Enginyeria
UAB Universitat Autònoma de Barcelona

Universidad Autónoma de Barcelona

Escuela de Ingeniería

Grado en Ingeniería de Datos

Sistema Interactivo para el Seguimiento de Salud y Predicción de Hábitos

Alumno: Pau Montesinos Cáliz

Tutor: Oriol Cortés Comellas

Fecha: 20/04/2025

1. Objetivos

El objetivo principal de este proyecto es el desarrollo de una plataforma capaz de recoger hábitos y datos sobre las personas con el fin de analizar su estado de salud, estimar la probabilidad de padecer ciertas enfermedades y ofrecer recomendaciones personalizadas para mejorar sus hábitos. En el primer informe, se ofrecieron unos objetivos a cumplir y, varios de ellos, han sido ya solventados:

- Identificar uno o varios datasets útiles: han sido identificados tres datasets en total. Uno de ellos podría acabar descartándose, pero no habría problema si no se acaba realizando, pues este ya tiene su código totalmente definido.
- Revisar y limpiar las impurezas de los datasets escogidos: por cada dataset se han identificado las columnas necesarias de cambios y se ha aplicado la limpieza correspondiente.
- Utilizar los datos para entrenar varios modelos de predicción tanto de Machine Learning como de Deep Learning: por el momento, se han creado los modelos basados en Machine Learning y los basados en Deep Learning se crearán en un futuro cercano.

Hechos estos avances, sigue habiendo varios objetivos más a cumplir, más es necesario sumar algunos objetivos a los que ya teníamos a raíz del proceso que se ha ido realizando en el trabajo:

- Identificación de mejores parámetros de entrenamiento de los modelos creados.
- Identificación de mejores estrategias para conseguir métricas con valores mayores a los obtenidos.
- Creación y entrenamiento de modelos de Deep Learning.
- Aplicación de estrategias identificadas previamente y creación y desarrollo de nuevas.
- Evaluación de estrategias y resultados.
- Desarrollo de la plataforma que contenga todos los modelos creados anteriormente.

2. Metodología

Para la realización del proyecto se ha decidido usar una metodología ágil, concretamente Kanban, con la cual se ha podido organizar el trabajo en pequeñas tareas, permitiendo de esta manera mejorar y evaluar el trabajo realizado hasta el momento.

En cada tarea se ha definido un objetivo claro, por lo que se ha podido estructurar el proyecto de una forma más sencilla y así, por lo tanto, adaptarse de mejor manera a posibles imprevistos.

Por el momento, se ha estructurado el proyecto de una forma modular. De esta manera, cada módulo es independiente de los demás. Esto, además de ser una forma de tener estructurado el código, es una manera también de facilitar la adaptación a cambios futuros, debido a la posibilidad de corrección de errores e implementación de mejoras a sabiendas de que, en todo momento, el resto del sistema no estará comprometido.

Para realizar todo lo descrito, se han utilizado varias herramientas que han facilitado tanto la escritura de código como su guardado. Aquí se describen las más importantes:

- PyCharm: El entorno de desarrollo escogido gracias a su simple conexión con GitHub.
- Python: El lenguaje de desarrollo de código escogido debido a su cantidad de librerías importantes para el desarrollo de modelos de Machine y Deep Learning.
- Sklearn: Biblioteca escogida para entrenar los modelos de Machine Learning.
- XGBoost: Biblioteca escogida para el entrenamiento del modelo de Machine Learning que Sklearn no abarcaba.
- GitHub: Importante repositorio en la nube para tener siempre el código a buen recaudo y accesible desde cualquier dispositivo con conexión a internet.

Además, en un futuro, se utilizarán más herramientas para el desarrollo de los modelos de Deep Learning y la plataforma:

- Keras: Mediante Keras, se escribirá el código del modelo de Deep Learning.
- HTML/CSS: La plataforma será desarrollada mediante estos dos lenguajes.
- Flask: Se creará la plataforma procesando los scripts HTML y CSS mediante este Framework de Python
- Google Colab: Se entrenarán los modelos de Deep Learning utilizando las GPUs de Google Colab.

3. Desarrollo

Para la realización del progreso actual se ha creado un repositorio en GitHub que contiene el código actualizado. Este desarrollo se está llevando a cabo en la rama de desarrollo y se hará un *merge* a la rama definitiva cuando el código esté totalmente madurado.

Dentro del repositorio, podremos ver un conjunto de paquetes de Python con todas las funciones necesarias para el entrenamiento final de los modelos (de Machine Learning por el momento). En este caso particular, los archivos más destacables serían:

- **utils.py**: Archivo que contiene funciones generales reutilizables a lo largo de los diferentes módulos que forman el proyecto.

- **Limpieza de datos**: Se han creado tres archivos diferentes con el nombre de clean.py. Cada uno de estos archivos se encarga de hacer el proceso de limpieza de uno de los datasets para el posterior consumo de sus datos.

- **Modelos de Machine Learning**: Tres archivos independientes para los distintos modelos utilizados: LogisticRegression.py, RandomForest.py y XGBoost.py. Cada uno de estos archivos contiene una clase que ayuda a realizar las funcionalidades básicas de un modelo: entrenamiento, predicción y evaluación. Estas clases también contienen funcionalidades de guardado y carga de modelos, por lo que un modelo funcional se puede guardar para ser usado en un futuro y no tener que entrenarlo de nuevo.

3.1 Datasets utilizados

Para llevar el proyecto a cabo, se han utilizado un total de 3 datasets sobre los cuales se harán los entrenamientos y evaluaciones pertinentes. Los datasets son:

- **HeartDisease.csv:** En este dataset se recoge el registro de datos y hábitos de personas diferentes (un total de 18 columnas por cada una de las personas). Datos que a simple vista pueden parecer relevantes, aunque todos ellos lo son, podrían ser si fuman o beben, el sexo de la persona, la edad o si han tenido alguna enfermedad anteriormente como podría ser un derrame cerebral o diabetes. Mediante este dataset se intentará predecir si una persona tiene una enfermedad del corazón. Para ver si se ha acertado, el dataset también indica si la persona tiene esta enfermedad o no.
- **Stroke.csv:** En este dataset, igual que en el anterior, se recogen datos y hábitos sobre distintas personas (un total de 12 columnas). En este caso, tenemos datos igual datos relevantes, como podría ser el estado de fumador, el sexo de la persona o distintas enfermedades que tienen, más alguno de ellos podría ser eliminado como veremos más adelante.
- **Cancer.csv:** Este dataset es muy diferente al de los demás. En los anteriores se quería predecir si tenían o no una enfermedad. En este caso, se parte del hecho de que tienen un cáncer y se quiere saber si este es benigno o maligno. Para esto, lo que se tiene es un dataset con las características del cáncer (simetría, radio, perímetro…), un total de 31 columnas. Este tipo de predicción es interesante ya que, de esta manera, se puede tratar el proyecto desde diferentes puntos de vista.

3.2 Limpieza de datasets

Para cada uno de los datasets se ha definido una limpieza distinta:

- Cancer.csv:
  - Se cambian los valores de diagnosis (M,B) por 1,0 para que los modelos funcionen correctamente.
  - Todos los valores 0.0, que en este caso son valores faltantes, se transforman en NA y posteriormente se eliminan.
- Stroke.csv:
  - Se cambian los valores de “smoking_status” por (-1,0,1,2), siendo -1 el estado de Unknown.
  - Se rellenan los NA dentro de la columna “bmi” por su mediana.
  - Se elimina la columna “id” ya que no tiene ningún tipo de aporte.
  - Se factorizan las columnas “gender”, “Residence_type” y “work_type”.
- HeartDisease.csv:
  - Se cambian los valores de rango de edad por el valor mediano del rango
  - Las columnas “GenHealth” y “Diabetic” se mapean para cambiar sus valores por números ascendentes según el valor de cada fila
  - Las columnas “Smoking”, “AlcoholDrinking”, “HeartDisease”, “Stroke”, “SkinCancer”, “KidneyDisease”, “Asthma” y “DiffWalking” se mapean según si son yes (1) o no (0)
  - Las columnas “Sexo” y “Raza” se factorizan

3.3 Modelos creados

Se han creado un total de 3 modelos. Todos los modelos tienen los mismos métodos y lo que cambia entre ellos son los parámetros con los cuales se entrena:

- Train: Entrena el modelo con diferentes parámetros para buscar los mejores y luego devuelve el modelo entrenado con mejores resultados.
- Predict: predice los valores de un conjunto según el modelo entrenado. Falla si el modelo aún no ha sido entrenado.
- Predict_proba: predice las probabilidades de clasificación de un conjunto en cada uno de sus posibles valores.
- Evaluate: Si la predicción no está hecha la hace y luego da un resumen de las métricas resultantes de la predicción. Si las predicciones ya se le proporcionan, directamente da las métricas.
- Save_model: Guarda el modelo entrenado en un archivo pickle.
- Load_model: Carga un modelo entrenado.

Los modelos entrenados y las razones por las cuales han sido escogidos son:

- Logistic Regression:
    - Es rápido y fácil de entrenar.
    - Conforma una buena base para tomar como referencia.

- Random Forest:
    - Reduce el overfitting.
    - Suele ser bastante estable.

- XGBoost:
    - Es uno de los modelos más potentes actualmente.
    - Corrige errores anteriores.
    - Ofrece una alta precisión en gran variedad de casos.

3.4 Visualización de resultados

Por último, los resultados se mostrarán por pantalla según el modelo entrenado. Por el momento, se ha decidido que lo mejor para comprobar que los resultados son buenos es hacer una matriz de confusión y mirar un reporte de las métricas asociadas:

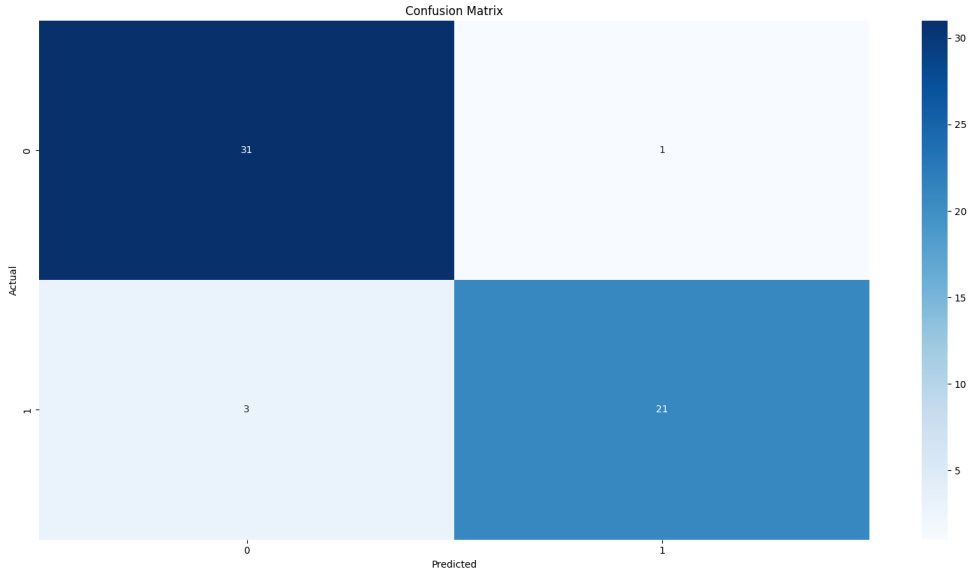

Figura 1. Ejemplo de matriz de confusión de los resultados de las predicciones

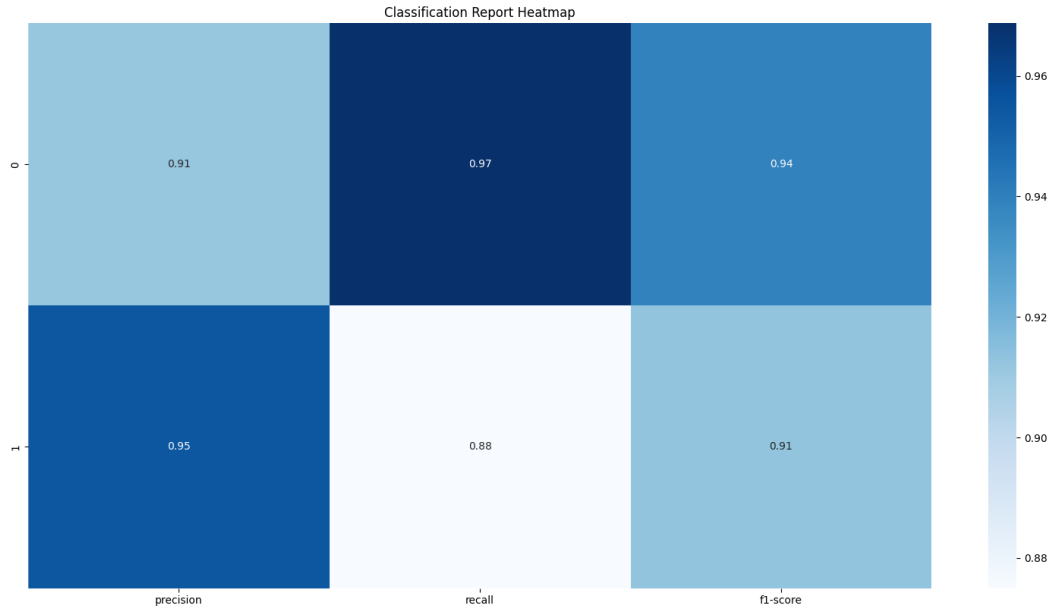

Figura 2. Ejemplo de métricas según la predicción de la matriz de la Figura 1

4. Planificación

La planificación propuesta en el informe inicial sigue su cauce sin mayores problemas. En primer lugar, se han detectado los datasets que ya se han mencionado anteriormente y se ha hecho una limpieza sobre ellos. Más tarde se crearon los modelos de Machine Learning y, por último, se evaluaron mediante los datasets escogidos. Hecho esto, quedará realizar el mismo proceso, pero con modelos de Deep Learning, y crear la plataforma donde se podrá visualizar

el resultado del modelo escogido finalmente. Todos estos pasos son definidos a continuación en un proyecto KanBan desarrollado con la herramienta Trello.

Planificación del proyecto

De igual manera, este diagrama de Gantt indica en que punto de la planificación está el proyecto, lo que ya se ha acabado y lo que aún queda por hacer:

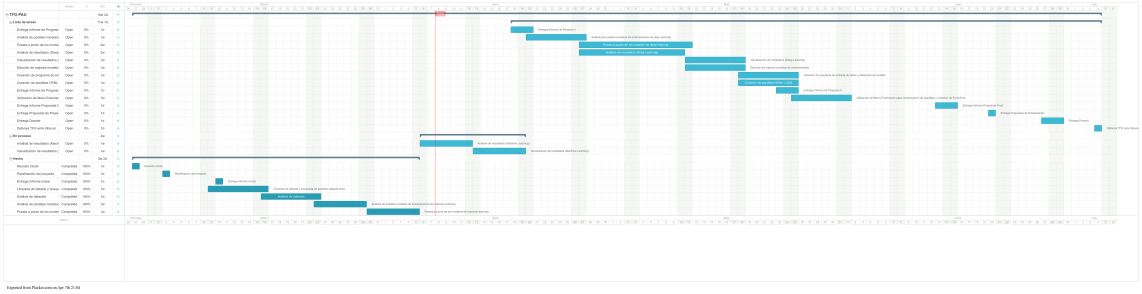

Figura 3. Diagrama de Gantt del proyecto.

5. Bibliografía

[1] Pandas, “pandas - Python Data Analysis Library,” *Pydata.org*, 2024. https://pandas.pydata.org/ (accessed Mar. 05, 2025).

[2] S. Developers, “scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation,” *Scikit-learn.org*, 2025. https://scikit-learn.org/stable/ (accessed Mar. 05, 2025).

[3] WHATWG, “HTML Standard,” *Whatwg.org*, 2025. https://html.spec.whatwg.org/ (accessed Mar. 05, 2025).

[4] W3C, “Cascading Style Sheets,” *W3.org*, 2024. https://www.w3.org/Style/CSS/ (accessed Mar. 05, 2025).

[5] P. Projects, “Welcome to Flask — Flask Documentation (3.1.x),” *Palletsprojects.com*, 2025. https://flask.palletsprojects.com/en/stable/ (accessed Mar. 05, 2025).

[6] Python Software Foundation, “sqlite3 — DB-API 2.0 interface for SQLite databases,” *Python documentation*, 2025. https://docs.python.org/3/library/sqlite3.html (accessed Mar. 05, 2025).

[7] Plotly, “Plotly,” *Plotly.com*, 2025. https://plotly.com/python/ (accessed Mar. 05, 2025).

[8] K. Team, “Keras documentation: Keras 3 API documentation,” *Keras.io*, 2025. https://keras.io/api/ (accessed Mar. 05, 2025).

[9] P. Team, “PyTorch documentation — PyTorch 2.6 documentation,” *Pytorch.org*, 2023. https://pytorch.org/docs/stable/index.html (accessed Mar. 05, 2025).

[10] Gaurav, “An Introduction to Gradient Boosting Decision Trees - Machine Learning Plus,” *Machine Learning Plus*, Jun. 12, 2021. https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees/ (accessed Mar. 06, 2025).
[11] IBM and E. K. Erika Russi, “XGBoost,” *Ibm.com*, May 09, 2024. https://www.ibm.com/es-es/think/topics/xgboost (accessed Mar. 06, 2025).
[12] Z. Xiang, Z. Zhang, and Q. Liu, “Sparse Attention-Based Neural Networks for Code Classification,” *arXiv.org*, 2023. https://arxiv.org/abs/2311.06575 (accessed Mar. 08, 2025).
[13] S. Developers, “sklearn.model_selection,” *scikit-learn*, 2025. https://scikit-learn.org/stable/api/sklearn.model_selection.html (accessed Mar. 19, 2025).
[14] Python Software Foundation, “typing — Support for type hints,” *Python documentation*, 2025. https://docs.python.org/3/library/typing.html (accessed Mar. 19, 2025).
[15] S. Developers, “LogisticRegression,” *scikit-learn*, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed Mar. 21, 2025).
[16] S. Developers, “RandomForestClassifier,” *scikit-learn*, 2025. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Mar. 22, 2025).
[17] Xgb. Developers, “XGBoost Documentation — xgboost 3.0.0 documentation,” *Readthedocs.io*, 2025. https://xgboost.readthedocs.io/en/release_3.0.0/ (accessed Mar. 24, 2025).