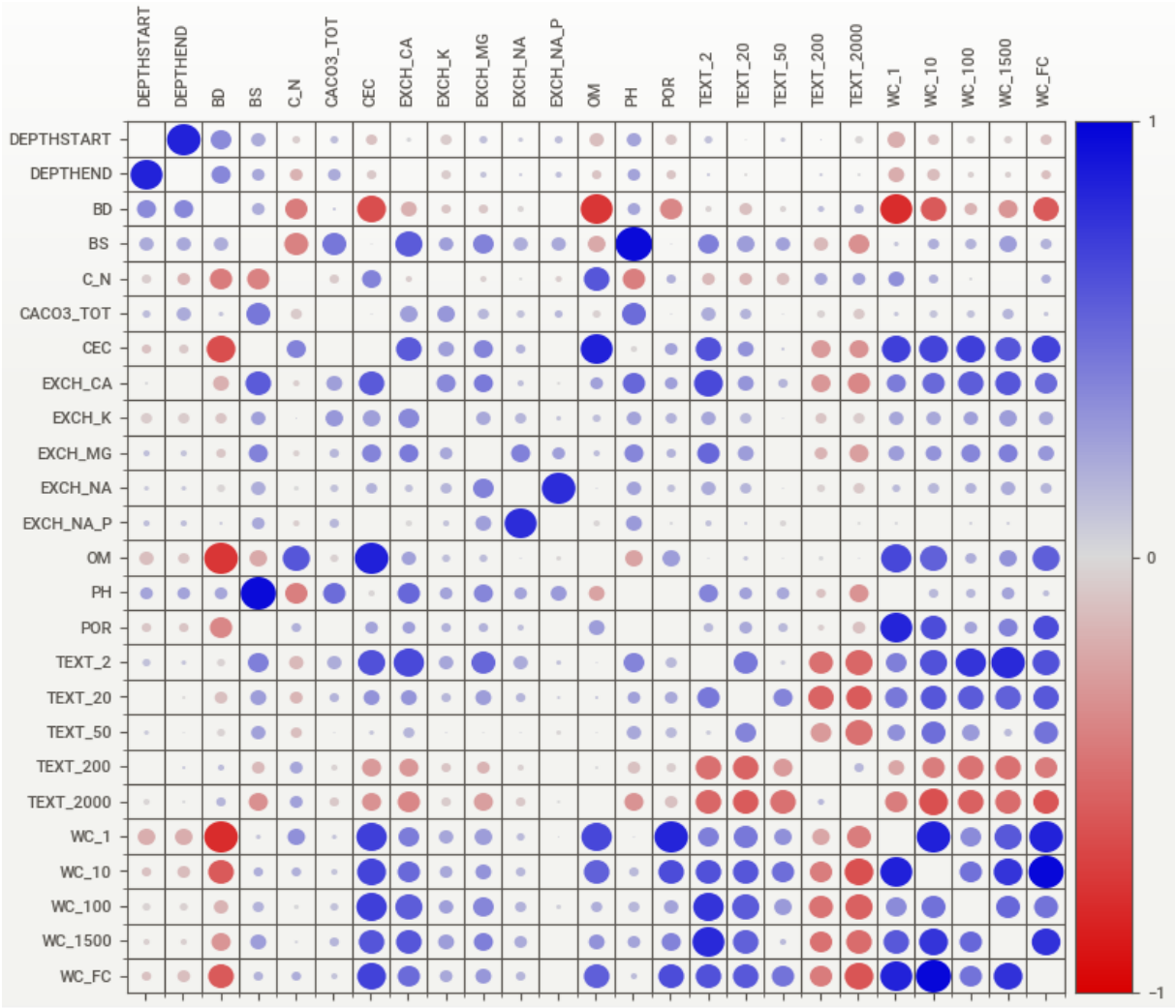
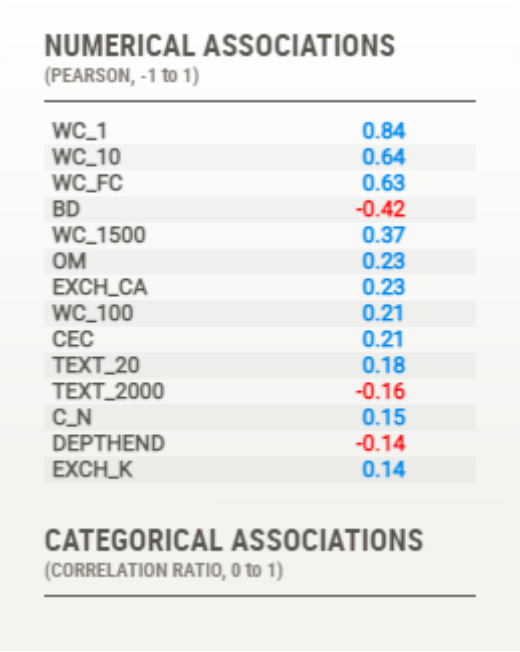


Figura 7: Gráfico de las asociaciones de las características incluidas en EST-HOR tras haber eliminado las características categóricas, de texto y las numéricas no relevantes.

Sweetviz también ofrece la posibilidad de obtener el resultado numérico de las relaciones entre variables (correlación numérica, coeficiente de incertidumbre o razón de correlación). Para cada variable categórica o numérica muestra las 14 asociaciones de mayor valor con el resto. En este caso, como se han eliminado las variables categóricas y de texto, todas las asociaciones que se muestran son coeficientes de correlación entre variables numéricas. En la Figura 8 se pueden observar las asociaciones numéricas de la característica de la porosidad total (POR) con el resto de las variables. El valor de estas asociaciones se calcula a partir del coeficiente de correlación de Pearson.



NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

WC_1	0.84
WC_10	0.64
WC_FC	0.63
BD	-0.42
WC_1500	0.37
OM	0.23
EXCH_CA	0.23
WC_100	0.21
CEC	0.21
TEXT_20	0.18
TEXT_2000	-0.16
C_N	0.15
DEPTHEND	-0.14
EXCH_K	0.14

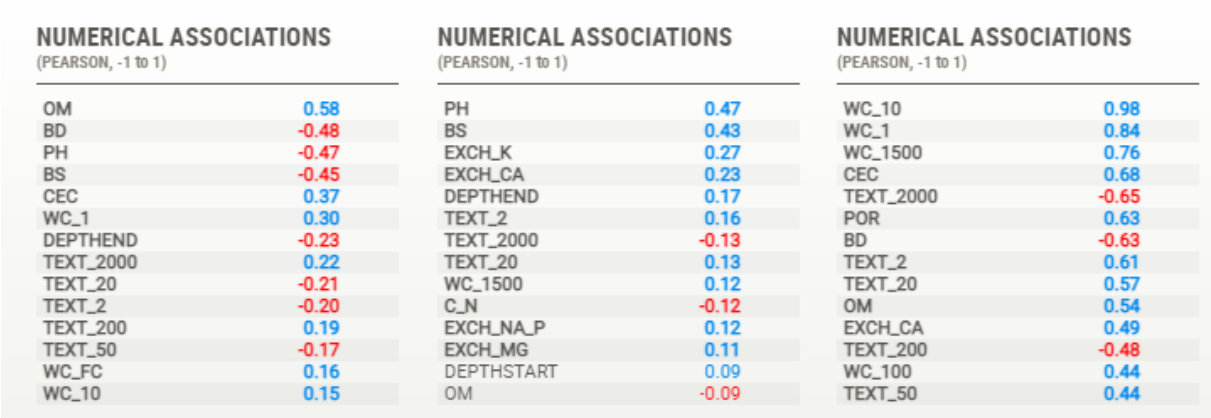
CATEGORICAL ASSOCIATIONS
(CORRELATION RATIO, 0 to 1)

Figura 8: Resultado de las asociaciones categóricas y numéricas de la característica POR con el resto de las variables. En la parte inferior de la figura se puede ver que no aparece ningún valor referente a la razón de correlación, puesto que se han eliminado las variables categóricas del análisis.

En base a estos datos se escogieron 3 características cuya obtención requiere de un análisis difícil y caro para predecirlas a partir de otras cuyo análisis es sistemático y sencillo. Además, estas características guardan una cierta correlación con las primeras, pero no lo suficientemente fuerte como para poder hacer una estimación directa de una en función de la otra. Por ello, las características elegidas como variables a estimar fueron C_N (relación carbono-nitrógeno), CACO3_TOT (peso equivalente de carbonato de calcio) y WC_FC (retención de agua del suelo a capacidad de campo), cuya correlación numérica con el resto de las variables se muestra en la Figura 9. Paralelamente, las características seleccionadas para predecir estas variables fueron el pH (PH), el contenido de materia orgánica (OM), la proporción de partículas de tamaño 200-2000 µm (TEXT_2000), 2-20 µm (TEXT_20) e inferior a 2 µm (TEXT_2) y las bases intercambiables de calcio (EXCH_CA) y de potasio (EXCH_K). Concretamente se pretende estimar:

- C_N a partir de PH, OM y TEXT
- CACO3_TOT a partir de PH, TEXT, EXCH_K y EXCH_CA
- WC a partir de OM, TEXT y EXCH_CA

En las siguientes subsecciones se analizarán los resultados conseguidos con la regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest* para cada una de las variables estimadas. Por último, en la sección 4.3 se realizará una discusión de los resultados obtenidos en la que se explicará, entre otras cosas, el motivo por el que no se muestran los resultados para la regresión de Poisson con datos normalizados.



NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

OM	0.58
BD	-0.48
PH	-0.47
BS	-0.45
CEC	0.37
WC_1	0.30
DEPTHEND	-0.23
TEXT_2000	0.22
TEXT_20	-0.21
TEXT_2	-0.20
TEXT_200	0.19
TEXT_50	-0.17
WC_FC	0.16
WC_10	0.15

NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

PH	0.47
BS	0.43
EXCH_K	0.27
EXCH_CA	0.23
DEPTHEND	0.17
TEXT_2	0.16
TEXT_2000	-0.13
TEXT_20	0.13
WC_1500	0.12
C_N	-0.12
EXCH_NA_P	0.12
EXCH_MG	0.11
DEPTHSTART	0.09
OM	-0.09

NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

WC_10	0.98
WC_1	0.84
WC_1500	0.76
CEC	0.68
TEXT_2000	-0.65
POR	0.63
BD	-0.63
TEXT_2	0.61
TEXT_20	0.57
OM	0.54
EXCH_CA	0.49
TEXT_200	-0.48
WC_100	0.44
TEXT_50	0.44

Figura 9: Resultado de las asociaciones numéricas de las características C_N (izquierda), CACO3_TOT (centro) y WC_FC (derecha) con el resto de las variables.

4.2.1 Estimación de la característica C_N

Para estimar la relación carbono-nitrógeno (C_N) se han realizado las siguientes pruebas:

- Estimación de C_N a partir de PH, OM y TEXT_2000

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	4.53	4.53	0.59	0.59
Regresión normal	4.50	4.69	0.60	0.59
Regresión Poisson	47	-	0.56	-
Regresión KNN	4.18	4.37	0.66	0.64
Decision Tree	5.26	5.06	0.63	0.64
Random Forest	**3.97**	**3.98**	**0.73**	**0.73**

Tabla 1: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de C_N a partir de PH, OM y TEXT_2000 empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de C_N a partir de TEXT_2000, TEXT_20 y TEXT_2

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	5.41	5.41	0.26	0.26
Regresión normal	5.41	5.41	0.26	0.27
Regresión Poisson	5.41	-	0.26	-
Regresión KNN	5.45	5.41	0.35	0.35
Decision Tree	5.73	5.71	0.52	0.53
Random Forest	**4.72**	**4.72**	**0.58**	**0.58**

Tabla 2: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de C_N a partir de TEXT_2000, TEXT_20 y TEXT_2 empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de C_N a partir de PH y OM

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	5.10	5.10	0.70	0.70
Regresión normal	5.10	5.55	0.70	0.69
Regresión Poisson	5.10	-	0.70	-
Regresión KNN	**4.89**	**4.77**	0.73	**0.74**
Decision Tree	5.80	5.73	0.67	0.68
Random Forest	4.99	4.98	**0.73**	0.73

Tabla 3: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de C_N a partir de PH y OM empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

En la Figura 9 se puede ver que la mayor correlación de las variables que se emplean para estimar C_N era de 0.58 con OM. Sin embargo, utilizando estimadores se consigue mejorar este dato ya que con la combinación de PH y OM y la regresión KNN se obtiene una correlación de 0.74 tal y como se muestra en la Tabla 3. Aquí también se puede observar que la correlación para los datos sin normalizar es mayor con *Random Forest* aunque, pese a esto, KNN es mejor puesto que presenta un error más bajo. En cambio, para los casos mostrados en la Tabla 1 y en la Tabla 2 el mejor estimador es *Random Forest* dado que es el que tiene un menor RMSE.

4.2.2 Estimación de la característica CACO3_TOT

Para estimar el peso equivalente de carbonato de calcio (CACO3_TOT) se han realizado las siguientes pruebas:

- Estimación de CACO3_TOT a partir de PH, EXCH_CA y EXCH_K

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	10.38	10.38	0.51	0.51
Regresión normal	10.59	10.74	0.49	0.49
Regresión Poisson	9.97	-	0.56	-
Regresión KNN	9.25	8.64	0.65	0.70
Decision Tree	8.73	8.69	0.74	0.74
Random Forest	**7.23**	**7.13**	**0.80**	**0.81**

Tabla 4: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de CACO3_TOT a partir de PH, EXCH_K y EXCH_CA empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de CACO3_TOT a partir de PH, EXCH_CA y TEXT_2

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	10.43	10.43	0.49	0.49
Regresión normal	10.60	10.79	0.48	0.47
Regresión Poisson	10.01	-	0.55	-
Regresión KNN	9.93	8.42	0.56	0.71
Decision Tree	8.44	7.94	0.76	0.77
Random Forest	**7.13**	**7.11**	**0.80**	**0.81**

Tabla 5: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de CACO3_TOT a partir de PH, EXCH_CA y TEXT_2 empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de CACO3_TOT a partir de PH, TEXT_2 y TEXT_20

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	10.49	10.49	0.49	0.49
Regresión normal	10.70	10.88	0.47	0.47
Regresión Poisson	10.41	-	0.51	-
Regresión KNN	10.28	8.80	0.53	0.69
Decision Tree	8.58	8.81	0.73	0.72
Random Forest	**7.62**	**7.66**	**0.78**	**0.77**

Tabla 6: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de CACO3_TOT a partir de PH, TEXT_2 y TEXT_20 empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

En la Figura 9 se puede ver que la mayor correlación de las variables que se emplean para estimar CACO3_TOT era de 0.47 con PH. Sin embargo, utilizando estimadores se consigue mejorar este dato ya que, tal y como se muestra en la Tabla 4, con la combinación de PH, EXCH_CA y EXCH_K y *Random Forest* se alcanza una correlación de 0.81. En la Tabla 4, Tabla 5 y Tabla 6 se puede observar que el mejor estimador en todos los casos es *Random Forest* dado que es el que consigue un RMSE más bajo.

4.2.3 Estimación de la característica WC_FC

Para estimar la retención de agua del suelo a capacidad de campo (WC_FC) se han realizado las siguientes pruebas:

- Estimación de WC_FC a partir de TEXT_2000, OM y EXCH_CA

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	6.70	6.70	0.73	0.73
Regresión normal	6.71	7.34	0.73	0.73
Regresión Poisson	7.01	-	0.70	-
Regresión KNN	6.43	6.37	0.76	0.77
Decision Tree	6.47	6.61	0.79	0.77
Random Forest	**5.41**	**5.40**	**0.84**	**0.84**

Tabla 7: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de WC_FC a partir de TEXT_2000, OM y EXCH_CA empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de WC_FC a partir de TEXT_2, TEXT_20 y TEXT_2000

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	6.90	6.90	0.71	0.71
Regresión normal	6.90	7.23	0.71	0.71
Regresión Poisson	7.07	-	0.70	-
Regresión KNN	6.63	6.54	0.74	0.75
Decision Tree	6.69	6.76	0.77	0.76
Random Forest	**5.61**	**5.61**	**0.82**	**0.82**

Tabla 8: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de WC_FC a partir de TEXT_2, TEXT_20 y TEXT_2000 empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

- Estimación de WC_FC a partir de TEXT_2, TEXT_2000 y OM

Estimador	RMSE datos sin normalizar	RMSE datos normalizados	Correlación datos sin normalizar	Correlación datos normalizados
Regresión lineal	6.40	6.40	0.76	0.76
Regresión normal	6.40	7.08	0.76	0.76
Regresión Poisson	6.76	-	0.73	-
Regresión KNN	5.86	5.80	0.80	0.81
Decision Tree	6.03	6.04	0.81	0.81
Random Forest	**4.89**	**4.91**	**0.87**	**0.87**

Tabla 9: RMSE y correlación de los datos sin normalizar y normalizados para la estimación de WC_FC a partir de TEXT_2, TEXT_2000 y OM empleando las técnicas de regresión lineal, regresión normal, regresión de Poisson, regresión KNN, *Decision Tree* y *Random Forest*.

En la Figura 9 se puede ver que la mayor correlación de las variables que se emplean para estimar WC_FC era de -0.65 con TEXT_2000. Sin embargo, utilizando estimadores se consigue mejorar este dato ya que, tal y como se muestra en la Tabla 9, con la combinación de TEXT_2, TEXT_2000 y OM y *Random Forest* se alcanza una correlación de 0.87. En la Tabla 7, Tabla 8 y Tabla 9 se puede observar que el mejor estimador en todos los casos es *Random Forest* dado que es el que consigue un RMSE más bajo.

4.3 Discusión de resultados

En esta sección se van a discutir y detallar aquellos aspectos más relevantes sobre los resultados obtenidos en la sección 4.2.

En primer lugar, hay que mencionar que la regresión de Poisson únicamente se ha utilizado con los datos sin normalizar ya que al normalizar los datos se obtienen valores negativos y el dominio del estimador basado en la regresión de Poisson es $[0, \infty)$.

Tras la estimación de las 3 características escogidas se puede comprobar que la normalización de los datos no es estrictamente necesaria en los casos analizados. Esto se debe a que no se aprecian diferencias significativas en los resultados obtenidos tras la normalización o sin normalizar. Sin embargo, de acuerdo con lo visto en la literatura es posible que en datos con otra distribución sí se aprecien diferencias y ventajas de la normalización. Además, tal y como se menciona en el párrafo anterior, hay que tener en cuenta la desventaja de la normalización en la regresión de Poisson debido al dominio de los valores con los que trabaja ese estimador.

En el caso de la estimación de C_N, la mayor correlación se obtuvo con la combinación de PH y OM. Sin embargo, si a estas características se añade TEXT_2000 se consigue disminuir el RMSE sin prácticamente afectar a la correlación. A priori, TEXT_2000 parecía que no iba a suponer una mejora puesto que la correlación con C_N era de apenas 0.22 como se puede ver en la Figura 9, pero combinándola con las características adecuadas logró una mejora significativa del estimador. Por lo tanto, se deduce que una característica que parece que no guarda mucha relación con la variable a estimar consigue un rendimiento del estimador bastante mayor, lo que sugiere que es positivo añadir esta característica al modelo.

En la estimación de CACO3_TOT la combinación que conseguía un menor RMSE era PH, EXCH_CA y TEXT_2. Este dato es significativo puesto que en el resto de las pruebas las variables empleadas para estimar eran el PH junto a las bases intercambiables de calcio y potasio por un lado, y con dos texturas por otro lado. Algo parecido ocurre con la predicción de WC_FC, que mediante la combinación de TEXT_2000, TEXT_2 y OM obtiene el RMSE más bajo en lugar de emplear tres texturas para ello. El motivo por el cual no se logran los mejores resultados con la combinación de características del mismo tipo se debe probablemente a que hay demasiada relación entre ellas. Esto quiere decir que, aunque la correlación de cada una de estas características con la que se va a estimar sea alta, la información que aportan es

redundante, mientras que si se combinan características distintas se complementa la información entre ellas.

Por último, se puede concluir que el mejor estimador es *Random Forest*. La regresión lineal es un método de predicción muy simple por lo que tiene sentido que no obtenga los mejores resultados. En cuanto a la regresión normal y de Poisson, ambas asumen que los datos tienen una distribución exponencial y, a la vista de los resultados, los datos que se han empleado para hacer las predicciones no cumplen esta condición. Las razones por las que la regresión KNN no es el mejor estimador pueden ser varias, ya que únicamente utiliza la distancia para clasificar e ignora totalmente la distribución espacial de los datos. Además, el valor de k para clasificar cada nuevo dato es fijo y único, cuando el número de vecinos necesarios debería de ser distinto en función de si el nuevo dato se encuentra en el centro de una clase de entrenamiento, o si se halla en la superposición de varias clases. El motivo por el que *Decision Tree* no consigue los mejores resultados es porque este estimador tiene una alta varianza. Este problema se soluciona con el método de predicción *Random Forest* ya que, al estar formado por un conjunto de árboles de decisión, las predicciones se calculan como el promedio de los árboles generados. Por lo tanto, debido al propio diseño de *Random Forest*, que combina varios árboles de decisión para aumentar la precisión y reducir la varianza, éste siempre conseguirá un resultado mejor que con un árbol de decisión individual.

Capítulo 5: Conclusiones y líneas futuras

Tras exponer los métodos y materiales utilizados y mostrar y discutir los resultados obtenidos, en este capítulo se presentan las conclusiones extraídas y las posibles líneas futuras por las que podría continuar este trabajo.

5.1 Conclusiones

Por último, tras analizar y comparar los resultados obtenidos se han obtenido las siguientes conclusiones:

- *Random Forest* es el mejor estimador para predecir las características. El estimador que consigue en un menor RMSE en la mayoría de los casos estudiados es *Random Forest*.
- La combinación de características de baja correlación con la variable estimada y el empleo de modelos de predicción consiguen mejorar la correlación. Antes de hacer las predicciones, la máxima correlación de las características a estimar con el resto de las variables era en valor absoluto de 0.58 para C_N, 0.47 para CACO3_TOT y 0.65 para WC_FC. En cambio, empleando estimadores se alcanzaron correlaciones de 0.74 para C_N, 0.81 para CACO3_TOT y 0.87 para WC_FC. De esta forma, el incremento del coeficiente de correlación ha estado entre 0.16 para el caso de C_N y 0.34 para el caso de CACO3_TOT.
- Añadir una característica de baja correlación con la variable estimada a un modelo de predicción puede conseguir una reducción del RMSE considerable. En el caso de la estimación de C_N se puede comprobar que añadir la característica TEXT_2000, con un coeficiente de correlación con C_N de 0.22, a la combinación PH y OM consigue reducir el valor del RMSE de 4.89 a 3.97.
- Normalizar los datos no supone una mejoría en el funcionamiento de los estimadores utilizados. No se observan diferencias significativas en los resultados entre los datos sin normalizar y los datos normalizados. Además, la regresión de Poisson no se puede utilizar con los datos normalizados puesto que su dominio de actuación no incluye los valores negativos.

5.2 Líneas futuras

A continuación, se proponen posibles líneas futuras con las que se puede complementar y continuar el trabajo desarrollado en este TFM:

- Utilizar otros métodos de *Machine Learning* para predecir las características como pueden ser las redes neuronales o las máquinas de vector soporte.

- Probar las diferentes configuraciones de las que disponen los métodos empleados, modificando los parámetros por defecto que se han utilizado en este trabajo, para ver si se consiguen mejorar los resultados.
- Realizar más combinaciones con las características escogidas para hacer la predicción.

Además de las propuestas anteriormente mencionadas, que serían líneas futuras enmarcadas en el análisis de los datos utilizados, existen otras líneas derivadas del trabajo de análisis de datos genérico realizado, como son:

- Generalizar el código desarrollado, de modo que permita realizar un análisis rápido similar al realizado en este trabajo cambiando únicamente la fuente de datos y unas pocas opciones de configuración.
- Relacionado con el punto anterior, desarrollar una interfaz de usuario amigable y de alto nivel, que permita realizar esta labor sin necesidad de modificar el código.

Referencias

[1] Y. el Miedany, “Artificial Intelligence,” in *Rheumatology Teaching*, Cham: Springer International Publishing, 2019, pp. 347–378. doi: 10.1007/978-3-319-98213-7_18.

[2] W. Fan *et al.*, “Machine learning applied to the design and inspection of reinforced concrete bridges: Resilient methods and emerging applications,” *Structures*, vol. 33, pp. 3954–3963, 2021, doi: 10.1016/j.istruc.2021.06.110.

[3] T. J. Cleophas and A. H. Zwinderman, “Artificial Intelligence,” in *Statistics Applied to Clinical Studies*, Springer Netherlands, 2012, pp. 627–637. doi: 10.1007/978-94-007-2863-9_58.

[4] C. Zhang and Y. Lu, “Study on artificial intelligence: The state of the art and future prospects,” *Journal of Industrial Information Integration*, vol. 23, p. 100224, Sep. 2021, doi: 10.1016/j.jii.2021.100224.

[5] J. Wojtusiak, “Machine Learning,” in *Encyclopedia of the Sciences of Learning*, N. M. Seel, Ed. Springer US, 2012, pp. 2082–2083. doi: 10.1007/978-1-4419-1428-6.

[6] M. Aria, C. Cuccurullo, and A. Gnasso, “A comparison among interpretative proposals for Random Forests,” *Machine Learning with Applications*, vol. 6, p. 100094, 2021, doi: 10.1016/j.mlwa.2021.100094.

[7] M. van Oijen, “Machine Learning,” in *Bayesian Compendium*, Springer International Publishing, 2020, pp. 141–149. doi: 10.1007/978-3-030-55897-0_20.

[8] F. Camastra and A. Vinciarelli, “Machine Learning,” in *Machine Learning for Audio, Image and Video Analysis: Theory and Applications*, Springer London, 2015, pp. 99–106. doi: 10.1007/978-1-4471-6735-8_4.

[9] G. K. F. Tso and K. K. W. Yau, “Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks,” *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007, doi: 10.1016/j.energy.2006.11.010.

[10] Dodge Yadolah, *The Concise Encyclopedia of Statistics*. Springer New York, 2008. doi: 10.1007/978-0-387-32833-1_348.

[11] M. Müller, “Generalized Linear Models,” in *Handbook of Computational Statistics*, Springer Berlin Heidelberg, 2012, pp. 681–709. doi: 10.1007/978-3-642-21551-3_24.

[12] H. Xie, A. M. Fischer, and P. G. Strutton, “Generalized linear models to assess environmental drivers of paralytic shellfish toxin blooms (Southeast Tasmania, Australia),” *Continental Shelf Research*, vol. 223, p. 104439, 2021, doi: 10.1016/j.csr.2021.104439.

[13] J. K. Lindsey, *Applying Generalized Linear Models*. Springer, 1997. doi: https://doi-org.ponton.uva.es/10.1007/b98856.

[14] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, Second Edition. Chapman and Hall, 1989.

[15] J. T. Belotti *et al.*, "Air pollution epidemiology: A simplified Generalized Linear Model approach optimized by bio-inspired metaheuristics," *Environmental Research*, vol. 191, p. 110106, 2020, doi: 10.1016/j.envres.2020.110106.

[16] Z. Pan, Y. Wang, and Y. Pan, "A new locally adaptive k-nearest neighbor algorithm based on discrimination class," *Knowledge-Based Systems*, vol. 204, p. 106185, 2020, doi: 10.1016/j.knosys.2020.106185.

[17] L. Xiong and Y. Yao, "Study on an adaptive thermal comfort model with K-nearest-neighbors (KNN) algorithm," *Building and Environment*, vol. 202, p. 108026, 2021, doi: 10.1016/j.buildenv.2021.108026.

[18] Y. Dong, X. Ma, and T. Fu, "Electrical load forecasting: A deep learning approach based on K-nearest neighbors," *Applied Soft Computing*, vol. 99, p. 106900, 2021, doi: 10.1016/j.asoc.2020.106900.

[19] D. Delgado Castillo, R. Martín Pérez, L. Hernández Pérez, R. Orozco Morález, and J. Lorenzo Ginori, "Algoritmos de aprendizaje automático para la clasificación de neuronas piramidales afectadas por el envejecimiento," *Revista Cubana de Informática Médica*, vol. 8, no. 3, pp. 559–571, 2016.

[20] scikit-learn 0.24.2 documentation, "scikit-learn: machine learning in Python." https://scikit-learn.org/stable/index.html (accessed Sep. 11, 2021).

[21] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Decision tree regression for soft classification of remote sensing data," *Remote Sensing of Environment*, vol. 97, no. 3, pp. 322–336, 2005, doi: 10.1016/j.rse.2005.05.008.

[22] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random Forests," in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. Springer US, 2012, pp. 157–176. doi: 10.1007/978-1-4419-9326-7_5.

[23] C. Vens, "Random Forest," in *Encyclopedia of Systems Biology*, W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, Eds. Springer New York, 2013, pp. 1812–1813. doi: 10.1007/978-1-4419-9863-7.

[24] European Commission and Joint Research Centre, "European Soil Data Centre (ESDAC)." https://esdac.jrc.ec.europa.eu/ (accessed Aug. 10, 2021).

[25] European Commission and Joint Research Centre, "European Soil Database. Introduction to the European Soil Database (distribution version v2.0)." https://esdac.jrc.ec.europa.eu/ESDB_Archive/ESDBv2/intro.htm#SPADBE (accessed Aug. 10, 2021).

[26] F. Bertrand, “sweetviz - PyPI,” 2021. https://pypi.org/project/sweetviz/ (accessed Sep. 11, 2021).

[27] Y. Liu, “Mean Square Error of Survey Estimates,” in *Encyclopedia of Quality of Life and Well-Being Research*, no. 10, A. C. Michalos, Ed. Springer Netherlands, 2014, pp. 3892–3893. doi: 10.1007/978-94-007-0753-5_1754.

[28] NumPy v1.21 Manual, “numpy.corrcoef,” 2021. https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html (accessed Sep. 11, 2021).

Anexo I: Tablas con las siglas presentes en la base de datos y su significado

En este anexo se recogen las tablas con las siglas que aparecen en EST-PROF y EST-HOR junto con su significado.

A1.1 Siglas utilizadas en EST-PROF

Siglas	Significado
PROF_NUM	Identificador único de perfil
COUNTRY	Código del país
SOIL	Nombre completo (CEC modificado 1985) de la leyenda de la FAO de 1974
SOIL90	Nombre completo del suelo de la leyenda FAO-Unesco de 1990
TEXT1	Clase de textura de superficie dominante
TEXT2	Clase de textura de superficie secundaria
PM	Material padre/primario dominante
SMU	Identificador de la Unidad de Mapeo del Suelo al que se refiere el perfil (dado por el autor)
STU	Identificador de unidad tipográfica de suelo a la que se refiere el perfil (dado por el autor)
LU	Uso de la tierra dominante
GWL_HI	Nivel medio más alto de una capa freática permanente o encaramada
GWL_LO	Nivel medio más bajo de una capa freática permanente o encaramada
DR_EFF_A	Media eficaz de profundidad de las raíces para A (cm)
DR_EFF_BE	Media eficaz de profundidad de las raíces para remolacha (cm)
DR_EFF_CO	Media eficaz de profundidad de las raíces para algodón (cm)
DR_EFF_MA	Media eficaz de profundidad de las raíces para maíz (cm)
DR_EFF_OL	Media eficaz de profundidad de las raíces para aceitunas (cm)
DR_EFF_SC	Media eficaz de profundidad de las raíces para cereales de primavera (cm)
DR_EFF_SG	Media eficaz de profundidad de las raíces para hierba corta (cm)
DR_EFF_WC	Media eficaz de profundidad de las raíces para cereales de invierno (cm)
DR_TOT_A	Media total de profundidad de las raíces para A (cm)
DR_TOT _BE	Media total de profundidad de las raíces para remolacha (cm)
DR_ TOT _CO	Media total de profundidad de las raíces para algodón (cm)
DR_TOT _MA	Media total de profundidad de las raíces para maíz (cm)
DR_TOT _OL	Media total de profundidad de las raíces para aceitunas (cm)
DR_TOT _SC	Media total de profundidad de las raíces para cereales de primavera (cm)
DR_TOT _SG	Media total de profundidad de las raíces para hierba corta (cm)

DR_TOT _WC	Media total de profundidad de las raíces para cereales de invierno (cm)
DEPTH_HORO	Origen de los datos para profundidades de horizontes
BD_O	Origen de los datos para la densidad a granel
BS_O	Origen de los datos para la saturación de la base
C_N_O	Origen de los datos para la relación carbono/nitrógeno
CACO3_TOTO	Origen de los datos para el equivalente de carbonato de calcio
CASO4_O	Origen de los datos para el yeso
CEC_O	Origen de los datos para la capacidad de intercambio catiónico
EC_O	Origen de los datos para la conductividad eléctrica
EXCH_CA_O	Origen de los datos para la base intercambiable de calcio
EXCH_K_O	Origen de los datos para la base intercambiable de potasio
EXCH_MG_O	Origen de los datos para la base intercambiable de magnesio
EXCH_NA_O	Origen de los datos para la base intercambiable de sodio
EXCH_NA_PO	Origen de los datos para el porcentaje de sodio intercambiable de la CEC
GRAVEL_O	Origen de los datos para la clase de porcentaje de piedras y grava en el suelo
OM_O	Origen de los datos para el contenido de materia orgánica
PH_O	Origen de los datos para el pH de acidez
POR_O	Origen de los datos para la porosidad total
STRUCT_O	Origen de los datos para el tipo de estructura
TEXT_2_O	Origen de los datos para la proporción de partículas de tamaño inferior a 2 µm
TEXT_20_O	Origen de los datos para la proporción de partículas de tamaño 2-20 µm
TEXT_50_O	Origen de los datos para la proporción de partículas de tamaño 20-50 µm
TEXT_200_O	Origen de los datos para la proporción de partículas de tamaño 50-200 µm
TEXT_2000O	Origen de los datos para la proporción de partículas de tamaño 200-2000 µm
WC_1_O	Origen de los datos para la retención de agua en el suelo a -1 kPa
WC_10_O	Origen de los datos para la retención de agua en el suelo a -10 kPa
WC_100_O	Origen de los datos para la retención de agua en el suelo a -100 kPa
WC_1500_O	Origen de los datos para la retención de agua en el suelo a -1500 kPa
WC_FC_O	Origen de los datos para la retención de agua en el suelo a capacidad de campo

A1.2 Siglas utilizadas en EST-HOR

Siglas	Significado
PROF_NUM	Identificador único de perfil
HOR_NUM	Número secuencial del horizonte dentro del perfil
HOR_NAME	Nombre del horizonte según el sistema FAO
DEPTHSTART	Inicio (aparición) de la profundidad del horizonte
DEPTHEND	Fin (desaparición) de la profundidad del horizonte
BD	Densidad a granel (g/cm3)
BS	Saturación base (%)
C_N	Relación carbono/nitrógeno (%)
CACO3_TOT	Equivalente de carbonato de calcio (% en peso)
CASO4	Yeso (% en peso)
CEC	Capacidad de intercambio catiónico (cmol+/kg)
EC	Clase de conductividad eléctrica (dS/m rango a 25 °C)
EXCH_CA	Base intercambiable de calcio (cmol+/kg)
EXCH_K	Base intercambiable de potasio (cmol+/kg)
EXCH_MG	Base intercambiable de magnesio (cmol+/kg)
EXCH_NA	Base intercambiable de sodio (cmol+/kg)
EXCH_NA_%	Porcentaje de sodio intercambiable de la CEC (%)
GRAVEL	Porcentaje de clase de piedras y grava en el suelo.
OM	Contenido de materia orgánica en capas órgano-minerales (% en peso)
PH	pH de acidez (en agua 1:2.5)
POR	Porosidad total (%)
STRUCT	Tipo de estructura
TEXT_2	Proporción (%) de partículas de tamaño inferior a 2 µm
TEXT_20	Proporción (%) de partículas de tamaño 2-20 µm
TEXT_50	Proporción (%) de partículas de tamaño 20-50 µm
TEXT_200	Proporción (%) de partículas de tamaño 50-200 µm
TEXT_2000	Proporción (%) de partículas de tamaño 200-2000 µm
WC_1	Retención de agua del suelo a -1 kPa (porcentaje de volumen de agua)
WC_10	Retención de agua del suelo a -10 kPa (porcentaje de volumen de agua)
WC_100	Retención de agua del suelo a -100 kPa (porcentaje de volumen de agua)
WC_1500	Retención de agua del suelo a -1500 kPa (porcentaje de volumen de agua)
WC_FC	Retención de agua del suelo a capacidad de campo (porcentaje de volumen de agua)

Anexo II: Códigos desarrollados en el TFM

Este anexo reúne los códigos realizados en el TFM para mostrar los datos de EST-PROF y de EST-HOR, así como el código para procesar y analizar los datos presentes en EST-HOR.

A2.1 Código para mostrar los datos de EST-PROF

```
import pandas as pd
import sweetviz as sv

# Import data
data_PROF = pd.read_excel('./../data/EST-PROF.xlsx')

# Remove invalid data
for nombre in data_PROF:
        data_PROF.loc[(data_PROF[nombre] == -998) | (data_PROF[nombre] == -
        999), nombre] = float('NaN')

# Select numeric columns
cols = ["DR_EFF_BE", "DR_EFF_MA", "DR_EFF_SC", "DR_EFF_SG", "DR_EFF_WC",
        "DR_TOT_BE", "DR_TOT_MA", "DR_TOT_SC", "DR_TOT_SG", "DR_TOT_WC"]
data_PROF_num = data_PROF[cols]

# Generate report
PROF_report = sv.analyze(data_PROF)
PROF_report.show_html('./html/PROF.html')
PROF_num_report = sv.analyze(data_PROF_num)
PROF_num_report.show_html('./html/PROF_num.html')
```

A2.2 Código para mostrar los datos de EST-HOR

```
import pandas as pd
import sweetviz as sv

# Import data
data_HOR = pd.read_excel('./../data/EST-HOR.xlsx')

# Remove invalid data
for nombre in data_HOR:
        data_HOR.loc[(data_HOR[nombre] == -998) | (data_HOR[nombre] == -999),
        nombre] = float('NaN')

# Select numeric columns
cols = ["DEPTHSTART", "DEPTHEND", "BD", "BS", "C_N", "CACO3_TOT", "CEC",
        "EXCH_CA", "EXCH_K", "EXCH_MG", "EXCH_NA", "EXCH_NA_P", "OM", "PH",
        "POR", "TEXT_2", "TEXT_20", "TEXT_50", "TEXT_200", "TEXT_2000", "WC_1",
        "WC_10", "WC_100", "WC_1500", "WC_FC"]
data_HOR_num = data_HOR[cols]
```

```
# Generate report
HOR_report = sv.analyze(data_HOR)
HOR_report.show_html('./html/HOR.html')
HOR_num_report = sv.analyze(data_HOR_num)
HOR_num_report.show_html('./html/HOR_num.html')
```

A2.3 Código del procesado y análisis de datos de EST-HOR

```
import pandas as pd
from sklearn import linear_model, neighbors, metrics, ensemble, tree
import numpy as np
import sweetviz as sv

# Import data
data_HOR = pd.read_excel('./../data/EST-HOR_random.xlsx')

# Remove invalid data
for nombre in data_HOR:
        data_HOR.loc[(data_HOR[nombre] == -998) | (data_HOR[nombre] == -999),
        nombre] = float('NaN')
data_X_Y = ["PH", "TEXT_2", "TEXT_20", "CACO3_TOT"]
data_HOR = data_HOR.dropna(subset=data_X_Y)


# Normalización y obtención de los índices de las caraterísticas a analizar
data_HORn = (data_HOR - data_HOR.mean()) / data_HOR.std()
idx_data_HOR = [data_HOR.columns.get_loc(col) for col in data_X_Y]
idx_data_HORn = [data_HORn.columns.get_loc(col) for col in data_X_Y]


# Select data to train (65%) and predict (35%)
X_train = data_HOR.iloc[0:int(len(data_HOR)*0.65), [idx_data_HOR[i] for i in
        range(0, len(idx_data_HOR)-1)]]
Y_train = data_HOR.iloc[0:int(len(data_HOR)*0.65), idx_data_HOR[-1]]
X_predict = data_HOR.iloc[int(len(data_HOR)*0.65):len(data_HOR),
        [idx_data_HOR[i] for i in range(0, len(idx_data_HOR)-1)]]
Y_predict = data_HOR.iloc[int(len(data_HOR)*0.65):len(data_HOR),
        idx_data_HOR[-1]]

# Select data normalized to train (65%) and predict (35%)
Xn_train = data_HORn.iloc[0:int(len(data_HORn)*0.65), [idx_data_HORn[i] for i
        in range(0, len(idx_data_HORn)-1)]]
Yn_train = data_HORn.iloc[0:int(len(data_HORn)*0.65), idx_data_HORn[-1]]
Xn_predict = data_HORn.iloc[int(len(data_HORn)*0.65):len(data_HORn),
        [idx_data_HORn[i] for i in range(0, len(idx_data_HORn)-1)]]
Yn_predict = data_HORn.iloc[int(len(data_HORn)*0.65):len(data_HORn),
        idx_data_HORn[-1]]


# Predicción de datos a partir de regresión lineal, normal, poisson, knn, DT y
RF
reg_linear = linear_model.LinearRegression().fit(X_train, Y_train)
Y_estimated_LR = reg_linear.predict(X_predict)
```

```
reg_normal = linear_model.TweedieRegressor(power=0,
        link='identity').fit(X_train, Y_train)
Y_estimated_N = reg_normal.predict(X_predict)
reg_poisson = linear_model.TweedieRegressor(power=1, link='log').fit(X_train,
        Y_train)
Y_estimated_P = reg_poisson.predict(X_predict)
reg_knn = neighbors.KNeighborsRegressor(n_neighbors=10).fit(X_train, Y_train)
Y_estimated_KNN = reg_knn.predict(X_predict)
reg_tree = tree.DecisionTreeRegressor(random_state=0).fit(X_train, Y_train)
Y_estimated_DT = reg_tree.predict(X_predict)
reg_forest = ensemble.RandomForestRegressor(random_state=0).fit(X_train,
        Y_train)
Y_estimated_RF = reg_forest.predict(X_predict)

# Predicción de datos normalizados a partir de regresión lineal, normal, knn,
DT y RF
regN_linear = linear_model.LinearRegression().fit(Xn_train, Yn_train)
Yn_estimated_LR = regN_linear.predict(Xn_predict)
regN_normal = linear_model.TweedieRegressor(power=0,
        link='identity').fit(Xn_train, Yn_train)
Yn_estimated_N = regN_normal.predict(Xn_predict)
regN_knn = neighbors.KNeighborsRegressor(n_neighbors=10).fit(Xn_train,
        Yn_train)
Yn_estimated_KNN = regN_knn.predict(Xn_predict)
regN_tree = tree.DecisionTreeRegressor(random_state=0).fit(Xn_train, Yn_train)
Yn_estimated_DT = regN_tree.predict(Xn_predict)
regN_forest = ensemble.RandomForestRegressor(random_state=0).fit(Xn_train,
        Yn_train)
Yn_estimated_RF = regN_forest.predict(Xn_predict)


# Show comparative between reg and reg_norm
Y_estimated_LR_DesN = Yn_estimated_LR * data_HOR[data_X_Y[-1]].std() +
        data_HOR[data_X_Y[-1]].mean()
Y_estimated_N_DesN = Yn_estimated_N * data_HOR[data_X_Y[-1]].std() +
        data_HOR[data_X_Y[-1]].mean()
Y_estimated_KNN_DesN = Yn_estimated_KNN * data_HOR[data_X_Y[-1]].std() +
        data_HOR[data_X_Y[-1]].mean()
Y_estimated_DT_DesN = Yn_estimated_DT * data_HOR[data_X_Y[-1]].std() +
        data_HOR[data_X_Y[-1]].mean()
Y_estimated_RF_DesN = Yn_estimated_RF * data_HOR[data_X_Y[-1]].std() +
        data_HOR[data_X_Y[-1]].mean()

print(f"\nRMSE {data_X_Y[-1]} Reg Lineal:
        {metrics.mean_squared_error(Y_estimated_LR, Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Normal:
        {metrics.mean_squared_error(Y_estimated_N, Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Poisson:
        {metrics.mean_squared_error(Y_estimated_P, Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg KNN:
        {metrics.mean_squared_error(Y_estimated_KNN,
        Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Decision Tree:
        {metrics.mean_squared_error(Y_estimated_DT, Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Random Forest:
        {metrics.mean_squared_error(Y_estimated_RF, Y_predict,squared=False)}")
```

```
print(f"RMSE {data_X_Y[-1]} Reg Lineal normalizada:
       {metrics.mean_squared_error(Y_estimated_LR_DesN,
       Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Normal normalizada:
       {metrics.mean_squared_error(Y_estimated_N_DesN,
       Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg KNN normalizada:
       {metrics.mean_squared_error(Y_estimated_KNN_DesN,
       Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Decision Tree normalizada:
       {metrics.mean_squared_error(Y_estimated_DT_DesN,
       Y_predict,squared=False)}")
print(f"RMSE {data_X_Y[-1]} Reg Random Forest normalizada:
       {metrics.mean_squared_error(Y_estimated_RF_DesN,
       Y_predict,squared=False)}")

print(f"\nCoef corr {data_X_Y[-1]} Reg Lineal: {np.corrcoef(Y_estimated_LR,
       Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Normal: {np.corrcoef(Y_estimated_N,
       Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Poisson: {np.corrcoef(Y_estimated_P,
       Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg KNN: {np.corrcoef(Y_estimated_KNN,
       Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Decision Tree:
       {np.corrcoef(Y_estimated_DT, Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Random Forest:
       {np.corrcoef(Y_estimated_RF, Y_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Lineal normalizada:
       {np.corrcoef(Yn_estimated_LR, Yn_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Normal normalizada:
       {np.corrcoef(Yn_estimated_N, Yn_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg KNN normalizada:
       {np.corrcoef(Yn_estimated_KNN, Yn_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Decision Tree normalizada:
       {np.corrcoef(Yn_estimated_DT, Yn_predict)[0,1]}")
print(f"Coef corr {data_X_Y[-1]} Reg Random Forest normalizada:
       {np.corrcoef(Yn_estimated_RF, Yn_predict)[0,1]}")


# Se añade a X_predict los datos deseados para mostrarlos posteriormente
X_predict.insert(X_predict.shape[1], "{0}".format(data_X_Y[-1]), Y_predict)
X_predict.insert(X_predict.shape[1], "{0}n".format(data_X_Y[-1]), Yn_predict)
X_predict.insert(X_predict.shape[1], "{0}_RegLin".format(data_X_Y[-1]),
       Y_estimated_LR)
X_predict.insert(X_predict.shape[1], "{0}_RegNorm".format(data_X_Y[-1]),
       Y_estimated_N)
X_predict.insert(X_predict.shape[1], "{0}_Poisson".format(data_X_Y[-1]),
       Y_estimated_P)
X_predict.insert(X_predict.shape[1], "{0}_knn".format(data_X_Y[-1]),
       Y_estimated_KNN)
X_predict.insert(X_predict.shape[1], "{0}_DTree".format(data_X_Y[-1]),
       Y_estimated_DT)
X_predict.insert(X_predict.shape[1], "{0}_RForest".format(data_X_Y[-1]),
       Y_estimated_RF)
X_predict.insert(X_predict.shape[1], "{0}n_RegLin".format(data_X_Y[-1]),
       Yn_estimated_LR)
```

```
X_predict.insert(X_predict.shape[1], "{0}n_RegNorm".format(data_X_Y[-1]),
        Yn_estimated_N)
X_predict.insert(X_predict.shape[1], "{0}n_knn".format(data_X_Y[-1]),
        Yn_estimated_KNN)
X_predict.insert(X_predict.shape[1], "{0}n_DTree".format(data_X_Y[-1]),
        Yn_estimated_DT)
X_predict.insert(X_predict.shape[1], "{0}n_RForest".format(data_X_Y[-1]),
        Yn_estimated_RF)

# Show data
predict_report = sv.analyze(X_predict)
predict_report.show_html('./html/Reg/HOR_{0}_TEXT2_TEXT2000_OM.html'.format(da
        ta_X_Y[-1]))
```