**Documento de Requerimientos para el Desarrollo del Sistema de Clasificación de Niveles de Obesidad**

**Proyecto:** Desarrollo de un Sistema de Clasificación de Niveles de Obesidad  
**Fecha:** 23 de mayo de 2024  
**Versión:** 1.0

**1. Introducción**  
Este documento define los requerimientos para el desarrollo de un sistema de software que implemente un modelo de Machine Learning para la clasificación multiclase de niveles de obesidad. El sistema constará de un modelo de ML, una interfaz de usuario (frontend), una lógica de servidor (backend) y una base de datos para la persistencia (opcional para MVP).

**2. Alcance del Proyecto (MVP - Producto Mínimo Viable)**  
El MVP se centrará en permitir a un usuario ingresar sus datos a través de una interfaz web simple, obtener una predicción de su nivel de obesidad (NObeyesdad) y una breve explicación o factores influyentes.

**3. Requerimientos Funcionales**

**3.1. Modelo de Machine Learning (ML)**

* **RF-ML-001:** El sistema deberá implementar un modelo de clasificación multiclase entrenado para predecir la variable NObeyesdad.
* **RF-ML-002:** El modelo deberá aceptar como entrada las siguientes características (tras obtener el diccionario de datos completo):
  + Gender (Categórica)
  + Age (Numérica)
  + Height (Numérica, ej. en metros)
  + Weight (Numérica, ej. en kg)
  + family\_history\_with\_overweight (Categórica: yes/no)
  + FAVC (Categórica/Numérica discreta - *pendiente definición*)
  + FCVC (Numérica discreta/Ordinal - *pendiente definición*)
  + NCP (Numérica discreta/Ordinal - *pendiente definición*)
  + CAEC (Categórica: Sometimes, Frequently, Always, no - *verificar si hay otros valores*)
  + SMOKE (Categórica: yes/no)
  + CH2O (Numérica discreta/Ordinal - *pendiente definición*)
  + SCC (Categórica: yes/no)
  + FAF (Numérica discreta/Ordinal - *pendiente definición*)
  + TUE (Numérica discreta/Ordinal - *pendiente definición*)
  + CALC (Categórica: Sometimes, Frequently, no - *verificar si hay otros valores*)
  + MTRANS (Categórica: Public\_Transportation, Automobile, Walking, etc.)
* **RF-ML-003:** El modelo deberá producir como salida una de las 7 clases de NObeyesdad.
* **RF-ML-004:** El sistema deberá proveer un indicador de la importancia de las características para la predicción realizada (ej. utilizando SHAP values o similar).
* **RF-ML-005 (Post-MVP):** El sistema deberá permitir el reentrenamiento y actualización del modelo con nuevos datos o mejoras algorítmicas.

**3.2. Frontend (Interfaz de Usuario - Web)**

* **RF-FE-001:** El sistema deberá presentar una página web para la entrada de datos del usuario correspondientes a las *features* listadas en RF-ML-002.
  + Los campos de entrada deberán ser intuitivos (ej. selectores para opciones categóricas, campos numéricos para edad, peso, altura).
* **RF-FE-002:** Tras el envío de los datos, el sistema deberá mostrar en la interfaz:
  + La clase de NObeyesdad predicha.
  + (Opcional para MVP, deseable) Un breve mensaje interpretativo o los factores más influyentes en la predicción.
* **RF-FE-003:** La interfaz deberá ser responsiva y usable en navegadores web modernos (desktop y móvil).
* **RF-FE-004 (Post-MVP):** Mostrar información adicional, como recomendaciones generales basadas en la clase predicha.

**3.3. Backend (Lógica del Servidor)**

* **RF-BE-001:** El backend deberá exponer un API endpoint (ej. /predict) que reciba los datos del usuario desde el frontend.
* **RF-BE-002:** El backend deberá realizar la validación de los datos de entrada.
* **RF-BE-003:** El backend deberá preprocesar los datos de entrada de la misma forma que se hizo durante el entrenamiento del modelo (codificación de categóricas, escalado de numéricas).
* **RF-BE-004:** El backend deberá cargar el modelo de ML entrenado y realizar la inferencia (predicción) utilizando los datos preprocesados.
* **RF-BE-005:** El backend deberá formatear la respuesta (clase predicha, importancia de características) y enviarla al frontend.
* **RF-BE-006 (Post-MVP):** Implementar endpoints para la gestión de usuarios y almacenamiento de historiales de predicción.

**3.4. Base de Datos (Opcional para MVP, necesario para Post-MVP)**

* **RF-DB-001 (Post-MVP):** El sistema deberá permitir almacenar los datos de entrada de los usuarios y sus predicciones (con el debido consentimiento).
* **RF-DB-002 (Post-MVP):** El sistema deberá permitir almacenar perfiles de usuario si se implementa funcionalidad de login.
* **RF-DB-003:** La base de datos (si se implementa) deberá asegurar la integridad y seguridad de los datos.

**4. Requerimientos No Funcionales**

* **RNF-001 (Rendimiento):** El tiempo de respuesta desde el envío de datos hasta la visualización de la predicción en el frontend no deberá exceder los 5 segundos para el MVP.
* **RNF-002 (Seguridad):** Se deberán aplicar prácticas básicas de seguridad para proteger la API y cualquier dato almacenado (HTTPS, validación de entradas). La privacidad de los datos del usuario es primordial.
* **RNF-003 (Usabilidad):** La interfaz de usuario deberá ser clara, intuitiva y fácil de usar sin necesidad de entrenamiento extensivo.
* **RNF-004 (Mantenibilidad):** El código fuente (frontend, backend, scripts de ML) deberá estar bien estructurado y comentado para facilitar futuras modificaciones y mantenimiento.
* **RNF-005 (Escalabilidad - Post-MVP):** La arquitectura deberá permitir una futura escalabilidad para manejar un mayor número de usuarios concurrentes.

**5. Dependencias y Bloqueadores**

* **DEP-001 (CRÍTICO):** **Obtención de un diccionario de datos completo y preciso para todas las variables del dataset.** Sin esto, el preprocesamiento y la interpretación del modelo no pueden realizarse correctamente.
* **DEP-002:** Definición de las métricas de éxito para la evaluación del modelo (ej. umbral mínimo de F1-score).

**6. Tecnologías Sugeridas (Flexibles)**

* **Modelo ML:** Python (Scikit-learn, XGBoost, LightGBM, TensorFlow/Keras).
* **Backend:** Python (Flask, FastAPI, Django) o Node.js (Express).
* **Frontend:** JavaScript (React, Vue, Angular) o HTML/CSS/JS simple para MVP.
* **Base de Datos (Post-MVP):** PostgreSQL, MySQL, MongoDB.
* **Control de Versiones:** Git.

**7. Próximos Pasos Inmediatos**

1. **Prioridad Máxima:** Conseguir y documentar el diccionario de datos.
2. Refinar las *features* de entrada y su preprocesamiento.
3. Desarrollar y entrenar una primera versión del modelo de ML.
4. Diseñar y desarrollar el frontend básico para la entrada de datos y visualización de resultados.
5. Desarrollar el API backend para conectar el frontend con el modelo.