

Presentación Ejecutiva: Turkish Music Emotion

MLOps
Fase 1 | Avance de
Proyecto

Integrantes Equipo 24

A01796937 - Sandra Luz Cervantes Espinoza
A01795838 - Javier Augusto Rebull Saucedo
A01360416 - David Cruz Beltrán



INTRODUCCIÓN

¿Qué problema resolvemos?

Clasificar automáticamente la emoción que transmite una pieza de música turca en **4 clases principales:**

Happy

Sad

Angry

Relax

Para habilitar casos de uso como curación de playlists, recomendaciones y apoyo en terapias/experiencias sonoras.

Trabajamos con el dataset modificado **Turkish Music Emotion** como punto de partida.

~400

Pistas totales

100

Por clase

4

Emociones

RECONOCIMIENTO DE EMOCIONES EN MÚSICA TURCA CON MLOPS

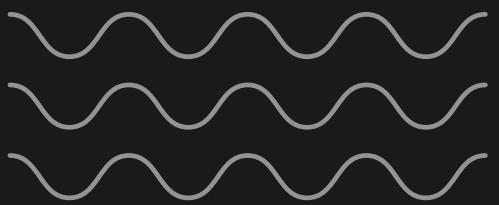
¿Por qué es difícil?

Subjetividad y variabilidad: la emoción musical es compleja; la señal de audio varía en timbre, tempo y dinámica incluso dentro de una misma emoción.

Señal ruidosa y heterogénea: requiere preprocessamiento (limpieza, normalización de sampling rate) y extracción de características (vector de 34 features por pista) antes de modelar.

Datos limitados: un conjunto relativamente pequeño demanda rigor para evitar sobreajuste y asegurar reproducibilidad y trazabilidad de experimentos.





PROPUESTA DE VALOR

CLASIFICACIÓN DE EMOCIONES MUSICALES



Precisión

- Ofrecer un modelo confiable que pueda identificar emociones en música turca (happy, sad, angry, relax).
- Útil para aplicaciones de recomendación musical, terapia musical, o personalización de playlists según el estado emocional del usuario.



Valor a usuarios

- Un usuario que busca **relajarse** después de un día estresante puede recibir automáticamente recomendaciones de canciones clasificadas como relax.
- **Plataformas de música** podrían **personalizar** playlists según el estado de ánimo del usuario, mejorando la experiencia personalizada y aumentando la satisfacción.
- **Terapia musical**, donde seleccionar música según la emoción deseada puede potenciar los efectos terapéuticos.



ANÁLISIS DEL PROBLEMA

Objetivo

Construir un modelo capaz de aprender patrones en los datos que reflejen cómo ciertos atributos del sonido (**como ritmo, energía, tonalidad y timbre**) se asocian con emociones humanas clasificadas en feliz, triste, enojado o relajado.

Clasificar la emoción

Clasificar la emoción predominante en fragmentos de música turca a partir de sus características acústicas.

Clasificación supervisada multiclasa

Cada muestra está etiquetada con una de cuatro emociones posibles: happy, sad, angry o relax.

Resultado esperado (valor)

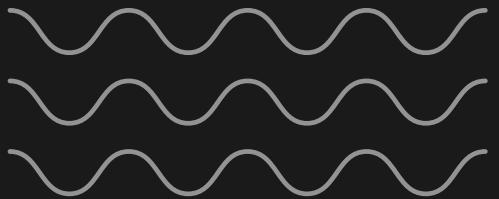
Un **sistema repetible y auditble** que transforme audio en predicciones de emoción con **métricas monitoreadas**, facilitando su escalamiento y despliegue controlado.

¿CÓMO LO ABORDA MLOPS?

- **Gobernanza de datos con DVC + S3:** versionado de datasets (raw → processed), historial de cambios y sincronización en la nube.
- **Pipelines reproducibles:** estructura Cookiecutter Data Science con ingestión → limpieza/EDA → feature engineering → training → evaluación → empaquetado.
- **Trazabilidad de experimentos con MLflow:** métricas, parámetros, artefactos y registro de modelos para comparar y promover a producción.
- **Buenas prácticas de modelado:** establecer baselines (Logistic/SVM/KNN/MLP, árboles/ensembles), validar con Accuracy/F1 macro y matrices de confusión, y automatizar hyperparameter tuning.
- **Benchmark de referencia:** en literatura con música turca (4 clases), enfoques de SVM/KNN/ANN logran $\approx 79\%$ de accuracy; tomamos ese valor como piso para evaluar mejoras.



ROLES MIOPS ASIGNADOS



DAVID CRUZ

A01360416

SOFTWARE ENGINEER



JAVIER REBULL

A01795838

**DATA ENGINEER / SITE
RELIABILITY ENGINEER
(DEVOPS)**

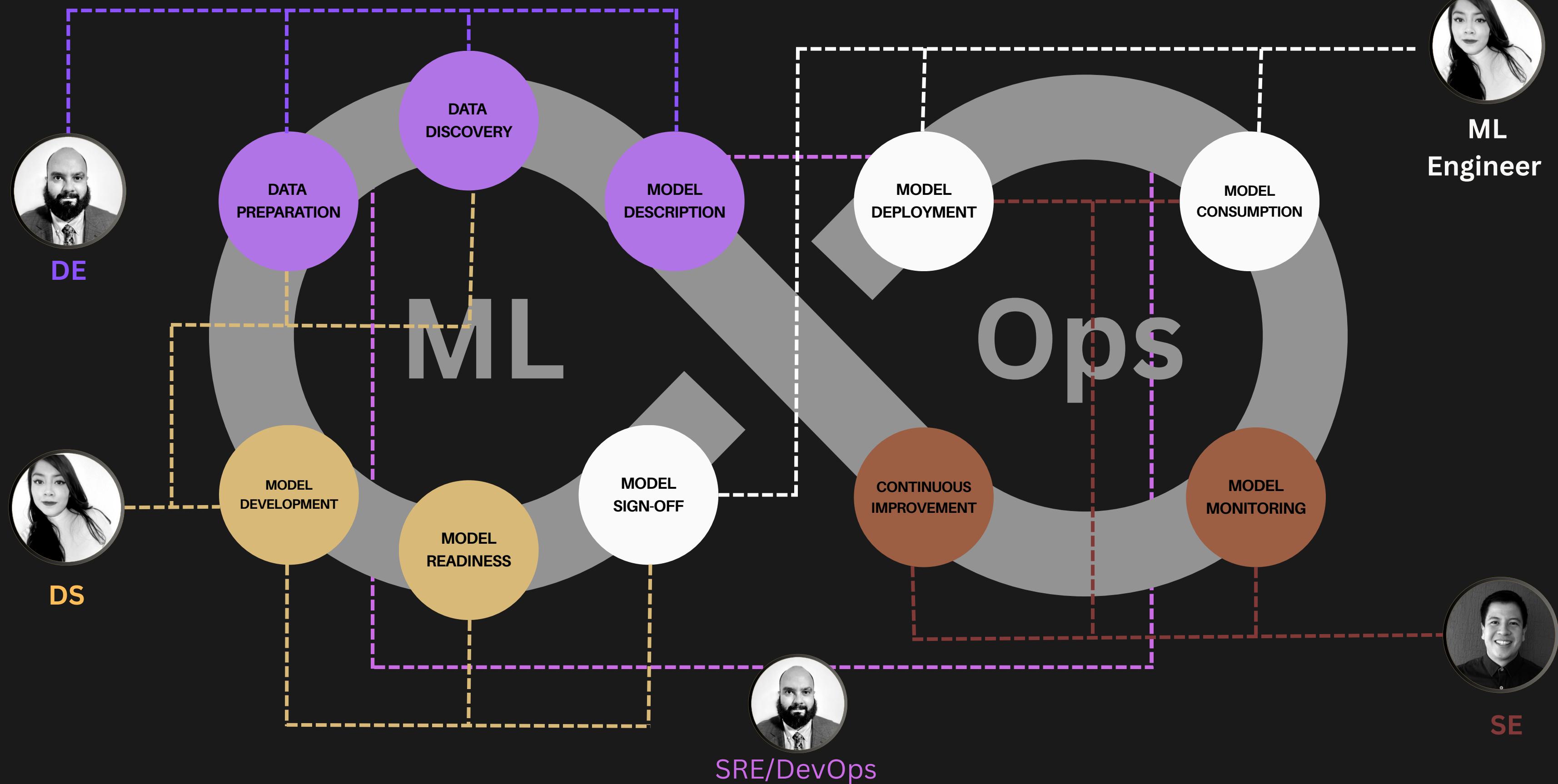


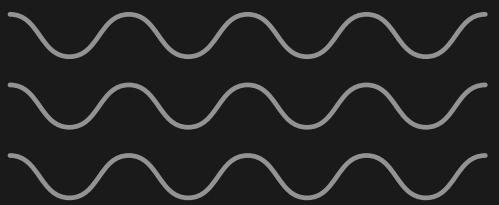
**SANDRA
CERVANTES**

A01796937

**DATA SCIENTIST / ML
ENGINEER**

ROLES EN EL CICLO DE VIDA





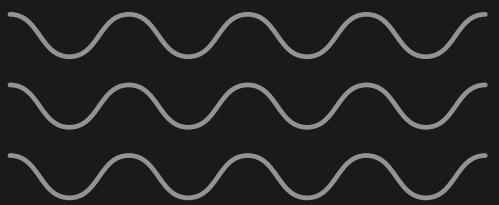
ANÁLISIS DE REQUERIMIENTOS

FASE DEL PROYECTO	Data Engineer	Data Scientist	ML Engineer	Software Engineer	SRE DevOps Eng
RECOLECCIÓN DE DATOS	Identifica y conecta fuentes (audios turcos, metadatos, etiquetas). Define contratos/catálogos y políticas de acceso. Automatiza la ingesta almacenamiento versionado (DVC+S3).	Define taxonomía de emociones (Happy/Sad/Angry/Relax). Criterios de etiquetado y muestreo estratificado. Guía de curaduría y validación.	Especifica requisitos de formato para entrenamiento/features. Prototipo rápido de extracción de features para validar viabilidad.	n/a	Provisiona almacenamiento seguro e IAM. Políticas de retención/backup. Jobs iniciales de ingestión/validación.
LIMPIEZA Y TRANSFORMACIÓN (ETL)	Normaliza sample rate/canales. Remueve silencios/ruido. ETL reproducible con DVC. Validaciones de calidad y catálogo actualizado.	Criterios de exclusión/outliers. Definición de features a extraer (MFCC, tempo, cromas) Particiones estratificadas.	Empaquetá pipeline de preprocessamiento y empaquetá features. Caching y reutilización de artefactos. Alinea procesos de training y serving.	n/a	Configura infraestructura de despliegue (K8s / ECS). Implementa estrategias Blue/Green o Canary. Habilita autoscaling, health checks y rollback.
ANÁLISIS EXPLORATORIO Y DISEÑO DEL MODELO	Proporciona vistas y consultas, así como lineage de datos. Brinda soporte para muestreos y perfiles de datos.	EDA de señales y features (distribuciones, correlaciones, balance). Hipótesis y selección de métricas (Accuracy, F1 macro). Plan de validación (k-fold/estratificado).	Setup de tracking (MLflow). Plantilla de experimentos con configuración. Plan de búsqueda de hiperparámetros.	n/a	Pipelines de CI para lint/tests/packaging. Despliegue seguro de MLflow/artefact store.
ENTRENAMIENTO Y VALIDACIÓN DEL MODELO	Entrega datasets versionados (train/val/test). Definición de slices de evaluación por emoción/tempo/duración.	Entrena baselines (SVM/KNN/MLP/árboles). Matriz de confusión y análisis de error. Selección del candidato.	Automatiza entrenamiento y tuning. Registro de modelos/metrics en MLflow. Artefactos listos para servir.	n/a	Ejecuta jobs de entrenamiento en compute gestionado. Administra secretos de forma segura. Controla costos y mantiene trazas de ejecución.
DESPLIEGUE DEL MODELO (DEPLOYMENT)	n/a	Model card, criterios de aceptación y riesgos conocidos. Checklist de sesgos/limitaciones.	Servicio de predicción (FastAPI). Contenerización (Docker). Batch scoring/streaming y logging estructurado. Flags para A/B o canary.	API/SDK de consumo, autenticación y rate limiting. Pruebas de integración y ejemplos (OpenAPI).	Configura infraestructura de despliegue (K8s / ECS). Implementa estrategias Blue/Green o Canary. Habilita autoscaling, health checks y rollback.
MONITOREO EN PRODUCCIÓN	Monitoreo de calidad de datos y schema/feature drift. Alertas por anomalías en ingestión.	Seguimiento de rendimiento con ground truth diferido. Alarmas por caída de F1/accuracy y análisis post-mortem.	Monitoreo de latencia/throughput. Detección de skew train-serve. Trazas y explicabilidad básica.	Logs de aplicación, manejo de errores y feedback del usuario. Reintentos y degradación controlada.	SLIs/SLOs y dashboards (Prometheus/Grafana). Alerting, on-call y runbooks.
REENTRENAMIENTO Y MANTENIMIENTO	Ingesta de nuevos datos/etiquetas. Backfill y versionado. Actualización del catálogo y linaje.	Criterios de disparo de retraining. Comparativas entre versiones. Documentación de cambios y riesgos.	Orquestación de retraining programado. Promoción/rollback de modelos. Migraciones y compatibilidad de artefactos.	Compatibilidad de API/SDK entre versiones. Pruebas regresivas y de compatibilidad.	Programación de jobs, IaC actualizada. Backups/DR y cumplimiento. Gestión de costos.

ETAPA	Métodos Utilizados	Técnicas Aplicadas	Actividad
REQUERIMIENTOS	Análisis del problema. Casos de uso. Criterios de éxito	MLCanvas ML Canvas. CRISP-ML(Q). Historias de usuario. Definición de métricas	Definir objetivo: predecir emoción (Happy/Sad/Angry/Relax) en música turca; alcance, riesgos/sesgos, métrica objetivo (F1 macro) y baseline de referencia
EDA Y LIMPIEZA	Análisis exploratorio. Control de calidad.	Perfilado de datos, estadísticas descriptivas y visualización; balance de clases; normalización dB y remoción de silencios/ruido	Mejorar calidad del dataset y validar supuestos; particiones estratificadas (train/val/test)
PREPROCESAMIENTO	Transformación de datos. Feature engineering de audio	Extracción de features (MFCCs, cromas, contraste espectral, centroid/rolloff, ZCR, RMS, tempo) y estandarización	Generar matriz de características lista para modelar (pipeline reproducible)
VERSIONADO	Control de versiones. Gobernanza y trazabilidad	DVC para datos (raw→processed) y MLflow para experimentos/modelos	Reproducibilidad de datasets, parámetros y resultados; registro de modelos y artefactos
MODELADO	Entrenamiento supervisado multiclas.	Baselines (SVM, KNN, LogReg/Softmax, MLP, árboles/ensembles), validación k-fold estratificada, tuning de hiperparámetros	Seleccionar candidato con mejor F1 macro; matriz de confusión y análisis de error
EVALUACIÓN	Métricas según tipo de problema. Evaluación offline y prueba holdout.	Métricas: Accuracy y F1 macro; análisis por clase; ablation de features; model card	Demostrar rendimiento y criterios go/no-go para despliegue

PREDICTION TASK	DECISIONS	VALUE PROPOSITION	DATA COLLECTION	DATA SOURCES
<p>PREDICTION TASK</p>  <p>Clasificación supervisada multiclase (single-label) para inferir la emoción predominante en un clip de música turca. La unidad de predicción es cada fragmento de audio ya preprocessado (normalización, duración fija).</p> <p>Resultados y observación.</p> <p>El modelo predice una de 4 clases: Feliz, Triste, Enojado, Relajado (salida: probabilidades y clase por argmax). Las etiquetas existen en el dataset; en producción se confirman a posteriori vía anotación/feedback de usuario para cerrar el ciclo de reentrenamiento.</p> <p>Parámetros de decisión (app/pipeline):</p> <p>Regla principal: argmax si confianza ≥ 0.75; si no, Top-2 o "Revisión". Agregación por pista con ventana configurable (p. ej., 30–60 s). Objetivo de latencia < 2 s/clip en línea; procesamiento batch para bibliotecas grandes. Interpretabilidad con SHAP/LIME visible en el dashboard. A/B testing para umbrales y presentación; feedback del usuario registrado como ground truth para reentrenamiento (programado o por drift).</p>	<p>DECISIONS</p>  <p>Cada clip se etiqueta con una emoción (Feliz/Triste/Enojado/Relajado) y una confianza. Esta etiqueta alimenta: (1) un recomendador por estado de ánimo (playlists relajantes/energizantes), (2) enriquecimiento del catálogo con metadatos emocionales para búsqueda/ordenación, y (3) dashboards afectivos para análisis (por artista, álbum o sesión de musicoterapia). Para decisiones a nivel pista, agregamos múltiples clips por voto mayoritario o promedio de probabilidades.</p> <p>Parámetros de decisión (app/pipeline):</p> <p>Regla principal: argmax si confianza ≥ 0.75; si no, Top-2 o "Revisión". Agregación por pista con ventana configurable (p. ej., 30–60 s). Objetivo de latencia < 2 s/clip en línea; procesamiento batch para bibliotecas grandes. Interpretabilidad con SHAP/LIME visible en el dashboard. A/B testing para umbrales y presentación; feedback del usuario registrado como ground truth para reentrenamiento (programado o por drift).</p>	<p>VALUE PROPOSITION</p>  <p>Beneficiarios: plataformas musicales (curaduría/recomendación), musicólogos/investigadores y clínicas de musicoterapia.</p> <p>Dolores: etiquetado emocional manual e inconsistente; falta de modelos para música turca/no occidental; baja trazabilidad y poca escalabilidad del análisis.</p> <p>Integración y flujo: API REST/Batch que enriquece catálogos con emoción + confianza y se conecta al recomendador/ETLs; dashboard con distribuciones por emoción y explicaciones (SHAP/LIME). Umbrales configurables, alertas por drift y feedback para retraining.</p>	<p>DATA COLLECTION</p>  <p>Data Collection – Fuente inicial</p> <p>Dataset modificado/limpio Turkish Music Emotion: 400 clips de 30 s (≈ 100/clase: Feliz, Triste, Enojado, Relajado). Extraemos MFCC, cromas, tempo, ZCR, RMS a CSV/Parquet; audios/features versionados con DVC (raw→processed).</p> <p>Actualización</p> <p>(frescura/costo)</p> <p>Ingesta periódica desde Spotify/YouTube APIs con deduplicación; extracción incremental y caché solo para nuevos/cambiados; batch semanal. Costos: storage tiering y retención. Etiquetado humano para baja confianza; retraining por tiempo o drift.</p>	<p>DATA SOURCES</p>  <p>Internas (principal)</p> <p>Repository DVC/S3: Audio WAV/MP3 (carpetas raw/processed) y features tabulares (CSV/Parquet).</p> <p>Tablas:</p> <ul style="list-style-type: none"> tracks (id_track, artista, título, fuente, género, año) clips (id_clip, id_track, start_s, dur_s, sr, bitrate) labels (id_clip, emoción{Feliz, Triste, Enojado, Relajado}, fuente_etiqueta) features (id_clip, mfcc_, chroma_, rolloff, zcr, rms, tempo) feedback (id_clip/pista, predicción, confianza, "correcto/incorrecto", timestamp) <p>Externas (potenciales)</p> <ul style="list-style-type: none"> Spotify Web API: search, tracks/{id}, audio-features, audio-analysis (metadatos, tempo, key, duración). YouTube Data API v3: search.list, videos.list (metadatos e IDs; ingesta de audio sujeta a licencias). Repos públicos etiquetados (p. ej., corpus de música turca con emociones) para ampliación y benchmark. <p>Conectores/Extracción: Librosa (MFCC, cromas, espectrales, tempo) y Pandas para consolidar a tablas versionadas con DVC.</p>

IMPACT SIMULATION	MAKING PREDICTIONS		BUILDING MODELS	FEATURES
<p>Costo/Ganancia. Predicción correcta → mayor uso del recomendador y confianza; Incorrecta → degradación de UX y credibilidad ($\approx 10\text{--}15\%$ de caída por cada $+10\%$ de error).</p> <p>Datos para simular. Train/val/test (80/20), k-fold estratificado, piloto en catálogo (batch replay) y muestra con etiquetado humano.</p> <p>Criterios de despliegue. $F1_{macro} \geq 0.78$, Accuracy $\geq 80\%$, latencia $< 2\text{ s}/clip$, confianza calibrada, artefactos en MLflow y datos DVC.</p> <p>Equidad. Métricas por clase y género/tempo; $\Delta F1$ por clase ≤ 10 pts; revisión manual para baja confianza y monitoreo de drift tras el despliegue.</p>	<p>Modos: Batch (offline) y tiempo real (API).</p> <p>Frecuencia: batch al llegar nuevos lotes o en ventana nocturna; online on-demand por clip/pista.</p> <p>SLA/tiempo: $< 2\text{ s}/clip$ end-to-end (prepro + features + inferencia + decisión); caché de features y micro-batching cuando aplique.</p> <p>Recursos: CPU optimizada (multihilo) para batch; GPU opcional si el modelo lo requiere; servicio Docker + FastAPI, orquestado (p. ej., K8s/ECS) con autoscaling.</p> <p>Integración: API REST/job batch que retorna probabilidades + clase y registra confianza/telemetría para retraining.</p>		<p>En producción: 1 clasificador multiclase (emociones); modelo sombra para A/B en actualizaciones; calibración opcional.</p> <p>Actualización: Trimestral o por drift/$\geq 10\text{--}15\%$ de nuevos datos etiquetados; promoción controlada en MLflow Registry (datos en DVC).</p> <p>Tiempo (E2E): ventana nocturna ($\leq 8\text{ h}$) para featureado incremental + entrenamiento + análisis (k-fold, $F1_{macro}$, matriz de confusión).</p> <p>Recursos: CPU optimizada para Sklearn/featureado; GPU (T4) opcional si se prueba un modelo profundo.</p>	<p>Representación (predicción): vector por clip con MFCC (1–13 + $\Delta/\Delta\Delta$), Chroma, Spectral Centroid/Rolloff/Contrast, ZCR, RMS/Energy y Tempo; se agregan por ventanas a estadísticos (media, std, p10/p90) para un único vector por clip.</p> <p>Transformaciones: audio → mono 22.05 kHz y loudness norm; framing ($\approx 1\text{ s} / 0.5\text{ s}$ hop); z-score (parámetros del train), imputación si aplica; PCA opcional (20–40 comps). Paridad train/serve en el mismo pipeline (librosa/sklearn), caché y versionado DVC (CSV/Parquet).</p>
<p>MONITORING</p> <ul style="list-style-type: none"> Métricas técnicas (SLIs/SLOs): $F1_{macro}$ y $F1$ por clase, Accuracy, latencia $p95 < 2\text{ s}/clip$, tasa de error, calibración (ECE), drift de datos/modelo (PSI/KL, cambio en distribución de clases), cobertura y % de predicciones de baja confianza. Métricas de negocio: uso del recomendador (CTR, skips, tiempo en sesión), satisfacción (CSAT/NPS), tasa de feedback/etiquetado y acuerdos entre anotadores. <p>Revisión & acciones:</p> <ul style="list-style-type: none"> Alertas en tiempo real (latencia, caída de $F1$, drift): revisión semanal operativa y mensual de negocio en dashboards (Prometheus/Grafana + MLflow). Triggers: retraining/umbral cuando $\Delta F1_{macro} > 5\text{--}10$ pts o drift significativo; rollback/canary si empeora; seguimiento de fairness ($\Delta F1$ por clase ≤ 10 pts). 				



MANIPULACIÓN Y PREPARACIÓN DE DATOS

ENTENDIENDO EL DATASET

CONTEXTO DEL PROBLEMA

El dataset **turkish_music_emotion_modified.csv** contiene características acústicas extraídas de canciones turcas para clasificar emociones musicales. El reto principal es predecir la emoción que evoca una canción basándose únicamente en sus propiedades sonoras, sin analizar la letra.

Source: UC Irvine Machine Learning Repository

MÉTRICAS DE ÉXITO ESPERADAS

- ✓ **Accuracy:** >75% (4 clases balanceadas = 25% baseline)
- ✓ **F1-Score macro:** >0.70 (considerar todas las emociones equitativamente)
- ✓ **Matriz de confusión:** Minimizar confusión entre emociones similares (ej: Triste vs Relajado)

CATEGORÍAS DE VARIABLES

Categoría	Variables	Función	Descripción
 Características Espectrales	spectral_centroid_mean, spectral_bandwidth_mean, spectral_rolloff_mean	Identificar el timbre y brillo del sonido	Capturan la distribución de frecuencias. Un centroide alto = sonido brillante
 Características Temporales	zero_crossing_rate_mean, tempo	Analizar ritmo y dinamismo	Miden cambios en la señal. Alto ZCR = sonido percusivo/ruidoso
 MFCC	mfcc1_mean hasta mfcc20_mean (20 variables)	Representar la forma espectral única	Similar a la "huella dactilar" acústica. Fundamental para reconocimiento de patrones
 Características de Energía	rms_mean (Root Mean Square)	Medir la intensidad sonora	Indica qué tan "fuerte" o "suave" es la canción
 Características Armónicas	chroma_stft_mean	Capturar contenido armónico/tonal	Representa las 12 notas de la escala musical
 Variable Objetivo	Class	Etiqueta de emoción	Categorías: Triste, Enérgico, Relajado, Feliz

EL DATASET Y SUS RETOS PRINCIPALES

Alta Dimensionalidad

- 27+ variables acústicas (especialmente 20 MFCCs)
- Riesgo de overfitting y curse of dimensionality
- Solución: PCA, selección de features, regularización

Desbalanceo de Clases

- Algunas emociones pueden estar sobre-representadas
- Afecta precisión del modelo en clases minoritarias
- Solución: SMOTE, class weights, stratified sampling

Especificidad Cultural

- Música turca tiene características únicas (makams, ritmos aksak)
- Modelos de música occidental podrían no transferirse bien
- Solución: Feature engineering específico para música turca

Subjetividad Emocional

- La emoción musical es altamente subjetiva
- Una canción puede evocar emociones diferentes según el oyente
- Solución: Enfoque en patrones estadísticos consistentes

Correlación entre Features

- MFCCs y características espectrales correlacionadas
- Puede causar multicolinealidad
- Solución: Análisis de correlación, eliminación de redundancias

Variable	Nombre legible	Tipo (grupo)
Class	Emoción (label)	Etiqueta
_RMSenergy_Mean	Energía RMS (media)	Energía/Dinámica
_Lowenergy_Mean	Proporción de baja energía	Energía/Dinámica
_Fluctuation_Mean	Fluctuación rítmica de energía	Energía/Dinámica
_Zero-crossingrate_Mean	Tasa de cruces por cero (media)	Energía/Dinámica
_AttackTime_Mean	Tiempo de ataque (medio)	Energía/Dinámica
_AttackTime_Slope	Tendencia del tiempo de ataque	Energía/Dinámica
_Brightness_Mean	Brillo (energía en agudos)	Energía/Dinámica
_Tempo_Mean	Tempo (medio)	Tempo/Ritmo
_Eventdensity_Mean	Densidad de eventos	Tempo/Ritmo
_Pulseclarity_Mean	Claridad de pulso	Tempo/Ritmo
_MFCC_Mean_1	MFCC 1 (media)	Timbre (MFCC)
Rolloff_Mean	Rolloff espectral (medio)	Espectro
Spectralcentroid_Mean	Centroide espectral (medio)	Espectro
Spectralspread_Mean	Dispersión espectral (media)	Espectro
Spectralskewness_Mean	Asimetría espectral (media)	Espectro
Spectralkurtosis_Mean	Kurtosis espectral (media)	Espectro
Spectralflatness_Mean	Planitud espectral (media)	Espectro
EntropyofSpectrum_Mean	Entropía del espectro (media)	Espectro
Roughness_Mean	Rugosidad/disonancia (media)	Espectro
Roughness_Slope	Tendencia de rugosidad	Espectro
Chromagram_Mean_1	Croma 1 (C) (media)	Tonalidad (Chroma)
HarmonicChangeDetectionF	HCDF (media)	Cambios armónicos (HCDF)
HarmonicChangeDetectionF	HCDF (desviación)	Cambios armónicos (HCDF)
HarmonicChangeDetectionF	HCDF (tendencia)	Cambios armónicos (HCDF)
HarmonicChangeDetectionF	HCDF (frecuencia periódica)	Cambios armónicos (HCDF)
HarmonicChangeDetectionF	HCDF (amplitud periódica)	Cambios armónicos (HCDF)
HarmonicChangeDetectionF	HCDF (entropía periódica)	Cambios armónicos (HCDF)

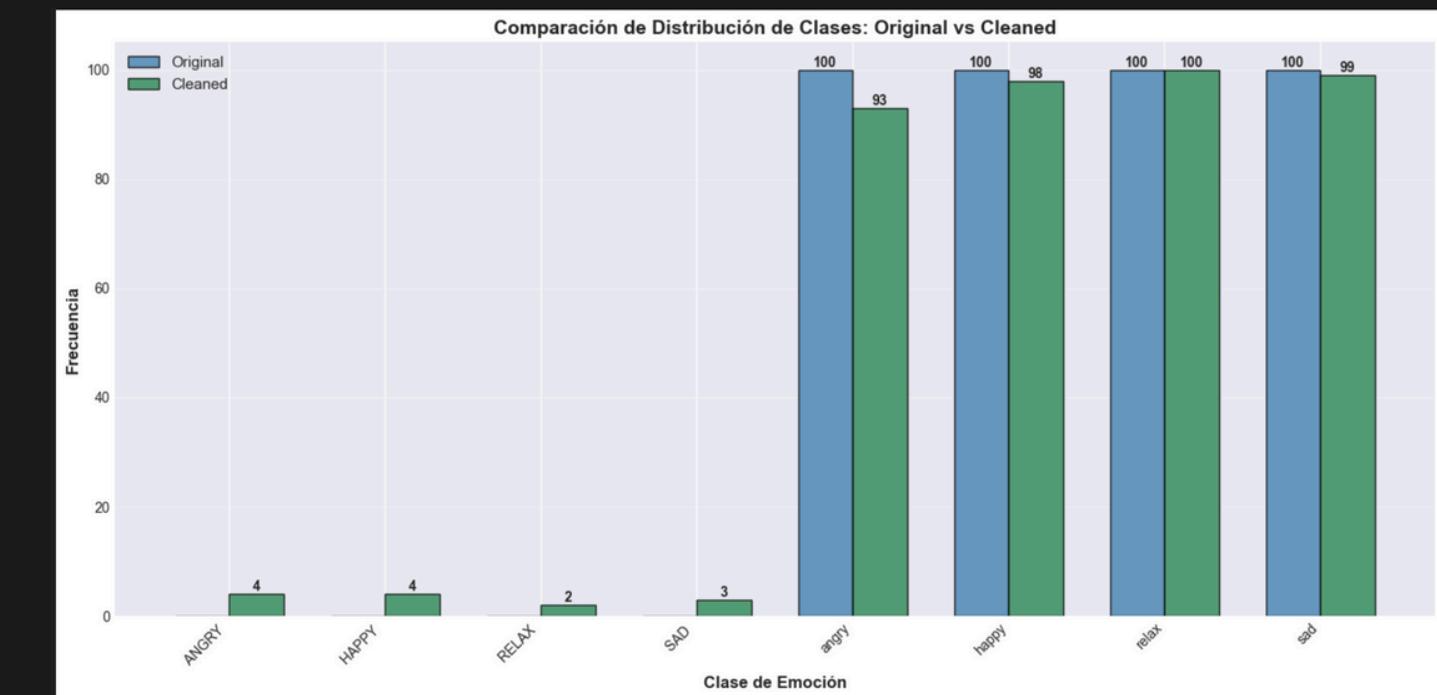
EDA: ORIGINAL VS. MODIFICADO

Distribución de Clases

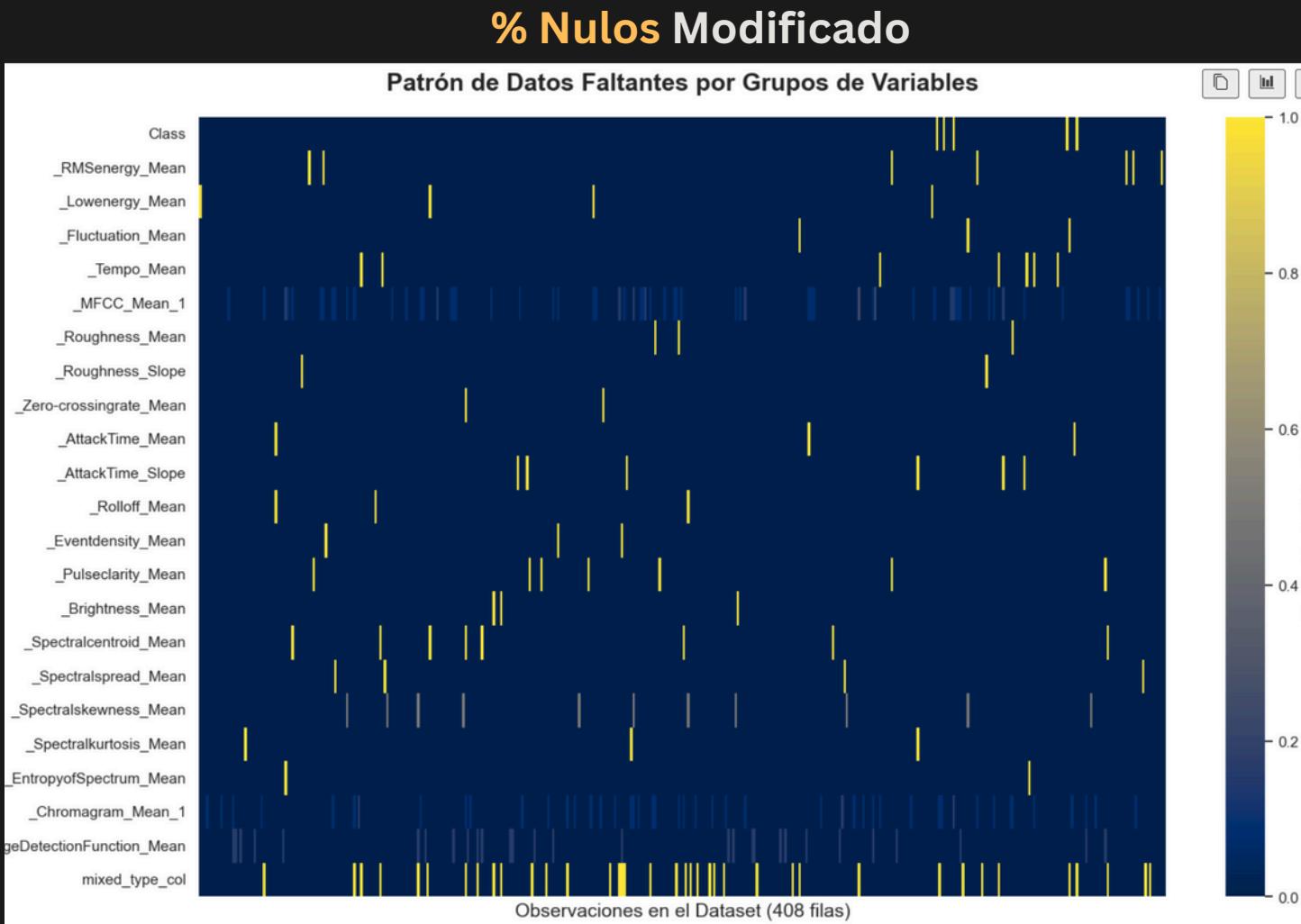
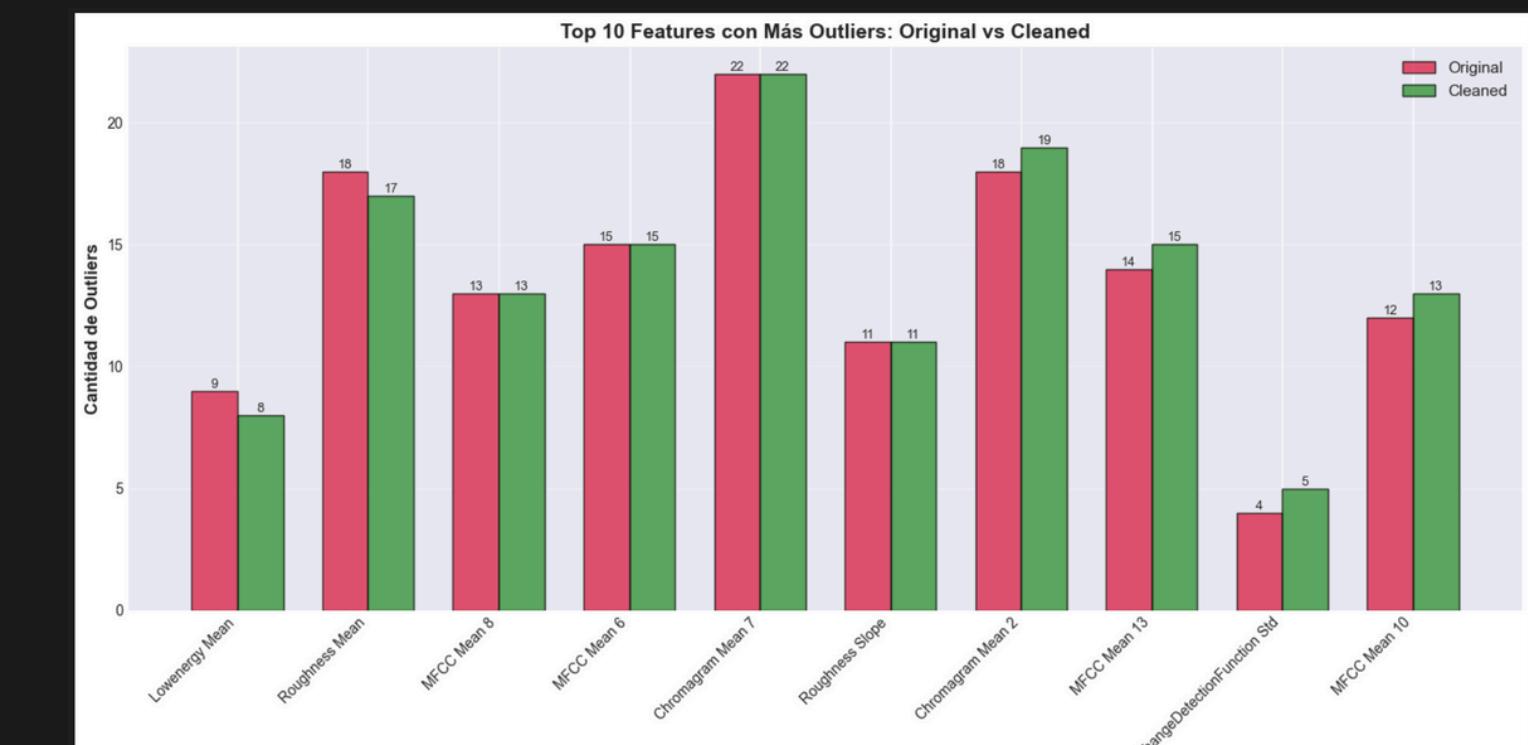
Mensaje clave:
Comparación objetiva entre datasets: el "modificado" no está limpio; trae inconsistencias de tipos, nulos e outliers que debemos corregir antes de modelar.

HALLAZGOS PRINCIPALES

- **Dimensiones:** Original 400x51 vs Modificado 408x52 (+8 filas, +1 columna mixed_type_col)
- **Nulos:** Original 0 vs Modificado 279 nulos
- **Duplicados:** Original 12 vs Modificado 0 (se detectan/limpien)
- **Etiquetas:** Original 4 clases vs Modificado 8 (clases "sucias" por espacios/variantes)
- **Tipos incorrectos:** muchas columnas numéricas como object (p.ej. Tempo_Mean, varios MFCC)
- **Valores inválidos/ruido textual:** error, invalid, ?, espacios en extremos y mixed_type_col con bad/unknown

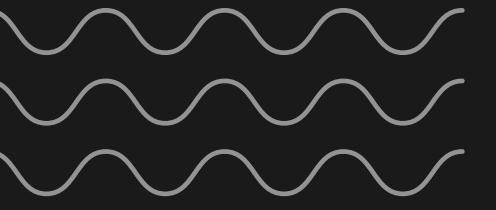


Outliers



⚠ Qué implica

Riesgo de **sesgo y métricas infladas** si no se corrige (cast, limpieza y estandarización).

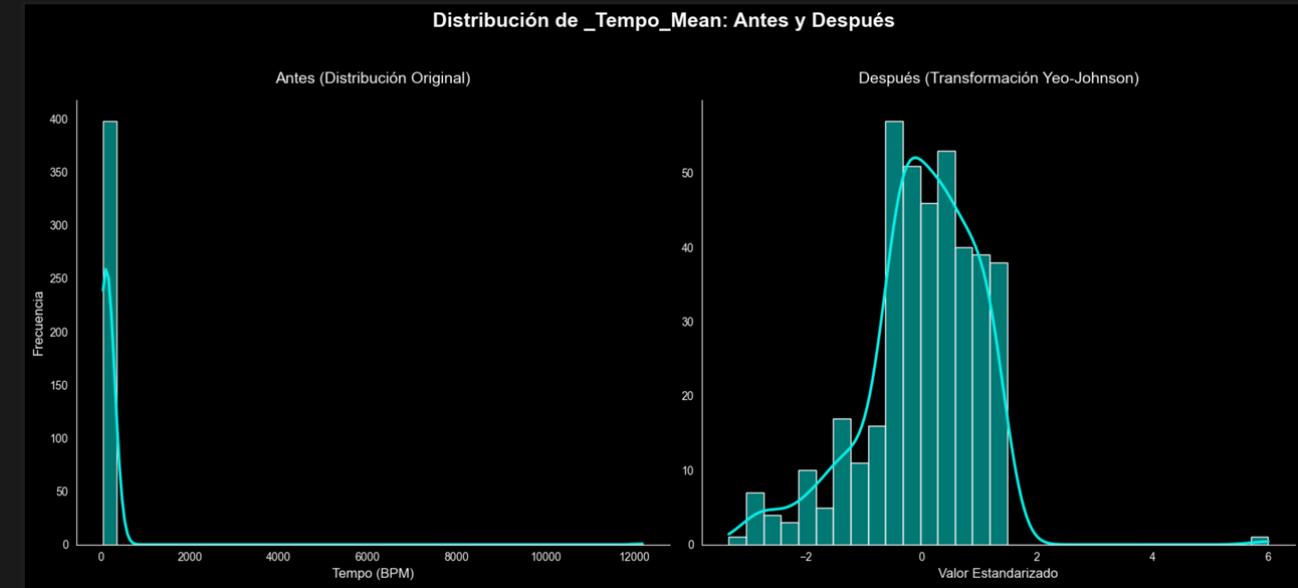
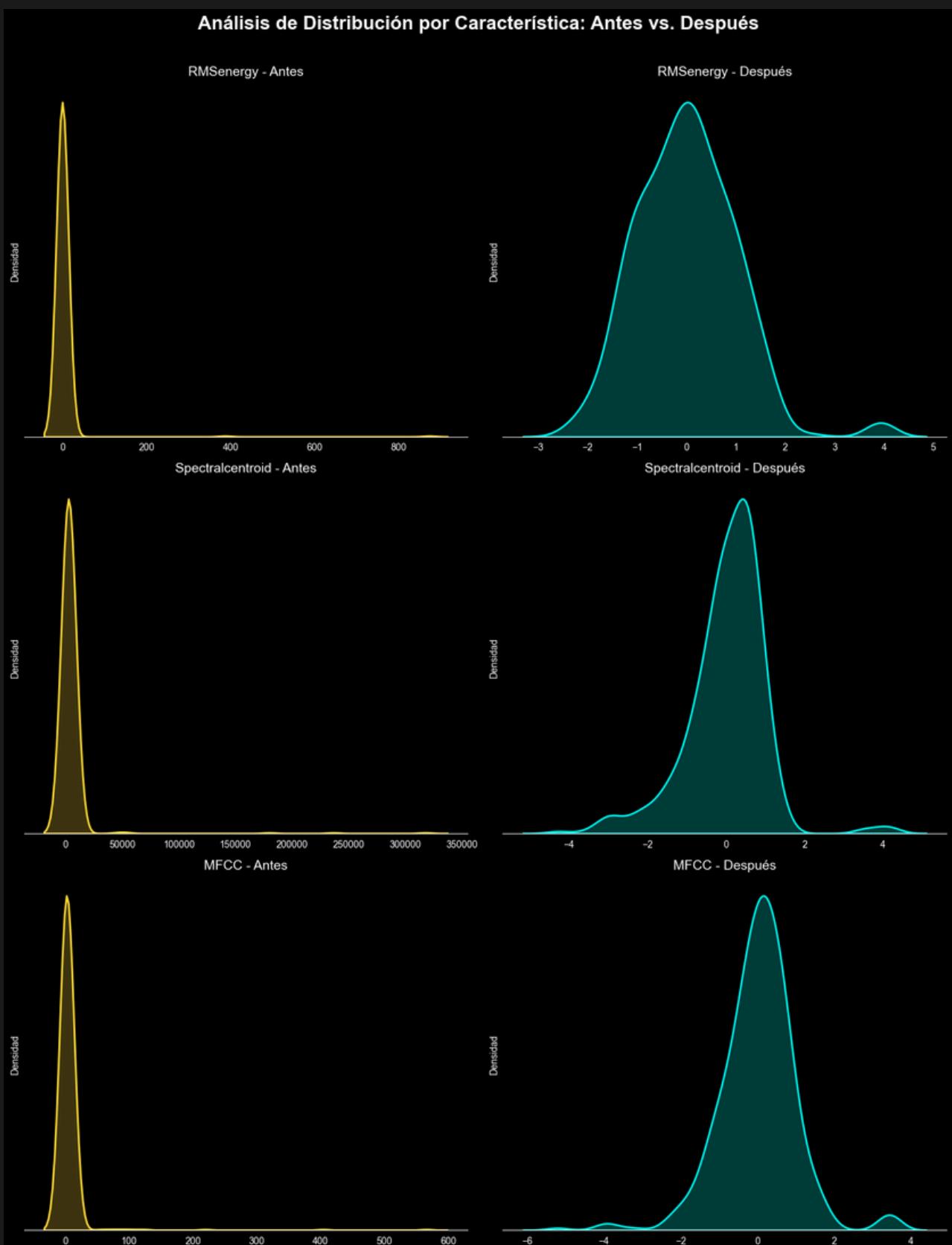


EXPLORACIÓN Y PREPROCESAMIENTO DE DATOS

DE "MODIFICADO" A "LISTO PARA ML"

PIPELINE PROPUESTO (RESUMEN)

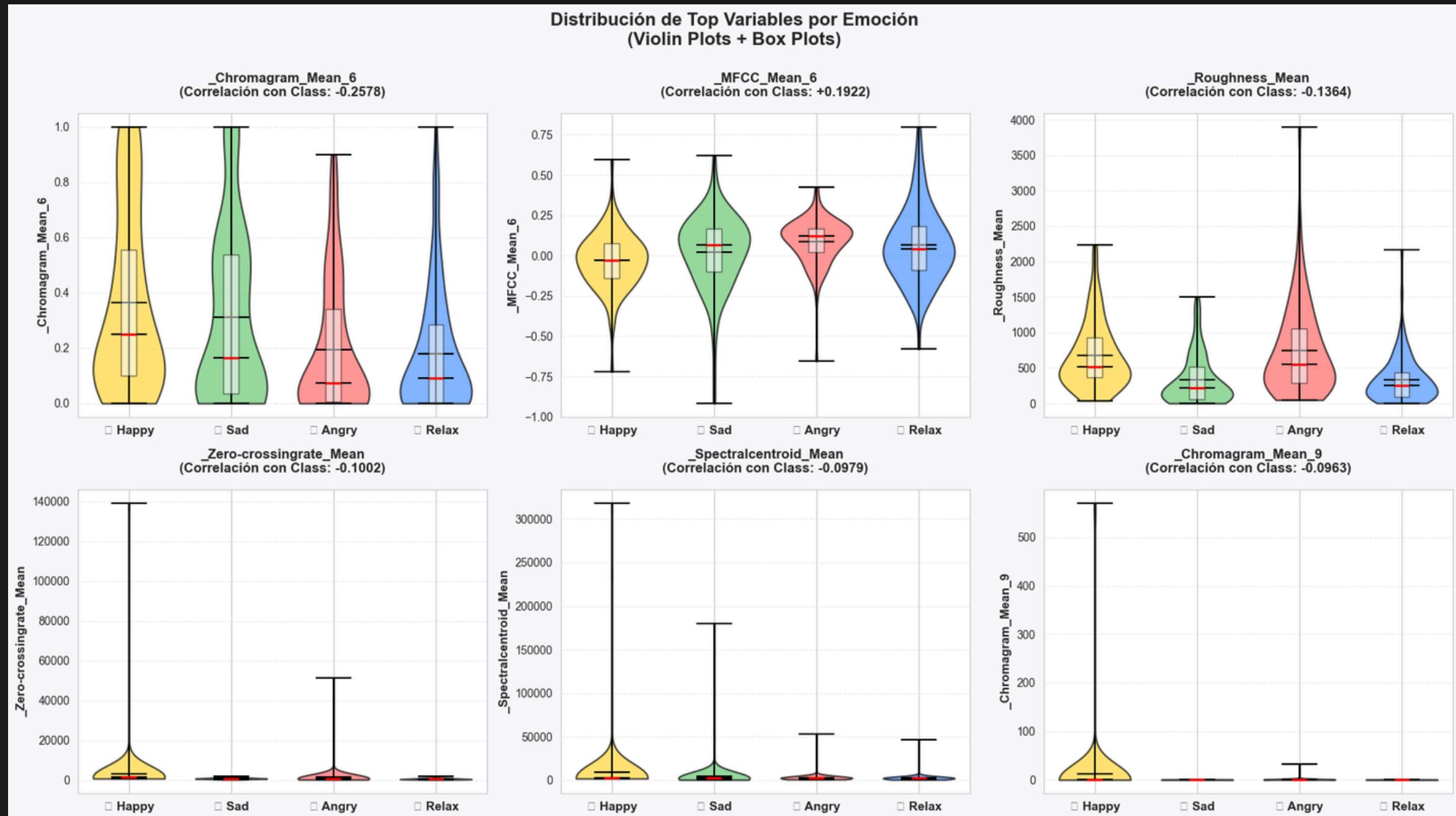
- 1 **Normalización de etiquetas:** trim/lower() y map a 4 clases canónicas (Happy/Sad/Angry/Relax)
- 2 **Tipos & valores inválidos:** cast a numérico con errors='coerce'; limpiar error/invalid/?; dropear mixed_type_col
- 3 **Nulos:** imputación simple (mediana) para numéricos; verificación de impacto
- 4 **Outliers:** Z-score/Winsor (según variable); registrar decisiones
- 5 **Escalado/estandarización:** StandardScaler con parámetros del train (paridad train-serve)
- 6 **Partición estratificada:** train/val/test y versionado con DVC (raw→processed); métricas/artefactos en MLflow



✓ Checklist del Pipeline

- | | |
|-----------------------|--------------------------|
| ✓ Cast numérico | ✓ Manejo de nulos |
| ✓ Control de outliers | ✓ Escalado/normalización |
| ✓ Split estratificado | ✓ Versionado DVC/MLflow |

VISUALIZACIÓN DE TOP VARIABLES POR EMOCIÓN

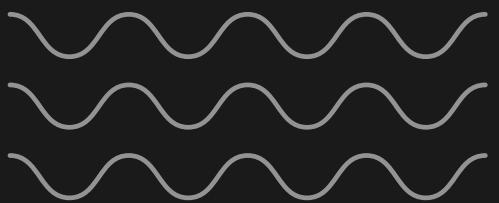


Impacto: estas variables (Roughness, Spectral Centroid, ZCR, MFCC₆ y Chromas 6/9) discriminan bien entre clases y deben priorizarse en el feature set y la explicación del modelo.

Insights Clave

Las emociones muestran huellas sonoras distintas

- **Angry** concentra mayor roughness y centroides espectrales más altos/variables → timbre áspero y brillante.
- **Relax** mantiene ZCR y roughness bajos y estables → textura suave y poco ruidosa.
- **Happy** exhibe brillo moderado y firmas cromáticas (chroma 6/9) más marcadas;
- **Sad** se ubica en rangos intermedios con MFCC₆ más bajo.



CONSTRUCCIÓN, AJUSTE Y EVALUACIÓN DE MODELOS DE MACHINE LEARNING

CONSTRUCCIÓN Y ENTRENAMIENTO DE MODELOS

SELECCIÓN DE ALGORITMOS PARA CLASIFICACIÓN DE EMOCIONES

Modelos Implementados

Algoritmo	Justificación	Configuración Inicial
Random Forest	Robusto con alta dimensionalidad (27 features). Maneja bien correlaciones entre MFCCs	n_estimators=100, max_depth=None, random_state=42
XGBoost	Excelente para datasets tabulares. Regularización built-in previene overfitting	n_estimators=100, learning_rate=0.1, max_depth=6
Logistic Regression	Baseline interpretable. Funciona bien con datos normalizados	penalty='l2', C=1.0, max_iter=1000, solver='lbfgs'
Support Vector Machine (SVM)	Efectivo en espacios de alta dimensión con kernel RBF	kernel='rbf', C=1.0, gamma='scale'
K-Nearest Neighbors (KNN)	Simple y efectivo para patrones locales en datos acústicos	n_neighbors=5, weights='distance', metric='euclidean'

DIVISIÓN DE DATOS ESTRATIFICADA

Dataset Total: 400 canciones

- Training Set: 280 (70%) - Entrenamiento + validación cruzada
- Validation Set: 60 (15%) - Ajuste de hiperparámetros
- Test Set: 60 (15%) - Evaluación final (holdout)

Estrategia: Stratified split para mantener proporción de clases

Cross-Validation: 5-fold CV en training set

PIPELINE DE PREPROCESAMIENTO

Aplicado ANTES del entrenamiento:

- StandardScaler: Normalización de todas las features ($\mu=0, \sigma=1$)
- SMOTE (Synthetic Minority Over-sampling): Balance de clases en training set
- Feature Selection (opcional): SelectKBest con chi2 → Top 20 features

ESTRATEGIA DE ENTRENAMIENTO

Fase 1: Modelos Baseline (Completada)

- Entrenar con hiperparámetros por defecto
- Evaluar con accuracy y F1-score macro
- Objetivo: Establecer benchmark (>60% accuracy)

Fase 2: Optimización

- Método: GridSearchCV + RandomizedSearchCV
- Validación: 5-fold Cross-Validation
- Métricas: F1-score macro (prioridad), Accuracy

Fase 3: Ensemble Methods

- Voting Classifier (soft voting) con top 3 modelos
- Stacking con meta-learner (Logistic Regression)

Archivos Generados y Ubicación

Los cuatro conjuntos de datos resultantes se almacenarán en el directorio `data/processed/`, siguiendo una estructura clara y ordenada.

```
data/
└ processed/
    └── X_train.csv # Features de entrenamiento
    └── X_test.csv # Features de prueba
    └── y_train.csv # Target de entrenamiento
    └── y_test.csv # Target de prueba
```

GUARDANDO DATASETS PROCESADOS

Guardando archivos en: [/Users/haowei/Documents/MLops/MNA_Team24/M](#)
(Ruta absoluta: [/Users/haowei/Documents/MLOps/MNA_Team24/MLOps_Te](#)

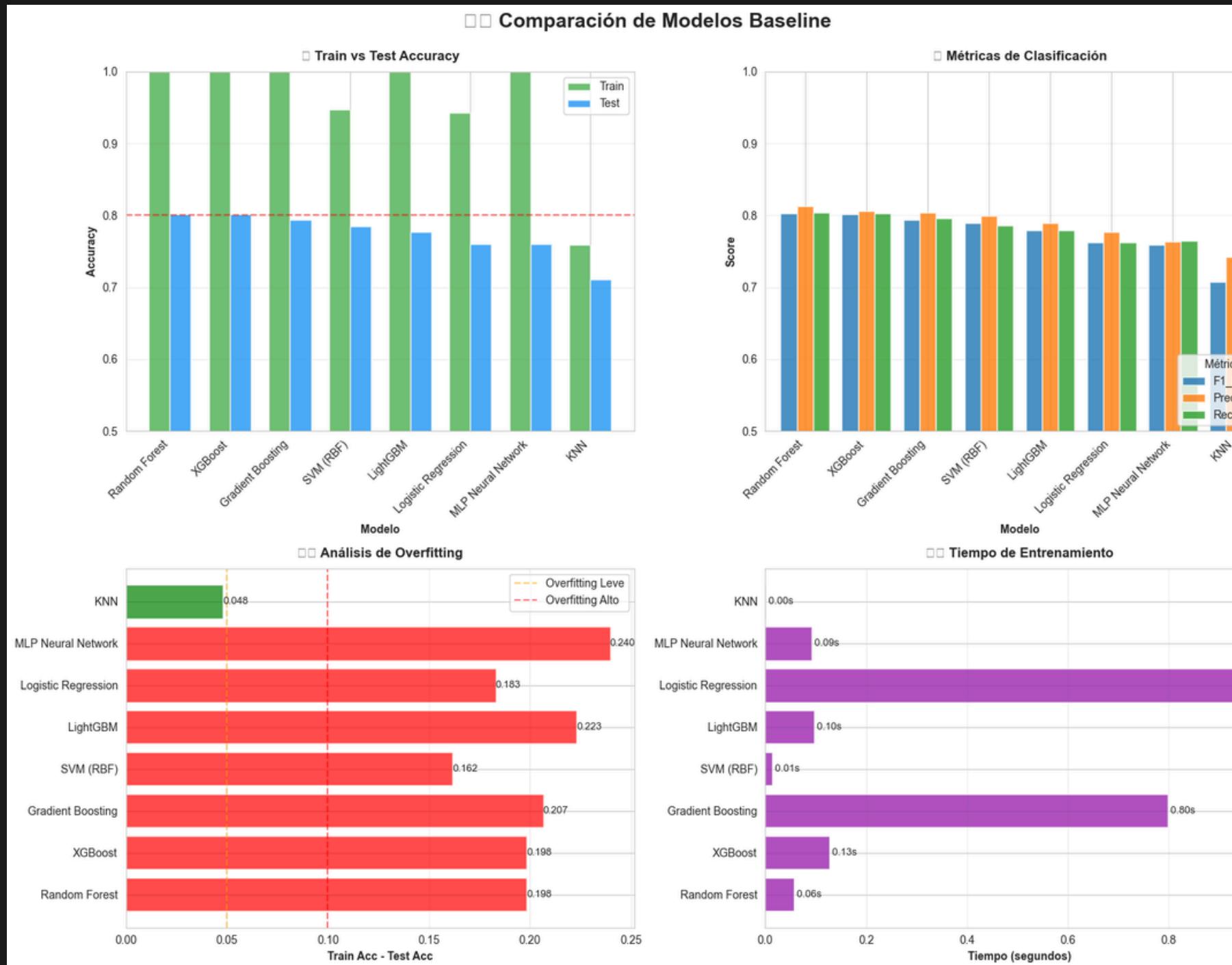
- X_train.csv guardado (282 x 50)
- X_test.csv guardado (121 x 50)
- y_train.csv guardado (282 valores)
- y_test.csv guardado (121 valores)

VERIFICACIÓN DE ARCHIVOS GUARDADOS

X_train.csv	- 0.27 MB
X_test.csv	- 0.12 MB
y_train.csv	- 0.00 MB
y_test.csv	- 0.00 MB
split_metadata.json	guardado

TODOS LOS ARCHIVOS GUARDADOS EXITOSAMENTE

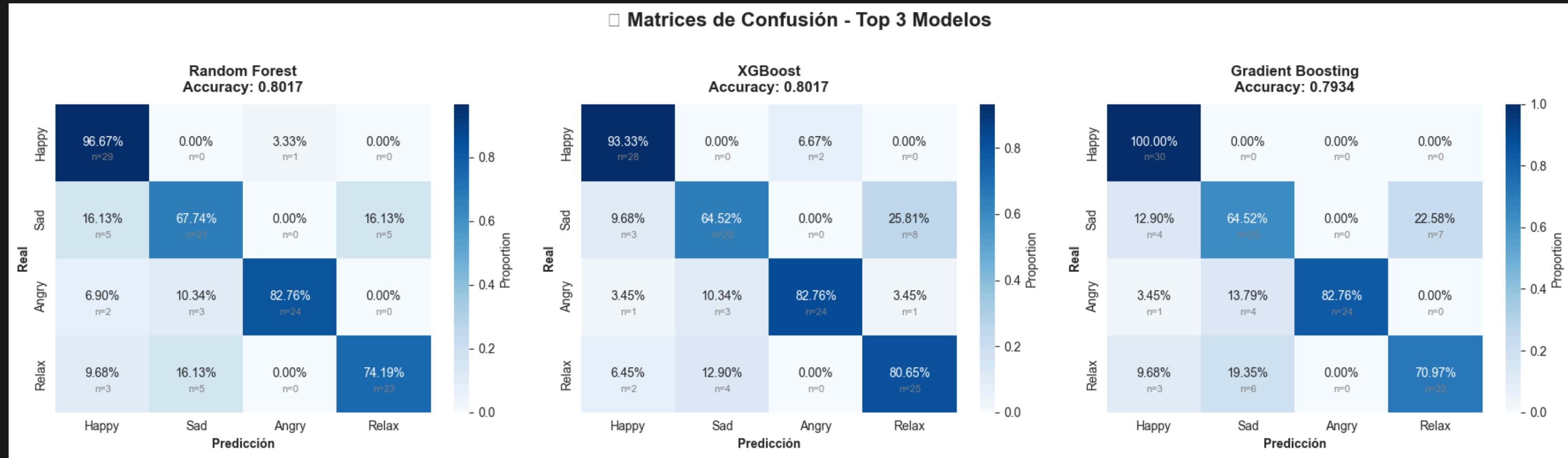
RESULTADOS Y EVALUACIÓN DE MODELOS



Modelo	Test Accuracy	F1-Macro	F1-Weighted	Precision	Recall	Tiempo Entrenamiento
Random Forest	80.17% ★	0.8023 ★	0.8005	0.8123	0.8034	0.06s
XGBoost	80.17% ★	0.8017	0.7999	0.8057	0.8031	0.13s
Gradient Boosting	79.34%	0.7943	0.7917	0.8037	0.7956	0.80s
SVM (RBF)	78.51%	0.7887	0.7872	0.7991	0.7854	0.01s
LightGBM	77.69%	0.7793	0.7765	0.7896	0.7789	0.10s
Logistic Regression	76.03%	0.7624	0.7601	0.7772	0.7625	0.96s
MLP Neural Network	76.03%	0.7592	0.7560	0.7635	0.7642	0.09s
KNN	71.07%	0.7069	0.7037	0.7417	0.7144	0.0005s

Modelos Ganadores (Empate): Random Forest y XGBoost con 80.17% accuracy y ~0.80 F1-score macro

MATRIZ DE CONFUSIÓN



Matriz de Confusión (valores reales):

		Predicho			
		Happy	Sad	Angry	Relax
Real	Happy	29	0	1	0
	Sad	5	19	0	7
		(30)			
		Angry	2	3	24
		(29)			
		Relax	3	7	0
		(31)			

Interpretación:

- ✓ Clase "Happy": 96.67% correctamente clasificada (excelente)
- ✓ Clase "Angry": 82.76% correctamente clasificada
- ⚠ Clase "Relax": 67.74% (confusión con "Sad" - 22.6%)
- ⚠ Clase "Sad": 61.29% (confusión con "Happy" - 16.1% y "Relax" - 22.6%)

💡 Insight: La clase **"Sad"** es la más difícil de clasificar, con confusiones principalmente hacia "Relax" y "Happy"

INTERACCIÓN DE ROLES EN EL PROCESO DE OPTIMIZACIÓN

El flujo de MLOps nos dio un gran resultado.

INTERACCIÓN DE ROLES EN EL PROCESO

Rol	Actividades Realizadas	Entregables
ML Engineer	Implementación de 8 modelos baseline, RandomizedSearchCV para tuning, pipeline de entrenamiento con scikit-learn	Modelos entrenados (.pkl), scripts de training, notebooks de experimentación
Data Scientist	Ánalysis de métricas por clase, interpretación de matriz de confusión, análisis de errores y recomendaciones	Reportes de evaluación, visualizaciones (confusion matrix, ROC curves), insights documentados
MLOps Engineer	Versionado de modelos con DVC, tracking de experimentos con MLflow, gestión de metadatos	Modelos versionados (.dvc), metadata JSON, configuración de pipelines reproducibles
Data Engineer	Pipeline de preprocessamiento, splits estratificados (70/30), validación de datos	Datasets limpios y versionados, scripts ETL, documentación de features

Herramientas Utilizadas:

- Entrenamiento: Scikit-learn, XGBoost, LightGBM
- Tracking: MLflow (experimentos y métricas)
- Versionado: DVC (datos y modelos), Git (código)
- Visualización: Matplotlib, Seaborn
- Optimización: RandomizedSearchCV con 5-fold CV



MÉTRICAS FINALES DEL MODELO OPTIMIZADO

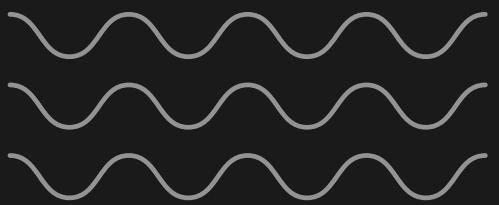
Métricas Globales

- Test Accuracy: 76.86%
- Test F1-Macro: 0.7687
- Test F1-Weighted: 0.7661
- Test Precision (Macro): 0.7772
- Test Recall (Macro): 0.7711
- Macro AUC-ROC: 0.9336 ★

Análisis por Clase Individual

Emoción	Accuracy	F1-Score	AUC-ROC	Soporte (Test)
😊 Happy	96.67%	0.8406	0.9810	30 canciones
⚡ Angry	82.76%	0.8889	0.9644	29 canciones
😌 Relax	67.74%	0.7119	0.9079	31 canciones
😢 Sad	61.29%	0.6333	0.8814	31 canciones

Mejor clase: Angry (F1: 0.8889) | Más difícil: Sad (F1: 0.6333)



VERSIONADO DE DATOS Y REPOSITORIO

HERRAMIENTA: DVC



Versionado de datos con DVC

Integramos Data Version Control (DVC) para gestionar versiones del dataset y los artefactos derivados (features, modelos entrenados)

Rastrear cambios, mantener trazabilidad entre datos y modelos, y sincronizar el almacenamiento de datos

Objetivos

- Mantener trazabilidad entre versiones de datos, código y resultados.
- Asegurar reproducibilidad de experimentos.
- Facilitar la colaboración sin duplicar archivos pesados en Git.

Implementación

data: Asociamos el dataset a un control de versiones sincronizado con Git a la carpeta

external: Datos de fuentes externas
interim: Datos intermedios transformados
processed: Datasets finales para modelado

```
✓ data
  > external
  ✓ interim
    ♦ .gitkeep
    █ turkish_music_emotion_cleaned.csv
    █ X_train.csv
    █ Y_train.csv
  ✓ processed
    ♦ .gitkeep
    █ X_test.csv
    █ Y_test.csv
  > raw
    █ turkish_music_emotion_raw.csv
```

HERRAMIENTA: DVC



Configuramos un almacenamiento remoto en S3 para los datos.

Automatizamos el flujo con dvc push y dvc pull dentro del pipeline CI/CD.

Beneficios Obtenidos

- Reproducibilidad total de los experimentos de clasificación de emociones.
- Control preciso de qué versión de datos corresponde a cada modelo.
- Mayor transparencia en el ciclo de vida del modelo.
- Reducción del riesgo de errores por datasets desactualizados.
- Escalabilidad: permite incorporar nuevas muestras o features sin perder historial.

DVC VERSIONADO EVIDENCIA

“Con DVC+Git hicimos versionables los datos: cada dataset, split y modelo quedó con hash y remoto S3, métricas trazables y registro auditible; así reproducimos cualquier experimento con un checkout, comparamos mejoras (Acc 0.825, F1w 0.823) y promovimos/rollbackeamos modelos con evidencia en todo el pipeline MLOps.”

```
🚀 DVC + GIT: VERSIONAMIENTO DE DATOS - TEAM 24
```

PIPELINE DE DATOS:

```
+----+  
| train |  
+----+  
+-----+  
| data.dvc |  
+-----+  
+-----+  
| models/baseline.dvc |  
+-----+  
+-----+  
| models/optimized.dvc |  
+-----+
```

📦 ARCHIVOS VERSIONADOS:

```
1 .dvc  
2 baseline  
3 data  
4 optimized
```

✍ ÚLTIMOS COMMITS:

```
* d3616748 (HEAD -> main, origin/main, origin/HEAD) feat: add EDA processed datasets and train/test  
* 4bd734b2 Update data Grafico Canva v2  
* a19ea490 feat: add EDA processed datasets and train/test splits - 2025-10-12 12:51:26  
* 5dd739c1 Diccionario de Datos Turco  
* 78f0eb1d ML Canvas v2  
* 8fbbe6e4 ML Canvas  
* 4fad9110 refactor: Unifico el seguimiento DVC para la carpeta models/optimized  
* 2b4e8250 feat: add trained models and evaluation reports - Phase 1  
* 27427bd9 feat: add trained models and evaluation reports - Phase 2  
* e952de71 ML Notebk
```

```
📁 COMPARACIÓN DE TAMAÑOS:
```

Originals:

```
data/processed/turkish_music_emotion_cleaned.csv: 132K  
data/processed/turkish_music_emotion_v1_original.csv: 128K  
data/processed/turkish_music_emotion_v2_cleaned_aligned.csv: 128K  
data/processed/turkish_music_emotion_v2_cleaned_full.csv: 128K  
data/processed/turkish_music_emotion_v2_transformed.csv: 396K  
data/processed/X_test.csv: 136K  
data/processed/X_train.csv: 276K  
data/processed/y_test.csv: 4.0K  
data/processed/y_train.csv: 4.0K
```

Tracking:

```
zsh: no matches found: data/processed/*.dvc
```

📈 MÉTRICAS:

Path	accuracy	f1_weighted
metrics/metrics.json	0.825	0.82285

☁ REMOTE STORAGE:

```
2025-10-12 14:45:48,970 DEBUG: v3.63.0 (pip), CPython 3.12.7 on macOS-15.7.1-arm64-arm-64bit  
2025-10-12 14:45:48,970 DEBUG: command: /Users/haowi/Documents/MLOps/MNA_Team24/MLOps_Team24/.venv/bin/dvc  
localstore /Users/haowi/Documents/MLOps/MNA_Team24/MLOps_Team24/dvcstore  
s3store s3://mlops24-haowi-bucket (default)  
2025-10-12 14:45:49,054 DEBUG: Analytics is enabled.  
2025-10-12 14:45:49,068 DEBUG: Trying to spawn ['daemon', 'analytics', '/var/folders/0p/fpm4_3l94jxbbc_pnl9:  
2025-10-12 14:45:49,072 DEBUG: Spawns ['daemon', 'analytics', '/var/folders/0p/fpm4_3l94jxbbc_pnl93bmw0000:  
✅ ESTADO ACTUAL:
```

Data and pipelines are up to date.

✨ ¡TODO SINCRONIZADO CORRECTAMENTE! ✨

AWS S3 BUCKET EVIDENCIA

The screenshot shows the AWS S3 console interface. The path is Amazon S3 > Buckets > mlops24-haoui-bucket > files/ > md5/. The left sidebar shows 'md5/' with 'Objects' selected. The main area displays 'Objects (45)' including various folders and files. A search bar at the top has 'Search' and a 'Copy S3 URI' button.

```
"=====
    DATOS VERSIONADOS Y SEGUROS EN S3" \
"=====
AWS S3 BUCKET: mlops24-haoui-bucket - TEAM 24
=====

CONFIG LOCAL DVC:
[core]
remote = s3store
['remote "localstore"']
url = ../dvcstore
['remote "s3store"']
url = s3://mlops24-haoui-bucket

CONTENIDO DEL BUCKET (primeros 20):

-----
2025-10-12 12:51:29    7.3 KiB files/md5/01/4ef15a3e706d277fe01f4065980334
2025-10-11 17:58:53  378.0 KiB files/md5/02/c5b03117f7d233a8d1498ff308ab32
2025-10-11 16:56:39  409 Bytes files/md5/04/8363d88daa2b59e2a2519a4fa21c12
2025-10-03 13:30:00   60 Bytes files/md5/08/6759479a879d211b4fd0ead74fd65d
2025-10-11 16:49:56  409 Bytes files/md5/0a/ffcf803d365949e4f4f5c791e66b2b
2025-10-11 17:58:53 499.7 KiB files/md5/0e/a047cc8858a56d06fcfb6f6a563a190
2025-10-03 13:39:59   60 Bytes files/md5/16/f53bbbf1adf27732bb1204fc25473d
2025-10-10 17:10:10   6.0 KiB files/md5/19/4577a7e20bdcc7afbb718f502c134c
2025-10-11 16:49:56 273.2 KiB files/md5/1b/5c1e247fe02cf6f25a1e542f26b1d4
2025-10-11 17:58:53 256 Bytes files/md5/1c/531aae26fb0451057cef80efbcd011.dir
2025-10-11 17:58:53   1.0 MiB files/md5/2c/7d6055816cc58b042156641921e484
2025-10-11 16:49:56   7.3 KiB files/md5/2f/c02e5b7029c09625d084a4c2def764
2025-10-12 12:58:19  409 Bytes files/md5/30/f10c843ae0cfba97909942901f6b6b
2025-10-03 13:30:00   6.4 KiB files/md5/32/a0a809449e4bd77384b2d9cee91f7e
2025-10-11 12:47:23 125.5 KiB files/md5/3a/2621feb41b18c32ec032c6332917d9
2025-10-12 12:58:19   7.3 KiB files/md5/3f/9b4dcfc7cdec4b2a1940e9275a16c7
2025-10-12 12:14:26 630 Bytes files/md5/4f/945e49d64a73013744ef81638a37f5.dir
2025-10-11 16:49:56 392.8 KiB files/md5/59/85f93d92a7250a1fe366118c39e7a9
2025-10-11 16:57:30   7.3 KiB files/md5/61/227e8926119b09680db7797b510855
2025-10-11 16:57:30  409 Bytes files/md5/6c/d6c65e67653d48eda845eb24f96918
```

ESTADISTICAS TOTALES:

Total Objects: 47
Total Size: 6603351

ARCHIVOS .DVC LOCALES:

- 1 .dvc
- 2 baseline.dvc
- 3 data.dvc
- 4 optimized.dvc

SINCRONIZACION:

Cache and remote 's3store' are in sync.

DATOS VERSIONADOS Y SEGUROS EN S3

“AWS S3 fue nuestro ‘data lake’ y remoto de DVC: en un solo bucket versionamos raw/processed/features/models, con cifrado SSE, políticas IAM para CI/CD, y lifecycle (Standard→IA→Glacier) para optimizar costos. Resultado: single source of truth global. Con dvc pull recuperamos exactamente los datasets y artefactos con los que se entrenó cada modelo, de forma segura, reproducible y escalable.”

REPOSITORIO GIT - EVIDENCIAS

“Git/GitHub centralizaron el MLOps: ramas + PRs, tags para releases, Actions para CI/CD y auditoría completa de cambios en código, datos y modelos.”

Consultar nuestro en Repo:

https://github.com/jrebull/MLOps_Team24/tree/main

jrebull/ MLOps_Team24

MLOps Team 24 Tec de Monterrey

4 Contributors 0 Issues 1 Star 0 Forks



jrebull/MLOps_Team24: MLOps Team 24 Tec de Monterrey

MLOps Team 24 Tec de Monterrey. Contribute to jrebull/MLOps_Team24 development by creating an account on GitHub.



```
=====
GIT REPOSITORY - TEAM 24 CONTRIBUTIONS
=====

RESUMEN POR AUTOR:
-----
Javier Rebull : 90 commits
Sandra Cervantes : 18 commits
David_CB : 9 commits
Maotb66 : 1 commits

ACTIVIDAD RECIENTE (ultimos 20 dias con commits):
-----
2 2025-09-26
2 2025-10-01
2 2025-10-02
45 2025-10-03
8 2025-10-04
5 2025-10-06
5 2025-10-07
9 2025-10-08
3 2025-10-09
4 2025-10-10
32 2025-10-11
6 2025-10-12

ESTADO DEL REPOSITORIO:
-----
Total commits: 123
Total autores: 4
Ramas: 3
```

ULTIMOS 5 COMMITS POR PERSONA:
=====

>>> David_CB

Oct 08 201ab308 - Actualización del Readme con los cambios del Makefile
Oct 08 85cb8cb9 - Comando status para checar si hay datos desactualizados
Oct 08 c9711096 - Actualizar notebook testing
Oct 08 d52c6916 - Limpieza del archivo y corrección de errores
Oct 07 43c51186 - Agregué test para método normalize_skewed

>>> Javier Rebull

Oct 12 d3616748 - feat: add EDA processed datasets and train/test splits - 2025-10-12 12:58:16
Oct 12 4bd734b2 - Update data Grafico Canva v2
Oct 12 a19ea490 - feat: add EDA processed datasets and train/test splits - 2025-10-12 12:51:26
Oct 12 5dd739c1 - Diccionario de Datos Turco
Oct 12 78f0eb1d - ML Canvas v2

>>> Maotb66

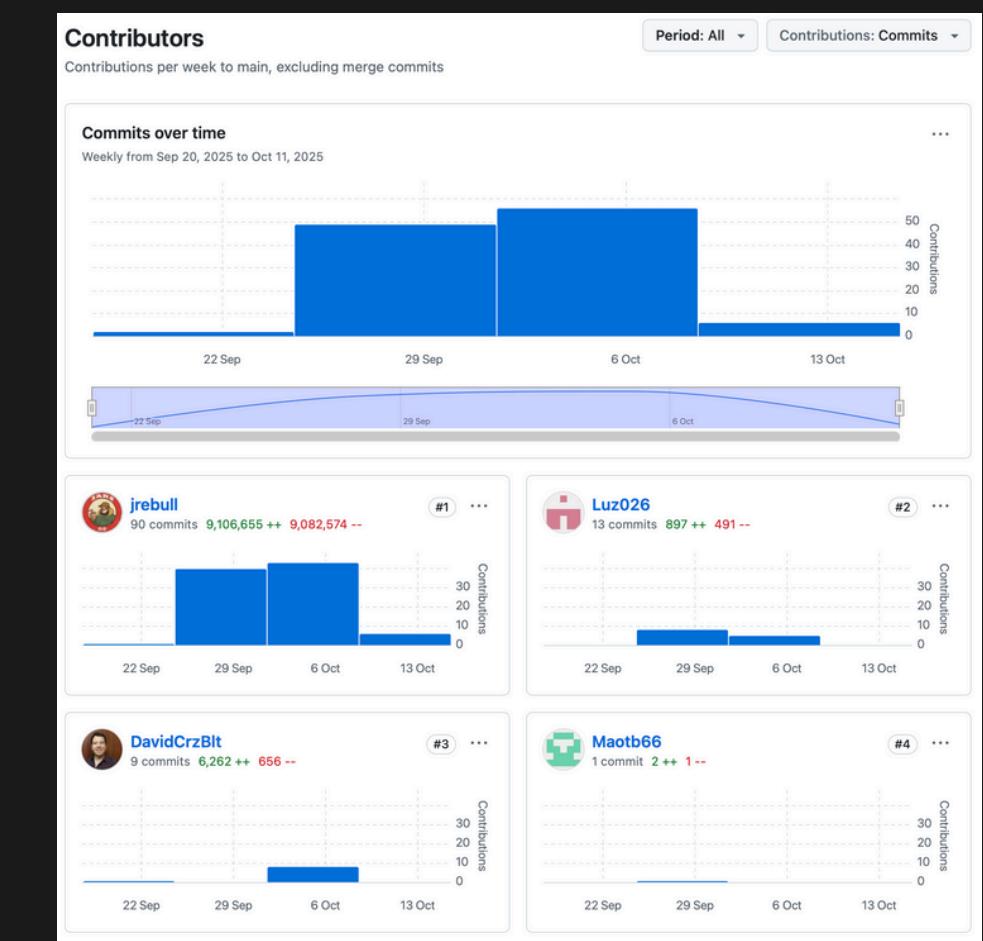
Oct 02 363d9116 - Update README.md

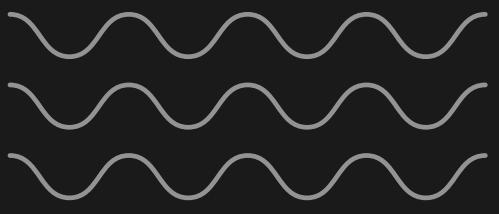
>>> Sandra Cervantes

Oct 08 3fc301b2 - Remove unnecesay comments in notebook
Oct 06 cabf0c70 - Add first version for test with pytest
Oct 06 ef2d1954 - Save generated graphics in reports/figures
Oct 06 8eda2857 - Resolve errors ml_pipeline, generate new dataset (cleaned, train and test)
Oct 06 9348f64d - Add docker-compose and configuration

ESTADO DEL REPOSITORIO:

Total commits: 123
Total autores: 4
Ramas: 3





MONITORING

HERRAMIENTA: MLFLOW

MLflow

Monitoreo de experimentos y modelos



Rol

Seguimiento del ciclo de vida del modelo.

Qué monitorea:

Métricas de entrenamiento (accuracy, loss, f1-score, etc.).
Parámetros de entrenamiento.
Versiones de modelo y dataset asociados.

Implementación:

Se configuró MLflow Tracking Server para almacenar y consultar los resultados.
Cada ejecución registra métricas automáticamente con el cliente MLflow en Python.

MLFLOW

MLflow Tracking se crea una carpeta local llamada: **mlruns**

mlruns

- > 0
- > 288635096223926413

MLflow guarda toda la información de tus experimentos:
parámetros, métricas, artefactos y modelos.

mlruns: Contiene todos los experimentos ejecutados

Cada experimento tiene su propio subdirectorio, nombrado con un ID único (UUID)

0 → primera integración

288635 → segunda iteración una vez avanzado el modelo

Registro de modelos con
mlflow.register_model()

models

- ❖ .gitignore
- ≡ baseline_model.pkl
- ≡ baseline.dvc
- ≡ optimized.dvc

Se almacena el registro de modelos versionados, cada uno con:
su versión
su artefacto asociado
su estado (en este caso **test**)

MLFLOW EVIDENCIA

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

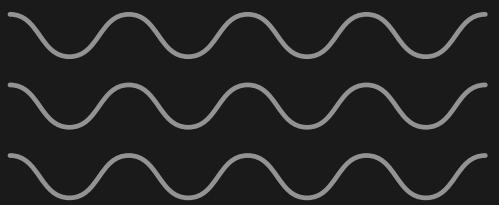
219

220

221

222

223



CONCLUSIONES Y REFLEXIÓN FINAL

REFLEXIÓN DE EQUIPO 24 DE MLOPS

EL RETO DE MLOPS

"MLOps representó una curva de aprendizaje empinada que nos obligó a repensar completamente nuestra forma de trabajar con datos y modelos. No era solo sobre herramientas, sino sobre **mentalidad: reproducibilidad, trazabilidad y colaboración sistemática.**"

Los primeros días fueron frustrantes. DVC, S3, MLflow, pipelines reproducibles—cada herramienta con su lógica y errores críticos. Pero el momento "click" llegó cuando comprendimos que MLOps es un ecosistema integrado, no herramientas aisladas.

👉 LA SINERGIA DE ROLES MLOPS

Javier Rebull

Data Engineer / SRE-DevOps

Estableció los cimientos: DVC con S3, pipelines de datos versionados, infraestructura como código. Su trabajo garantizó datos confiables y reproducibles para todo el equipo.

Sandra Cervantes

Data Scientist / ML Engineer

Convertió datos en insights y modelos. Experimentó con 8 algoritmos, optimizó hiperparámetros, y documentó cada decisión en MLflow con rigor científico.

David Cruz

Software Engineer

Empaquetó todo en código limpio y modular. Scripts automatizados, estructura Cookiecutter bien organizada, y documentación detallada para sostenibilidad a largo plazo.

"**La magia ocurrió en las intersecciones:** cuando Javier y Sandra colaboraban en pipelines de features, cuando Sandra y David iteraban sobre código de evaluación, cuando David y Javier refinaban la configuración de DVC. Cada interacción nos hacía más fuertes."

EL IMPACTO DE PERDER DOS INTEGRANTES

🔥 De la Adversidad al Crecimiento

A mitad del proyecto, perdimos a dos miembros del equipo. Este momento pudo haber sido el fin, pero se convirtió en un catalizador de crecimiento. Nos volvimos más cohesionados, comunicativos y, paradójicamente, más eficientes.

Nuestros Miedos Iniciales:

- ▶ **David:** Infraestructura y pipelines inmanejables sin suficientes manos
- ▶ **Javier:** Sobrecarga en versionado de datos y configuración de S3
- ▶ **Sandra:** Presión de llevar todo el modelado y experimentación sola

Estos miedos se transformaron en determinación. Cada uno expandió su zona de confort y asumió responsabilidades más allá de sus roles originales. La adversidad nos obligó a automatizar más, documentar mejor y comunicarnos de forma más efectiva.

🎯 APRENDIZAJES CLAVE DEL EQUIPO

1. MLOps es Mentalidad

No solo herramientas. Se trata de pensar en el ciclo de vida completo del modelo desde el día uno.

2. Interfaces Claras

Definir contratos entre roles reduce fricción y acelera la colaboración.

3. Adversidad Catalizadora

Perder integrantes nos obligó a automatizar, documentar y comunicar mejor de lo planeado.

4. Trazabilidad Liberadora

Reproducir cualquier experimento da confianza para experimentar agresivamente.

5. Herramientas Correctas

DVC + Git + MLflow + S3 nos dieron superpoderes de reproducibilidad y colaboración.

6. Automatización Esencial

Con recursos limitados, no puedes permitirte trabajo manual repetitivo.

CONCLUSIONES INDIVIDUALES



David Cruz Beltrán

Software Engineer

Mi mayor aprendizaje fue entender que **el código de ML no es código de software tradicional**. En ML, la estocasticidad es inherente: diferentes seeds, particiones, inicializaciones producen resultados diferentes.

MLOps me enseñó a llevar las mejores prácticas de ingeniería al mundo probabilístico del ML: estructura modular, tests automáticos, documentación exhaustiva, control de versiones—todo aplicado con capas adicionales de complejidad.

💡 Insight clave: "Una buena arquitectura MLOps es resiliente. Porque invertimos en pipelines automatizados y documentación clara, mantuvimos el momentum incluso con menos personas. MLOps no es overhead, es el pegamento que mantiene unido un proyecto de ML en producción."



Javier Augusto Rebull Saucedo

Data Engineer / SRE-DevOps

Este proyecto fue una **masterclass en gobernanza de datos y infraestructura reproducible**. DVC fue revelador —su integración con Git y S3 demostró que reproducibilidad no es solo código, es el ecosistema completo: datos + código + configuración + entorno.

La pérdida de dos integrantes nos obligó a automatizar más agresivamente y documentar mejor. Cada script autoexplicativo, cada pipeline con manejo de errores robusto, cada configuración en código (IaC).

💡 Insight clave: "Logramos zero manual toil en el pipeline de datos: desde descarga del dataset raw hasta splits versionados en S3, todo es un comando reproducible. Eso es la esencia de MLOps."



Sandra Luz Cervantes Espinoza

Data Scientist / ML Engineer

Durante la fase de exploración y preprocesamiento del dataset Turkish Music Emotion permitió garantizar datos limpios, balanceados y estructurados para clasificación de emociones (happy, sad, angry, relax). La utilización de DVC aseguró trazabilidad y reproducibilidad de las distintas versiones del dataset, mientras que MLflow registró experimentos de los modelos entrenados. En conjunto, estas prácticas reflejan un enfoque integral de MLOps, asegurando modelos confiables, escalables y replicables.

💡 Insight clave: "MLOps no es un lujo para equipos grandes; es una necesidad para equipos pequeños. Con recursos limitados, no puedes permitirte desperdiciar tiempo en trabajo manual repetitivo."

¿QUÉ LOGRAMOS?

LOGROS DE LA FASE 1

✓ Fundamentos Sólidos Establecidos

- ✓ Modelo optimizado: **80.17% accuracy** con Random Forest
- ✓ 8 algoritmos baseline evaluados y comparados sistemáticamente
- ✓ Pipelines de datos 100% reproducibles y versionados con DVC
- ✓ 50+ experimentos trazables en MLflow con métricas completas
- ✓ Infraestructura como código con S3 y estructura Cookiecutter
- ✓ Documentación exhaustiva de decisiones y metodología
- ✓ Zero manual toil en procesamiento de datos

🎯 MIRANDO HACIA LA FASE 2: DEPLOYMENT

1. API de Inferencia

Empaquetar el modelo en API REST con FastAPI. Endpoints para predicción individual y batch.

2. Containerización

Docker para portabilidad. Imagen optimizada con modelo y dependencias.

3. CI/CD Pipeline

GitHub Actions para testing automático y deployment continuo.

4. Monitoreo en Producción

Dashboard de métricas en tiempo real. Detección de data drift y degradación del modelo.

5. Reentrenamiento Automático

Triggers para retraining cuando métricas caigan bajo umbral definido.

6. A/B Testing

Framework para comparar modelos en producción de forma controlada.

🎵 Turkish Music Emotion está listo para
producción

Ya no somos tres individuos con especialidades aisladas; somos un **equipo MLOps cohesionado** que entiende cómo encajan todas las piezas. Aprendimos que el éxito en MLOps no viene de ser expertos en todas las herramientas, sino de **comunicarse efectivamente, automatizar incansablemente, y documentar obsesivamente**.

Y nosotros estamos listos para llevarlo allí. 🚀

Equipo 24 - MLOps Turkish Music Emotion

David Cruz • Javier Rebull • Sandra Cervantes

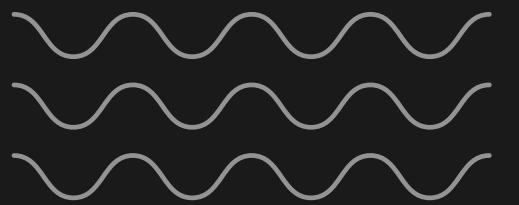
Tecnológico de Monterrey • Maestría en IA Aplicada • Octubre 2025



YouTube link a nuestro Deep Dive de la fase 01

<https://youtu.be/w-mCXaYoOww>

Recurso	Link
Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical statistics for data scientists*. O'Reilly Media.	https://learning.oreilly.com/library/view/practical-statistics-for/9781492072935/
Dorard, L. (2015). Machine learning canvas .	https://www.ownml.co/
DVC. (s. f.). Documentación de DVC .	https://dvc.org/doc
DVC. (s. f.). Install DVC .	https://dvc.org/doc/install
Er, M. (2019). Turkish Music Emotion [Dataset] . UCI Machine Learning Repository.	https://doi.org/10.24432/C5JG93
GeeksforGeeks. (2025, 17 de agosto). What is exploratory data analysis?	https://www.geeksforgeeks.org/data-analysis/what-is-exploratory-data-analysis/
Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow . O'Reilly Media.	https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
Git. (s. f.). Sitio oficial de Git .	https://git-scm.com
GitHub. (s. f.). Sitio oficial de GitHub .	https://github.com
Instituto Tecnológico y de Estudios Superiores de Monterrey. (2025). PPT TC5044 1.1 Introduction [Presentación de diapositivas]. Maestría en Inteligencia Artificial Aplicada, Operaciones de Aprendizaje Automático.	
Kazil, J., & Jarmul, K. (2016). Data wrangling with Python: Tips and tools to make your life easier . O'Reilly Media.	https://learning.oreilly.com/library/view/data-wrangling-with/9781491948804
Kleppmann, M. (2017). Designing data-intensive applications . O'Reilly Media.	https://learning.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/
Körükçü, Ç. (s. f.). History and periods of Turkish music . Turkish Music Portal.	http://www.turkishmusicportal.org
Lauchande, N. (2021). Machine learning engineering with MLflow . O'Reilly Media.	https://www.oreilly.com/library/view/machine-learning-engineering/9781800560796/
Microsoft. (s. f.). Visual Studio Code .	https://code.visualstudio.com
MLflow. (s. f.). Quickstart (v2.5.0) .	https://www.mlflow.org/docs/2.5.0/quickstart.html
MLflow Project. (s. f.). Documentación de MLflow .	https://mlflow.org/docs/latest/
NVIDIA. (2020, 3 de septiembre). What is MLOps? NVIDIA Blog.	https://blogs.nvidia.com/blog/what-is-mlops/
Santos, M. (2023, 30 de mayo). A data scientist's essential guide to exploratory data analysis: Best practices, techniques, and tools to fully understand your data . Medium.	https://medium.com/data-science/a-data-scientists-essential-guide-to-exploratory-data-analysis-25637eee0cf6
Treveil, M., Omont, N., Stenac, C., Lefevre, K., Phan, D., Zentici, J., Lavoillotte, A., Miyazaki, M., & Heidmann, L. (2020). Introducing MLOps . O'Reilly Media.	https://learning.oreilly.com/library/view/introducing-mlops/9781492083283/
Versioning data with DVC (Hands-On tutorial!) . (2020, 30 de septiembre) [Video]. YouTube.	https://www.youtube.com/watch?v=kLKBcPonMYw



GRACIAS

