|  |
| --- |
|  |

|  |
| --- |
| Pre-Fall – Sistema inteligente para la prevención y predicción de caídas |

|  |
| --- |
| **E4.3 – Modelos corregidos** |

|  |  |
| --- | --- |
| Proyecto | Pre-Fall – Sistema inteligente para la prevención y predicción de caídas |
| Entregable | E4.3 – Modelos corregidos |

Contenido

[Contenido 1](#_Toc151992852)

[Índice de figuras 2](#_Toc151992853)

[1 Introducción 3](#_Toc151992854)

[2 Resumen del módulo de entrenamiento de modelos 4](#_Toc151992855)

[3 Implementación del módulo 6](#_Toc151992856)

Índice de figuras

[Ilustración 1. Preprocesado y división del conjunto de datos en train y test. 4](#_Toc151992921)

[Ilustración 2. Entrenamiento de todos los modelos. 5](#_Toc151992922)

[Ilustración 3. Análisis y generación del mejor modelo 6](#_Toc151992923)

# Introducción

Este entregable está enmarcado en la tarea “T4.3: Corrección y adecuación de los modelos”, perteneciente al paquete de trabajo “PT4 – Pruebas y validación de la solución” dentro del proyecto PRE-FALL. En este documento se describirán las diferentes secciones del módulo de reentrenamiento de modelos que permite generar nuevos modelos de aprendizaje automático entrenados con los nuevos datos disponibles.

# Resumen del módulo de entrenamiento de modelos

Como se ha mencionado anteriormente el módulo de reentrenamiento “model\_train.py” se encarga de generar modelos de machine learning en formato .pkl de manera que estos se adecuen a los nevos datos que se vayan obteniendo. El funcionamiento general de este módulo consiste en leer todos los ficheros de datos existentes y generar una tabla resumen de cada uno de ellos con las métricas más importante. Una vez generado este conjunto resumen, se divide este conjunto en dos subconjuntos: un subconjunto de entrenamiento con el 75% de los datos y otro de test con el restante 25%.

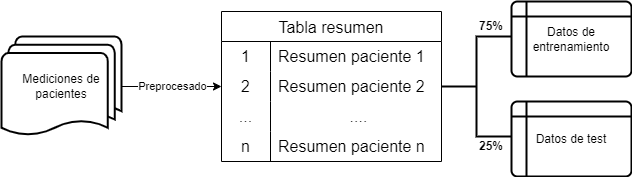


Ilustración 1. Preprocesado y división del conjunto de datos en train y test.

Una vez generados ambos subconjuntos se procede a entrenar varios modelos de machine learning con el mismo conjunto de entrenamiento. La lista de modelos es la siguiente:

* Random Forest
* XGBoost
* Regresión Logística
* Support Vector Machine
* Decision Tree
* K-vecinos

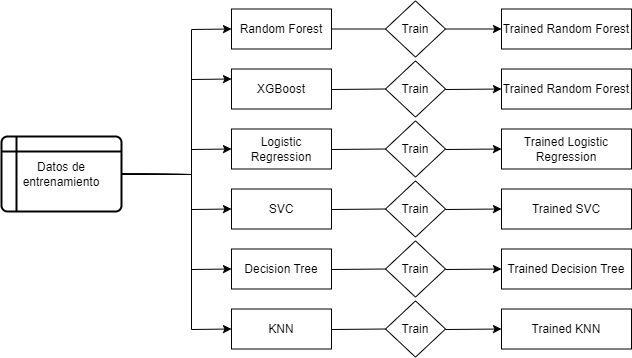


Ilustración 2. Entrenamiento de todos los modelos.

A continuación, se mide el rendimiento de todos estos modelos utilizando el conjunto de test. Cada uno de los modelos entrenados trata de predecir el valor de los datos de test y se obtienen métricas de los resultados. Las métricas utilizadas son:

* Accuracy: Mide la proporción de predicciones correctas del total de casos. Es la métrica más intuitiva y se calcula como el número de predicciones correctas dividido por el número total de predicciones. Se calcula como:
* Recall: mide la proporción de verdaderos positivos identificados correctamente del total de casos reales en esa clase. Se calcula como:
* Precision: mide la proporción de predicciones correctas de una clase en particular (por ejemplo, la clase "positiva") del total de predicciones hechas para esa clase.
* F1: El puntaje F1 es una medida que combina la precisión y el recall en una sola métrica. Es la media armónica de la precisión y el recall, proporcionando un balance entre ambas.

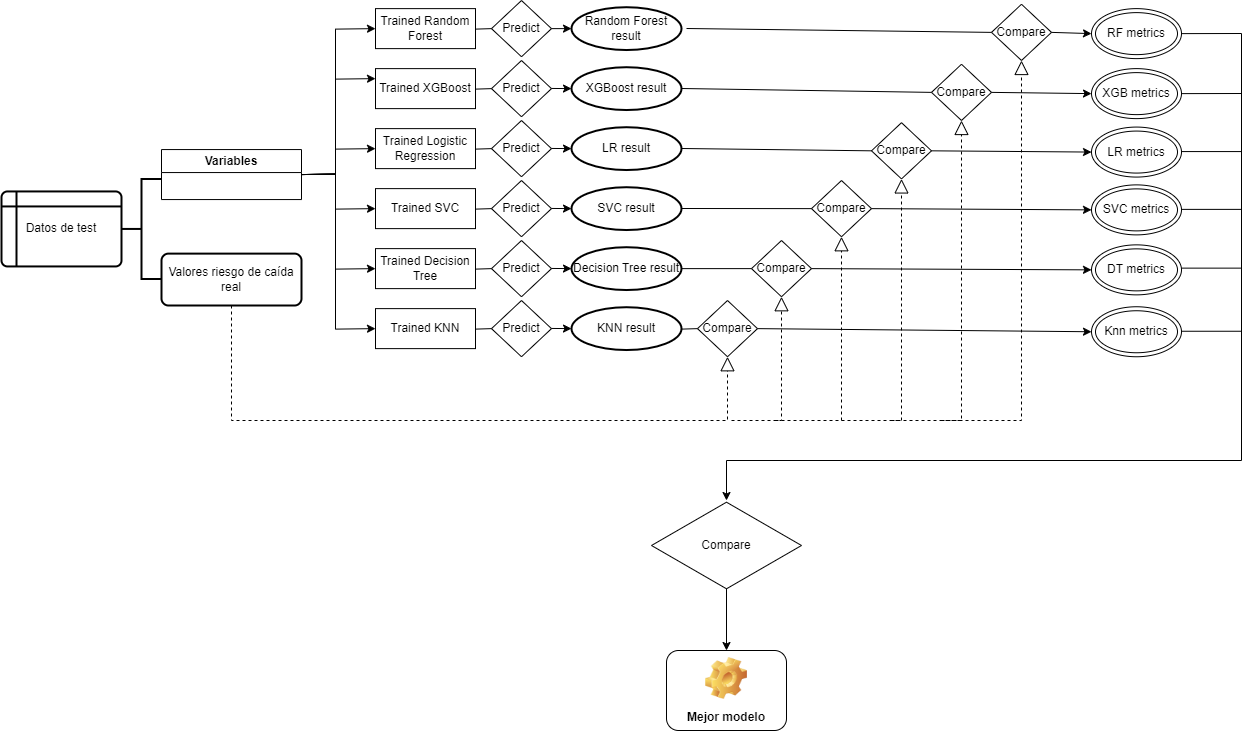


Ilustración 3. Análisis y generación del mejor modelo

El modelo que obtenga el mejor valor medio de todas estas métricas será elegido como el mejor de todos y sobreescribirá el modelo que se emplee actualmente de manera que en posteriores estimaciones del riesgo de caída de pacientes se empleará este modelo.

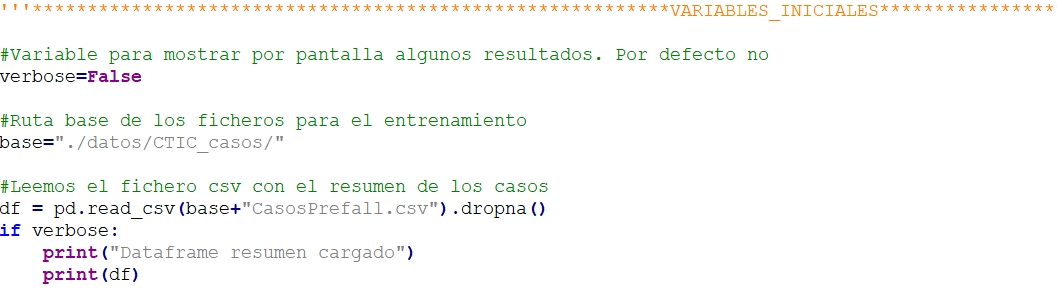
# Implementación del módulo

En esta sección se procede a explicar la implementación de la metodología anterior y cada una de sus secciones a nivel de código.

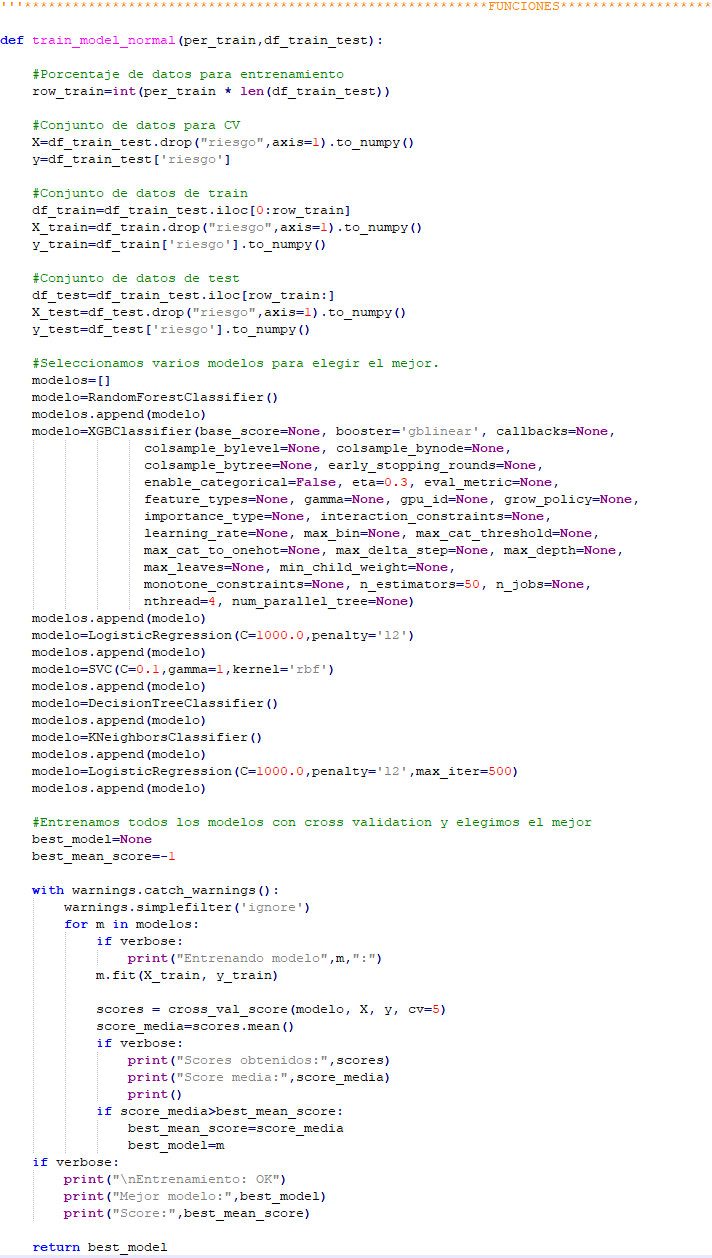


En primer lugar, se importan todas las librerías necesarias. En este caso destacan:

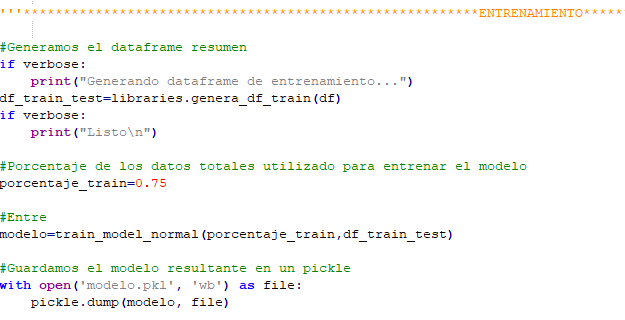
* os y io: Estos módulos son utilizados para interactuar con el sistema operativo y para realizar operaciones de entrada y salida, como leer y escribir archivos.
* Libraries: Este es el módulo que contiene todas las funciones auxiliares necesarias para preparar y preprocesar los datos de entrada así como generar las fases de la marcha y sus métricas.
* Librerías de sklearn.ensemble que contienen todos los modelos que se van a entrenar y analizar.
* sklearn.metrics: Este módulo incluye funciones de puntuación, métricas de rendimiento y emparejamiento de conjuntos de datos, como accuracy\_score, classification\_report y confusion\_matrix, que se utilizan para evaluar la calidad de un modelo.
* sklearn.model\_selection: Incluye herramientas para dividir los datos (train\_test\_split) y para realizar validación cruzada (cross\_val\_score), que son esenciales para el entrenamiento y la evaluación de modelos de aprendizaje automático.
* pickle: Es una forma de serializar y deserializar objetos en Python, lo que permite guardar modelos entrenados en un archivo y luego cargarlo más tarde.



La siguiente sección se encarga de definir el directorio donde se encuentran todos ficheros con los que se van a entrenar los modelos así como leer el resumen de los mismos para asignar a cada uno de ellos información adicional como el diagnóstico previo de riesgo de caída. Además se proporciona una variable “verbose” para indicar si se quieren mostrar por consola mensajes de información del progreso del programa lo cual puede ser útil para un posterior debug.



A continuación, se define la función que se encarga de entrenar cada uno de los modelos propuestos. Esta función crea en primer lugar una lista con cada uno de los modelos incluyendo algunos hiperparámetros predefinidos. Después, se entrena cada uno de ellos con una copia de los datos de entrenamiento previamente creados y se obtiene una puntuación de cada una de las métricas comentadas en el apartado anterior y se calcula su valor medio. Finalmente de entre todos ellos se elige aquel modelo con la puntuación media mayor de entre todos ellos y se retorna.



La sección final del módulo es la que realmente realiza cada una de las tareas empezando por la creación del conjunto de datos preprocesado y limpio de datos nulos o extremos a través de la librería “libraries” comentada anteriormente. A continuación, se determina un porcentaje de datos que será empleado para entrenar los modelos siendo este valor por defecto del 75%.

Seguidamente se llama al método “train\_model\_normal” definido anteriormente y este devuelve el modelo que mejores métricas obtiene para los datos de entrenamiento actuales.