Registro Diario de Avances – Modelo basado en IA para detección de somnolencia

**Fecha:** 23/05/2025

**Autor: Tirza Buendia**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el seguimiento diario del Sprint 0.2 del proyecto modelo basado en IA para detección de somnolencia, centrado en el desarrollo de funcionalidades básicas. Se incluyen actividades diarias, tareas completadas, evidencias de código, un burndown chart y un análisis del rendimiento del equipo.

**2. Lineamientos de Diseño**

Desarrollar una primera versión funcional del modelo basado en IA para detección de somnolencia en conductores que permita a los usuarios:

* Usar detección facial en el modelo.
* Monitorear el cierre ocular.

# 3. Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-0.3 | Entrenamiento del Modelo | 8 | 23/05 | 25/05/2025 |
| HU-0.4 | Integración del Modelo al Flujo Actual | 5 | 26/05 | 28/05/2025 |

## 3.1 Historia de Usuario 0.2 – Recolección y etiquetado del dataset

# 4. Evidencias de Código

## 4.1 Historia de Usuario 0.4 — Integración del Modelo al Flujo Actual

Archivo: Modelo/modelo\_xgb.py

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from xgboost import XGBClassifier

from sklearn.model\_selection import cross\_val\_predict, StratifiedKFold

from sklearn.metrics import classification\_report, confusion\_matrix

import pickle #para guardar modelo entrenado

# Cargar train y test

train = pd.read\_csv("train.csv")

test = pd.read\_csv("test.csv")

X\_train = train[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_train = train["Clase"]

X\_test = test[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_test = test["Clase"]

# Configurar validación cruzada

kfold = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Crear el modelo

modelo\_xgb = XGBClassifier(random\_state=42, use\_label\_encoder=False, eval\_metric='logloss')

# Validación cruzada

y\_pred\_cv = cross\_val\_predict(modelo\_xgb, X\_train, y\_train, cv=kfold)

print("\n=== XGBoost - Validación Cruzada ===")

print(classification\_report(y\_train, y\_pred\_cv, target\_names=["No Somnolencia", "Somnolencia"]))

# Entrenamiento final

modelo\_xgb.fit(X\_train, y\_train)

# Evaluación final en test

y\_pred\_test = modelo\_xgb.predict(X\_test)

print("\n=== XGBoost - Test Final ===")

print(classification\_report(y\_test, y\_pred\_test, target\_names=["No Somnolencia", "Somnolencia"]))

# Matriz de confusión

cm = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", xticklabels=["No Somnolencia", "Somnolencia"], yticklabels=["No Somnolencia", "Somnolencia"])

plt.title("Matriz de Confusión - XGBoost")

plt.xlabel("Predicho")

plt.ylabel("Real")

plt.tight\_layout()

plt.show()

# Guardar el modelo entrenado como archivo .pkl

with open("modelo\_xgboost.pkl", "wb") as f:

pickle.dump(modelo\_xgb, f)

print("Modelo XGBoost guardado como modelo\_xgboost.pkl")

Archivo: Modelo/analisis\_somnolencia.py

import pickle

import numpy as np

from deteccion\_facial import lectura\_datos

# Cargar modelo previamente entrenado

with open("modelo\_xgboost.pkl", "rb") as f:

modelo = pickle.load(f)

# Supón que ya tienes los datos de un frame:

EAR, MAR, Pitch, Yaw, Roll = lectura\_datos()

# Crear input para el modelo

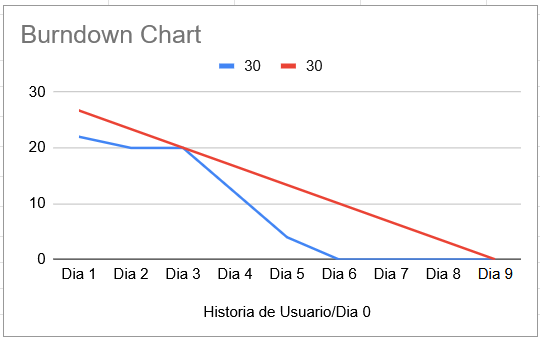
X\_nuevo = np.array([[EAR, MAR, Pitch, Yaw, Roll]])

# Hacer predicción

prediccion = modelo.predict(X\_nuevo)

# 5. Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 23/05 | 24/05 | 25/05 | 26/05 | 27/05 | 28/05 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |
| HU-0.3 | Entrenamiento del Modelo | 10 | 8 | 2 |  |  |  |  |
| HU-0.4 | Integración del modelo al flujo actual | 20 |  |  |  | 8 | 8 | 4 |
| Tiempo de trabajo | | 30 | 22 | 20 | 20 | 12 | 4 | 0 |



# 6. Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Se logró un buen ritmo de trabajo.
* El sprint se terminó en menos días por las actividades que se adelantaron en el sprint anterior.
* Se mantuvieron los días en los que se avanzó el sprint (6 días) ya que está más alineado a la duración de un sprint regular (1 semana).

# 7. Conclusiones

El Sprint se concluyó exitosamente con todas las historias de usuario. Se finalizó la documentación del avance, para luego guardar el modelo elegido y entrenado, y finalmente se integró dicho modelo en el flujo del programa del Sprint 1.

El equipo demostró capacidad de entrega puntual y buena gestión de riesgos menores. Las métricas indican que la carga fue bien distribuida y los objetivos del sprint se cumplieron completamente.

**El sprint duró menos días de lo planeado** debido a que se adelantaron 2 actividades del mismo en un sprint anterior. No se agregaron más actividades para llenar el hueco de días ya que se vió que con 6 días de duración el sprint 0.2 estaba mejor alineado a una duración de sprint regular (1 semana) en comparación con una duración de 9 días (estimación original).