

Sistema de Optimización de Asistencia Médica (SOAM)

SOAM optimiza la gestión de citas médicas utilizando Machine Learning.



El Problema: Recursos Médicos Desperdiciados

Situación Actual

Los CESFAM sufren por la **alta inasistencia** a citas, lo que causa el desperdicio de recursos médicos y listas de espera infladas.



1

El Problema

Inasistencia a citas genera recursos desperdiciados.

2

La Solución

Predicción de inasistencias antes de que ocurran.

3

El Objetivo

Gestión proactiva de los usuarios.



Arquitectura del Sistema: Modular y Escalable



Capa de Datos

Generación simulada de datos y comportamiento de pacientes.

Backend API

Microservicio para procesar peticiones y ejecutar predicciones en tiempo real.

Frontend Dashboard

Interfaz para visualizar métricas y apoyar decisiones operativas.

- **Ventaja Clave:** Arquitectura modular permite mantenimiento independiente e integración sencilla con sistemas existentes.

API Backend: Seguro y Eficiente



1

Validación de Datos

Esquemas Pydantic aseguran integridad.

2

Alto Rendimiento

Modelo en RAM para respuestas rápidas.

3

Transformación Automática

JSON a DataFrame sin errores manuales.

Claves

Diseñada para garantizar **seguridad, velocidad y confiabilidad**.

Dashboard Interactivo: Visualización en Tiempo Real

Análisis Exploratorio de Datos (EDA)

El dashboard proporciona una **visión completa del comportamiento del paciente**, facilitando la identificación de patrones y la toma de decisiones informadas.



Simulación en Tiempo Real

Flujo de pacientes simulado con métricas instantáneas.



Patrones Detectados

Altas tasas de abandono en especialidades como Dental y Salud Mental.



Correlaciones Clave

Más tiempo de espera se traduce en mayor inasistencia.

Modelo de Machine Learning: Precisión y Eficiencia

Gradient Boosting Classifier

Utilizamos un **pipeline automatizado** con imputación y codificación para garantizar predicciones reproducibles y consistentes.

Prioridad: Maximización del Recall

Nos enfocamos en minimizar los **falsos negativos** (clase "No Asiste"), conservando horas médicas valiosas y evitando citas perdidas.



1 Pipeline Automatizado

Procesamiento integrado de datos para consistencia.

2 Enfoque en Recall

Minimiza falsos negativos y optimiza recursos.

3 Tolerancia a Fallos

Sistema robusto ante errores de conexión.

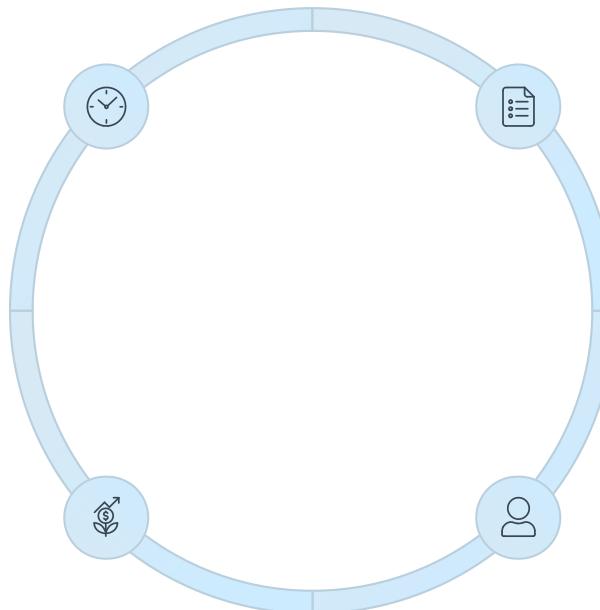
Impacto Esperado y Democratización del Acceso

Reducción de Tiempos Ocosos

Optimización de la capacidad médica

Impacto Sostenible

Mejora del sistema de salud pública



Listas de Espera Optimizadas

Agendamiento inteligente

Herramienta Intuitiva

Decisiones basadas en datos para el personal

"Transformamos datos en decisiones simples, maximizando el impacto del tiempo y recursos médicos."

Video demostrativo



YouTube

Machine Learning – Prediccion de Agendamientos ¿Se puede predecir quién f...

Las inasistencias médicas cuestan millones y aumentan las listas de espera. En este video presentamos nuestro proyecto de Minería de Datos: un Sistema Predictivo de...



```
# --- 1.
data_path = "data/raw/dataset_cesfam_stream.csv"
if not os.path.exists(data_path):
    raise FileNotFoundError(f"No se encontró el dataset en {data_path}. Ejecuta primero data_generator.py")

df = pd.read_csv(data_path)
print(f"✓ Datos cargados: {df.shape[0]} registros.")

# --- 2.
target = 'target_no_asiste'
X = df.drop(columns=[target, 'paciente_id'])
y = df[target]
```

```
# --- 3.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
print(f"◆ Datos de entrenamiento: {X_train.shape[0]}")
print(f"◆ Datos de prueba: {X_test.shape[0]}")

# --- 4.
model = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)

full_pipeline = Pipeline([
    ('preprocessor', get_preprocessing_pipeline()),
    ('classifier', model)
])
```

```
# --- 5.
print("⏳ Entrenando el modelo (esto puede tardar unos segundos)...")
full_pipeline.fit(X_train, y_train)
print("✅ Entrenamiento completado.")

# --- 6.
print("\n--- 📈 Evaluación del Modelo (Set de Prueba) ---")
y_pred = full_pipeline.predict(X_test)
y_proba = full_pipeline.predict_proba(X_test)[:, 1]

print(classification_report(y_test, y_pred))

auc = roc_auc_score(y_test, y_proba)
print(f"📊 ROC-AUC Score: {auc:.4f}")
```

```
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()
print(f"\nMatriz de Confusión:")
print(f"Verdaderos Negativos (Asiste predicho OK): {tn}")
print(f"Falsos Positivos (Error tipo 1): {fp}")
print(f"Falsos Negativos (Error grave - No asiste y no avisamos): {fn}")
print(f"Verdaderos Positivos (No asiste detectado): {tp}")

# --- 7
model_dir = "models"
os.makedirs(model_dir, exist_ok=True)
model_path = os.path.join(model_dir, "model_pipeline.pkl")

joblib.dump(full_pipeline, model_path)
print(f"\nMODEL Modelo guardado exitosamente en: {model_path}")
print("Listo para ser usado por la API.")

if __name__ == "__main__":
    train_model()
```