Manual Técnico — Modelo basado en IA para detección de somnolencia

**Fecha**: 20/06/2025

**Autor**: Tirza Buendia González

**Versión**: 1.0

# 1. Introducción

Este manual técnico documenta los aspectos internos del desarrollo del modelo basado en IA para la detección de somnolencia en conductores. El documento está dirigido a desarrolladores, técnicos de soporte y cualquier persona interesada en comprender la estructura, tecnologías y funcionamiento interno del sistema.

# 2. Tecnologías Utilizadas

| Componente | Tecnología o Herramienta |
| --- | --- |
| Interfaz simulada | Python - Tkinter |
| Lógica del sistema | Python (programación estructurada y modular) |
| Modelo de IA | XGBoost (clasificación binaria) |
| Dataset | CSV (con métricas faciales: EAR, MAR, pitch, yaw, roll) |
| Detección facial | MediaPipe (Face Mesh de Google) |
| Preprocesamiento de video | OpenCV |
| Captura en tiempo real | OpenCV + threading (multihilo en Python) |
| Alertas visuales | Tkinter (ventana emergente con mensaje) |
| Alertas auditivas | winsound (pitido de alerta en Windows) |

# 3. Estructura del Proyecto

/Modelo

│

├── captura\_video.py # Módulo para capturar frames desde la cámara en tiempo real

├── deteccion\_facial.py # Procesamiento de landmarks faciales con MediaPipe

├── analisis\_somnolencia.py # Script principal: predicción y activación de alertas

├── gestor\_alertas.py # Interfaz de alerta (ventana + buzzer)

├── modelo\_xgboost.pkl # Modelo entrenado (XGBoost serializado con pickle)

├── registro\_csv.py

├── registro\_alertas.py

├── data/

│ └── dataset\_somnolencia.csv # Dataset con métricas faciales y etiquetas

├── Registros

└──registro.csv

└── incidentes\_frames.png/jpg

# 4. Arquitectura General

Arquitectura en **capas**:

* **Capa de Presentación:** Python - Tkinter (interfaz gráfica emergente para alerta visual)
* **Capa de Lógica de Negocio:** Python - OpenCV, MediaPipe, XGBoost. Se encarga del procesamiento de video, extracción de características, predicción de somnolencia y control de alertas.
* **Capa de Acceso a Datos:** Python - Lectura y escritura de archivos .csv para análisis y almacenamiento de métricas.
* **Capa de Persistencia:** Python - Almacenamiento local del modelo entrenado (.pkl), datasets .csv y capturas (.jpg).

# 5. Descripción de Componentes

## 5.1 Interfaz de Usuario (Frontend)

Lenguaje: Python Librería: Tkinter Estilo visual: Interfaz básica con colores de alerta (fondo rojo, texto blanco). Interacción: Se activa automáticamente al detectar somnolencia y muestra un mensaje visual junto con un sonido (buzzer). Componentes clave:

* Ventana de Alerta: Se muestra solo cuando se detecta somnolencia mantenida.
* Sonido de Buzzer: Se activa en paralelo a la ventana, con un tono intermitente.

## 5.2 Lógica de Negocio (Backend)

Lenguaje: Python Frameworks / Bibliotecas:

* OpenCV: Captura y procesamiento de video.
* MediaPipe: Extracción de métricas faciales (landmarks).
* XGBoost: Modelo de detección de somnolencia entrenado previamente.
* threading, keyboard, time: Control de ejecución y concurrencia.

Flujo principal:

* Captura de video y extracción de métricas (EAR, MAR, Pitch, Yaw, Roll).
* Predicción usando un modelo XGBoost entrenado con dataset externo.
* Activación o cierre de alerta según detección sostenida de somnolencia.

## 5.3 Persistencia y Almacenamiento de Datos

Formato: CSV Ubicación: Local (en la carpeta del proyecto) Datos almacenados:

* Dataset de entrenamiento (frames y métricas).
* Registros de eventos de alerta (timestamp y condición).
* Modelo entrenado (formato .pkl generado con pickle).

# 6. Instalación y Ejecución

## 6.1 Requisitos

* Python 3.9+
* Bibliotecas necesarias: opencv-python, mediapipe, xgboost, tkinter, keyboard
* Sistema operativo: Windows 10+ (uso de winsound)

## 6.2 Instalación

* Clonar o copiar el repositorio del proyecto.
* Instalar dependencias con pip:
* pip install opencv-python mediapipe xgboost keyboard

## 6.3 Ejecución

* Ejecutar el script principal: python analisis\_somnolencia.py
* Presionar 'x' para detener el sistema.

# 7. Seguridad

* Acceso Local Restringido: Los archivos del sistema (CSV, modelo, imágenes) solo pueden ser accedidos por la aplicación. Se recomienda configurar permisos de solo lectura para otros usuarios.
* Validación de Datos: Se inspeccionan los registros CSV para evitar entradas maliciosas o caracteres inválidos.
* Seguridad Física (Cámara): La cámara transmite video en tiempo real, sin almacenar grabaciones completas, previniendo vigilancia indebida.
* Logging Seguro: Los logs generados solo contienen información técnica como timestamps y estados de alerta. No se almacenan datos sensibles ni personales.

# 8. Buenas Prácticas

* Modularización del código: Separación clara entre módulos de captura, análisis, alertas y lógica principal.
* Código legible y comentado: Cada módulo y función cuenta con comentarios que explican su propósito.
* Validación previa: Se verifica la existencia y el formato correcto de los datos antes de procesarlos.
* Uso de hilos (threads): Para mantener la interfaz responsiva y la ejecución paralela entre captura, predicción y alerta.
* Control de errores: Se encapsula el bucle principal en bloques try/except para evitar caídas inesperadas.
* Separación de responsabilidades: La lógica de detección, alerta y captura están implementadas en archivos distintos según su rol.

# 9. Futuras Mejoras

* Generación de reportes con métricas de somnolencia (para ser analizados).
* Uso de cámara infrarroja para mejor detección facial en condiciones de baja luz.
* Sistema embebido para una fácil aplicación en escenarios de conducción.

# 10. Contacto

Para soporte técnico o colaboración: 71417350@continental.edu.pe