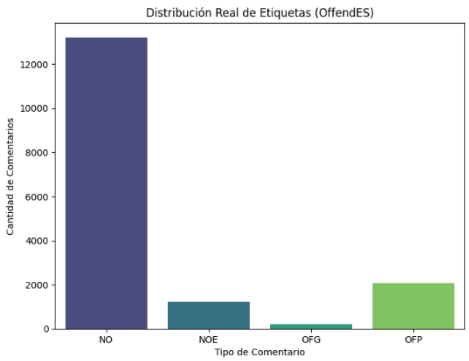
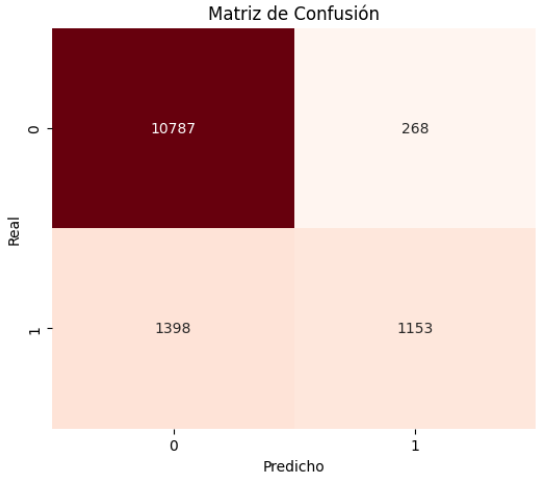
Fecha	Actividad	Solución y Decisiones Técnicas	Evidencias
Lunes 22 de Diciembre de 2025	**Definición del Alcance y Carga de Datos** Originalmente, planteamos un modelo multiclasificación para 4 categorías: 1. Ofensivo (Insulto directo). 2. Vulgar (Groserías sin ofensa). 3. Ambos. 4. Ninguno. **Problema Crítico:** Al descargar el dataset (training_set.tsv) y cargarlo localmente, el DataFrame mostró dimensiones erróneas (4, 6). Parecía un archivo corrupto	**Solución de Carga:** Cambio de estrategia a lectura remota. Utilizamos la URL "Raw" del repositorio de GitHub para forzar la descarga íntegra. **Ajuste de Formato:** Detectamos que el uso de comas en los comentarios rompía el CSV. Se forzó el parámetro sep='\t' (tabulaciones). Dimensiones finales correctas: (16710, 6).	**Error de Carga:** ParserError: Error tokenizing data. C error: Expected 2 fields in line 6, saw 17 **Carga Exitosa:**  `import pandas as pd` `# URLs del repositorio` `url_train = 'https://raw.githubusercontent.com/fmplaza/OffendES/main/split_MeOffendES/training_set.tsv'` `url_test = 'https://raw.githubusercontent.com/fmplaza/OffendES/main/split_MeOffendES/test_set.tsv'` `# Tuvimos un error en una versión anterior por no especificar sep='\t' porque es un archivo separado` `# por tabulaciones y pensamos que era por comas.` `try:` `df_train = pd.read_csv(url_train, sep='\t')` `df_test = pd.read_csv(url_test, sep='\t')` `print(f"Datos cargados. Train: {df_train.shape}, Test: {df_test.shape}")` `except Exception as e:` `print("Error cargando los datos:", e)` `Datos cargados. Train: (16710, 6), Test: (13606, 6)`

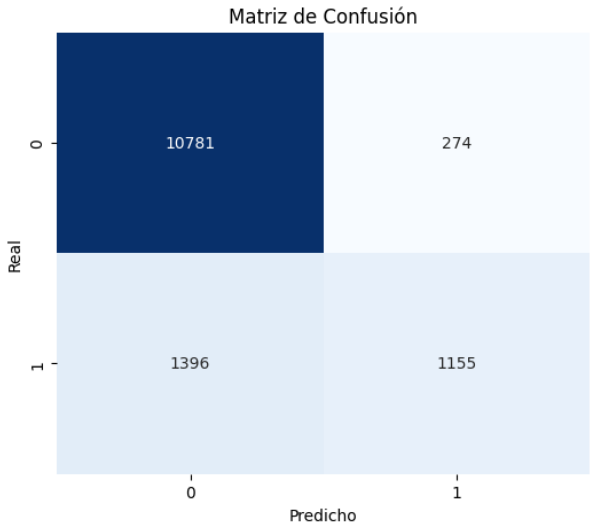
Fecha	Actividad	Solución y Decisiones Técnicas	Evidencias
	o una "vista previa".		
Martes 23 de Diciembre de 2025	**Análisis Exploratorio (EDA) y Re-definición** **Hallazgo:** Al graficar las 4 etiquetas originales, encontramos un desbalance inviable: - NO (No ofensivo): 13,212 muestras. - OFG (Ofensivo Grupal): Solo 212 muestras. **Riesgo:** Con tan pocos datos en OFG, el modelo podría ignorar esta clase para inflar el Accuracy.	**Decisión:** Simplificar el problema de Multiclase a **Clasificación Binaria**. **Agrupación:** - NO y NOE **=Clase 0 (No Ofensivo)**. - OFP y OFG **=Clase 1 (Ofensivo)**. Esto nos permite consolidar la minoría y mejorar el aprendizaje.	**Gráfica de Desbalance:**  **Diccionario de Mapeo:** ``` # Mapeo de etiquetas y_train = df_train['label'].map({'NO': 0, 'NOE': 0, 'OFP': 1, 'OFG': 1}) y_test = df_test['label'].map({'NO': 0, 'NOE': 0, 'OFP': 1, 'OFG': 1}) ```

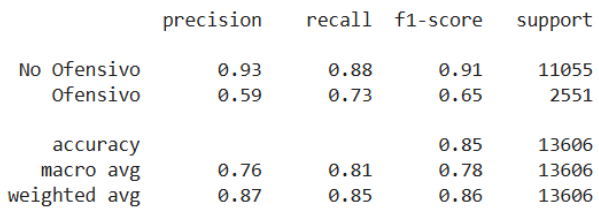
Fecha	Actividad	Solución y Decisiones Técnicas	Evidencias
	Preprocesamiento **Problema:** Necesitábamos limpiar el texto sucio de redes sociales. **Acción:** Aplicamos una limpieza estándar con Regex, eliminando todo carácter que no fuera letra o número. Eliminamos los emojis.	**Implementación:** Se creó la función limpieza_basica y se vectorizó con TF-IDF. *Nota: En este punto, creímos que eliminar caracteres especiales era la mejor práctica para reducir ruido.*	 ``` import re # Definimos la función de limpieza def limpieza_basica(text): text = str(text).lower() # Mantenemos solo letras y espacios. text = re.sub(r'[^a-záéíóúñ\s]', '', text) text = re.sub(r'\s+', ' ', text).strip() return text # Aplicamos la limpieza df_train['comment_clean'] = df_train['comment'].apply(limpieza_basica) df_test['comment_clean'] = df_test['comment'].apply(limpieza_basica) ```
Sábado 27 de diciembre de 2025	**Entrenamiento Modelo Base (SVM v1)** **Configuración:** Utilizamos	Por los resultados, vemos que el modelo está sesgado hacia la clase mayoritaria (No Ofensivo)	**Reporte de Clasificación:** Resultados precision recall f1-score support No Ofensivo 0.89 0.98 0.93 11055 Ofensivo 0.81 0.45 0.58 2551 accuracy 0.88 13606 macro avg 0.85 0.71 0.75 13606 weighted avg 0.87 0.88 0.86 13606

Fecha	Actividad	Solución y Decisiones Técnicas	Evidencias
	LinearSVC por defecto. **Resultado:** Accuracy engañoso del 88%. **Falla:** Al revisar la matriz de confusión, notamos que el modelo casi nunca predecía la clase 'Ofensivo' (Recall ~0.45).	debido al desbalance que detectamos en la sesión anterior, además, nos estamos cuestionando si mantener los emojis porque creemos que la limpieza les quitó contexto a los insultos. Necesitamos arreglar la limpieza y el balanceo.	**Matriz de Confusión Roja:** 

Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
Lunes 29 de Diciembre de 2025	**Análisis de Errores** Al revisar manualmente los "Falsos Negativos", notamos que muchos insultos solo podían ser considerados	**Implementación de Librería emoji:** Decidimos no borrar los emojis, sino traducirlos a texto para que el modelo pudiera interpretarlos semánticamente.	``` # Nueva función de limpieza # Ahora usamos emoji.demojize antes de borrar caracteres. def clean_text(text): text = str(text) #Traducimos emojis a texto text = emoji.demojize(text, language='es', delimiters=(" ", " ")) # Pasamos a minúsculas text = text.lower() # Limpieza de caracteres (permitimos guiones bajos _ por los emojis) text = re.sub(r'[^a-záéíóúñ0-9\s_]', '', text) text = re.sub(r'\s+', ' ', text).strip() return text # Aplicamos la limpieza df_train['comment_clean'] = df_train['comment'].apply(clean_text) df_test['comment_clean'] = df_test['comment'].apply(clean_text) ```

Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
	insultos con los emojis. El bajo rendimiento no es culpa del algoritmo, sino de que estamos borrando la información más valiosa (los emojis) con nuestra limpieza.	**Función clean_text:** Usamos emoji.demojize(text, language='es'). Esto debería dar al modelo las palabras clave que le faltaban.	
	Entrenamiento y Persistencia del Error Entrenamos el mismo modelo (LinearSVC) con los nuevos datos enriquecidos con emojis. **Resultado:**	El problema raíz es el Desbalance de Clases (detectado en la Sesión 2), no solo la limpieza. Necesitamos forzar matemáticamente al modelo a prestar atención a la clase minoritaria.	Resultados: precision recall f1-score support No Ofensivo 0.89 0.98 0.93 11055 Ofensivo 0.81 0.45 0.58 2551 accuracy 0.88 13606 macro avg 0.85 0.71 0.75 13606 weighted avg 0.87 0.88 0.86 13606

Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
	Aunque el *Accuracy* se mantuvo alto (~88%), la métrica clave Recall (Sensibilidad) de la clase 'Ofensivo' apenas mejoró, pues siguió estancada alrededor de 0.45. Mejorar los datos no fue suficiente. El modelo optimiza sus aciertos prediciendo siempre la clase mayoritaria ('No Ofensivo') porque no tiene penalización por fallar en la minoritaria.	Investigamos y encontramos el hiperparámetro class_weight='balanced' para SVM y Regresión Logística.	

Sesión / Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
Viernes 2 de enero de 2026	**Corrección del Desbalance** **Acción:** Implementamos el parámetro class_weight='balanced' en el SVM y probamos también con Regresión Logística para comparar. **Resultado:** El modelo dejó de ignorar los insultos. La sensibilidad (Recall) de la clase ofensiva saltó de ~0.48 a **0.68** en la primera prueba. **Observación:** Al fin el modelo está castigando sus errores en la clase minoritaria.	**Validación Cruzada:** Confirmamos que la mejora no fue suerte usando validación simple. La métrica F1-Score subió considerablemente. **Decisión:** Nos quedamos con SVM (LinearSVC) como modelo principal, pero necesitamos afinar la regularización para evitar *overfitting*.	 precision recall f1-score support No Ofensivo 0.93 0.88 0.91 11055 Ofensivo 0.59 0.73 0.65 2551 accuracy 0.85 13606 macro avg 0.76 0.81 0.78 13606 weighted avg 0.87 0.85 0.86 13606

Sesión / Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
	Afinación de Hiperparámetros (GridSearch) **Problema:** ¿Cuál es el mejor valor de regularización (C)? **Acción:** Ejecutamos una búsqueda exhaustiva (GridSearchCV) probando valores de C: [0.01, 0.1, 1, 10, 100]. **Resultado:** La mejor configuración fue C=0.1. Valores más altos de C causaban ruido y valores más bajos no aprendían bien.	**Optimización Final:** Se re-entrenó el modelo con C=0.1 y class_weight='balanced'. **Métricas Finales en Test:** - Accuracy: 86% - Recall (Ofensivo): 0.73 - Precision (Ofensivo): 0.60 Consideramos esto un equilibrio aceptable para un sistema de detección de odio.	Mejor configuración encontrada: {'C': 0.1} precision recall f1-score support No Ofensivo 0.93 0.89 0.91 11055 Ofensivo 0.60 0.73 0.66 2551 accuracy 0.86 13606 macro avg 0.77 0.81 0.78 13606 weighted avg 0.87 0.86 0.86 13606

Sesión / Fecha	Actividad y Problema Detectado	Solución y Decisiones Técnicas	Evidencia
Sábado 4 de enero de 2026	**Pruebas Manuales y Documentación** Probamos el modelo con frases propias (slang mexicano y emojis) para verificar robustez.	Observamos que el modelo demostró ser capaz de entender jerga de internet y contextos con emojis que en la fallaban. Compilación del Notebook final y exportación de requerimientos (requirements.txt).	Pruebas del modelo Frase: 'Me das cringe' Limpieza: 'me das cringe' Resultado: OFENSIVO Frase: 'Eres un inútil, no sirves para nada' Limpieza: 'eres un inútil no sirves para nada' Resultado: OFENSIVO Frase: 'Ojalá te mueras pronto maldito' Limpieza: 'ojalá te mueras pronto maldito' Resultado: OFENSIVO Frase: 'No eres tonta' Limpieza: 'no eres tonta' Resultado: OFENSIVO Frase: 'Me gustan tus videos' Limpieza: 'me gustan tus videos' Resultado: No Ofensivo