PARTE 10.

ANÁLISIS DE SENTIMIENTOS DE RESEÑAS DE PELÍCULAS.

Minería de datos.
Cinthia Camila Bravo Marmolejo.
LISC
01/12/2025

Parte 10. Análisis de Sentimientos de Reseñas de Películas.

Cinthia Camila Bravo Marmolejo
Licenciatura en Ingeniería en Sistemas Computacionales
División de Ingenierías Campus Irapuato-Salamanca DICIS
Salamanca, Guanajuato, México
cc.bravomarmolejo@ugto.mx

***Resumen*— En este trabajo se presenta una arquitectura modular para el procesamiento y análisis de información no estructurada procedente de fuentes diversas (texto, documentos PDF, comentarios de usuarios, imágenes y audios). El sistema propuesto realiza: (1) ingestión y normalización de entradas heterogéneas; (2) limpieza y preprocesamiento lingüístico mediante eliminación de stopwords, normalización, tokenización y segmentación; (3) transformación de texto en representaciones numéricas mediante codificación, TF-IDF y embeddings semánticos; (4) aplicación de técnicas de minería de datos para clasificación supervisada o agrupamiento no supervisado; (5) evaluación de desempeño con métricas cuantitativas y visualizaciones (gráficos, nubes de palabras, matrices de confusión); y (6) generación automática de conclusiones y estadísticas relevantes para el dominio.**

Términos clave—TF-IDF, Matriz de confusión, Pipeline, Features y Resultados preliminares.

I. Introducción

En la era de la información, una gran parte de los datos generados por usuarios y organizaciones se presenta en formato no estructurado, especialmente texto. Este proyecto aborda el análisis y la clasificación de información textual mediante técnicas de minería de datos y aprendizaje automático, con el objetivo de transformar datos crudos en conocimiento útil y accionable. El trabajo incluye todas las etapas clásicas: recolección de datos no estructurados, limpieza y preprocesamiento, representación de la información, entrenamiento y evaluación de modelos, y despliegue de una interfaz para la interacción con el sistema.

Motivación: La capacidad de clasificar grandes volúmenes de texto de forma automática facilita tareas como la detección de temas, la clasificación de opiniones, la priorización de incidencias o la extracción de información relevante para la toma de decisiones.

Planteamiento: Dado un conjunto de documentos textuales (comentarios, artículos, mensajes, etc.), desarrollar un sistema que procesa el texto, lo representa mediante técnicas adecuadas (por ejemplo, Bag of Words o TF-IDF), y asigna una etiqueta o categoría mediante modelos de minería de datos. El sistema es evaluado mediante métricas estándar (exactitud, precisión, recuperación, puntuación F1) y proporcionar herramientas visuales (matrices de confusión, gráficas).

Objetivo: El proyecto cubre desde la recolección y procesamiento inicial de textos hasta la evaluación y despliegue de una interfaz básica para uso demostrativo. No se incluyen en el alcance actual tareas avanzadas de despliegue en producción (monitorización en tiempo real, escalado automático) ni la incorporación de modelos de lenguaje a gran escala salvo que se indica explícitamente.

II. Metodología

1. Adquisición de datos desde Kaggle y archivos externos

Para comenzar la construcción del sistema, se seleccionó un dataset público de Kaggle compuesto por reseñas de películas y su respectiva metadata.

- Se ingresó a Kaggle para localizar un conjunto de reseñas con licencia abierta.
- Se descargó un archivo comprimido (.zip) que contenía reseñas en formato .txt, clasificadas en carpetas “positivas” y “negativas”.
- Se descargó también la tabla de metadata en CSV, la cual contiene información del crítico, fecha y fragmentos de reseñas.
- Ambos conjuntos se integraron en un repositorio local del proyecto, manteniendo trazabilidad sobre la fuente y la fecha de adquisición.
- Se generó un archivo maestro que consolidó las fuentes originales para facilitar el análisis posterior.

Esta base de datos sirvió como punto de partida para todo el pipeline posterior. A continuación, se incluyen partes de las dos tablas una es una sección de Kaggle, Tabla I y Tabla II por las notas .txt reales extraídas del .zip y traducidas al español.

TABLA I. Categorías y metadata (origen: Kaggle) [1].

Nombre del crítico	***Persona que publicó***	***Fecha***	***Reseña***
Andrew L. Urban	Urban Cinefile	06/02/2010	Una aventura fantástica que combina la mitología griega

			con lugares y valores contemporáneos de Estados Unidos. Cualquiera de unos 15 años (más o menos un par) se emocionará con el espectáculo visual.
Louise Keller	Urban Cinefile	06/02/2010	Uma Thurman como Medusa, la gorgona con un peinado de serpientes retorcidas y una mirada hipnótica que convierte en piedra, es uno de los puntos más destacados de esta fascinante fantasía.
	FILMINK (Australia)	09/02/2010	Con un elenco de primera y efectos especiales deslumbrantes, esta película mantendrá entretenidos a los adolescentes hasta la próxima entrega de Harry Potter.
Ethan Alter	Hollywood Reporter	10/02/2010	Lo que realmente le falta a El ladrón del rayo es un auténtico sentido de asombro, eso mismo que hace que los espectadores regresen a Hogwarts una y otra vez.
David Germain	Associated Press	10/02/2010	Es más una lista de ingredientes que una poción mágica cinematográfica para disfrutar de principio a fin.
Nick Schager	Slant Magazine	10/02/2010	Las imitaciones de Harry Potter no suelen ser tan evidentes ni tan descuidadas como este intento de iniciar una franquicia, dirigido por Chris Columbus.
Bill Goodykoontz	Arizona Republic	10/02/2010	Percy Jackson no es una gran película, pero sí una buena, que presenta fragmentos de mitología griega como si fueran referencias del canal Disney.
Jordan Hoffman	UGO	10/02/2010	Divertida, ágil e imaginativa.
David Nusair	Reel Film Reviews	06/06/2010	Un tremendo paso adelante para la cineasta Nicole Holofcener...
Connie Ogle	Miami Herald	10/06/2010	Aunque Please Give es una película sobre la conciencia, también es astutamente divertida, con momentos penetrantes de perspicacia y calidez.
Robbie Collin	News of the World	16/06/2010	Bagatelas, trivialidades, fruslerías y frivolidades. Tenemos muchas palabras para describir cosas que no importan. Y ahora, también tenemos una película. Please Give.
Graham Young	Birmingham Post	16/06/2010	Inteligente y conmovedora, es una cita obligada para los amantes del drama humano.

TABLA II. CATEGORÍAS ARCHIVOS .ZIP.

Reseña	***Sentimiento***
Uno de los otros críticos mencionó que después de ver solo un episodio de Oz quedarás enganchado. Tenía razón, eso fue exactamente lo que me pasó. Lo primero que me impactó de Oz fue su brutalidad y sus escenas de violencia sin concesiones, presentes desde el principio. Créeme, este no es un programa para los débiles de corazón ni para los tímidos. No se guarda nada respecto a drogas, sexo o violencia. Es duro, en el sentido más clásico de la palabra. Se llama Oz porque es el apodo de la prisión estatal de máxima seguridad Oswald. Se centra principalmente en Ciudad Esmeralda, una sección experimental de la prisión donde todas las celdas tienen frentes de vidrio y dan hacia el interior, por lo que la privacidad no es prioridad. Allí viven arios, musulmanes, pandilleros, latinos, cristianos, italianos, irlandeses y más… así que los enfrentamientos, las miradas de odio y los acuerdos turbios nunca están lejos. Diría que el mayor atractivo de la serie es que se atreve a llegar donde otras no. Olvida las historias bonitas para el público general, olvida el encanto y el romance… Oz no juega. El primer episodio que vi me pareció tan desagradable que fue surrealista. No estaba preparado, pero mientras seguía viéndola, le tomé gusto a Oz y me acostumbré a los altos niveles de violencia explícita. No solo violencia, sino injusticia (guardias corruptos que se venden por unas monedas, reclusos que matan por órdenes y salen impunes, presos educados que terminan humillados por falta de experiencia callejera). Viendo Oz, uno puede llegar a sentirse cómodo con lo incómodo… si logras conectar con tu lado oscuro.	positivo
Una pequeña producción maravillosa. La técnica de filmación es muy discreta, al estilo de la vieja BBC, lo que da una sensación reconfortante —y a veces inquietante— de realismo en toda la pieza. Los actores fueron escogidos a la perfección: Michael Sheen no solo domina el "polari", sino también todas las voces. Se puede notar la edición impecable guiada por las referencias a los diarios de Williams. No solo vale la pena verla, sino que está magistralmente escrita e interpretada. Una producción maestra sobre uno de los grandes del humor y su vida. El realismo se refuerza con los pequeños detalles: la fantasía del guardia, que en lugar de usar técnicas tradicionales de "sueño" se mantiene sólida y luego desaparece. Juega con nuestro conocimiento y sentidos, especialmente en las escenas entre Orton y Halliwell, y los decorados —en especial el departamento con los murales pintados por Halliwell— están magníficamente logrados.	positivo
Pensé que era una forma maravillosa de pasar un caluroso fin de semana de verano, sentado en el cine con aire acondicionado viendo una comedia ligera. La trama es simple, pero los diálogos son ingeniosos y los personajes encantadores (incluso el sospechoso asesino en serie de buena educación). Algunos pueden decepcionarse al darse cuenta de que no es Match Point 2: Risk Addiction, pero para mí fue una prueba de que Woody Allen sigue dominando el estilo que muchos amamos. Hacía años que no me reía tanto con una comedia suya (¿una década quizá?). Aunque nunca me había impresionado Scarlett Johansson, aquí logra atenuar su imagen "sexy" y convertirse en una joven promedio pero llena de energía. Puede que no sea la joya de su carrera, pero fue más ingeniosa que El diablo viste a la moda y más interesante que Superman: una gran comedia para ver con amigos.	positivo
Básicamente hay una familia donde un niño pequeño (Jake) cree que hay un zombi en su armario y sus padres pelean todo el tiempo.	negativo

La película es más lenta que una telenovela… y de repente Jake decide convertirse en Rambo y matar al zombi. Primero que nada, cuando haces una película debes decidir si es un thriller o un drama. Como drama es pasable: los padres se divorcian y discuten como en la vida real. Pero luego está Jake con su armario, lo cual arruina toda la película. Esperaba ver algo como Boogeyman, y en cambio vi un drama con algunos momentos de suspenso sin sentido. 3 de 10, solo por los buenos actores que hacen de padres y los diálogos decentes. Las escenas con Jake: simplemente ignóralas.	
Love in the Time of Money, de Petter Mattei, es una película visualmente impresionante. Nos ofrece un retrato vívido de las relaciones humanas. Parece mostrarnos lo que el dinero, el poder y el éxito hacen con las personas en distintas situaciones. Basada en una obra de Arthur Schnitzler, el director traslada la acción al Nueva York actual, donde distintos personajes se cruzan y se conectan. Cada uno se enlaza con el siguiente de alguna manera, aunque nadie conoce el punto anterior de conexión. Visualmente, la película tiene un aire sofisticado y lujoso; observamos cómo viven estas personas en su propio hábitat. Lo que se percibe en todos ellos es el distinto grado de soledad que habitan. Una gran ciudad no es precisamente el mejor lugar para encontrar relaciones sinceras, como queda claro en la mayoría de los casos que vemos. Las actuaciones son buenas bajo la dirección de Mattei: Steve Buscemi, Rosario Dawson, Carol Kane, Michael Imperioli, Adrian Grenier y el resto del elenco dan vida a los personajes. Le deseamos buena suerte al Sr. Mattei y esperamos con ansias su próxima obra.	positivo
Probablemente mi película favorita de todos los tiempos, una historia de entrega, sacrificio y dedicación a una causa noble, pero sin ser moralista ni aburrida. Nunca envejece, aunque la haya visto unas 15 veces en los últimos 25 años. La actuación de Paul Lukas me conmueve hasta las lágrimas, y Bette Davis, en uno de sus pocos papeles realmente simpáticos, es un deleite. Los niños son, como dice la abuela, más "enanos disfrazados" que niños, pero eso los hace aún más divertidos de ver. Y el lento despertar de la madre ante lo que sucede en el mundo y en su propia casa es creíble y sorprendente. Si tuviera una docena de pulgares, todos estarían hacia arriba para esta película.	positivo
Me encantaría ver una nueva versión moderna de Sea Hunt con la tecnología actual; despertaría el niño emocionado que hay en mí. Crecí viendo televisión en blanco y negro y Sea Hunt junto con Gunsmoke eran mis héroes cada semana. Tienen mi voto para un regreso. Necesitamos un cambio de ritmo en la televisión y esto funcionaría: un mundo de aventuras bajo el agua. Ah, y gracias por este espacio donde se pueden compartir tantos puntos de vista sobre TV y cine. En fin, creo que ya dije lo que quería decir. Sería genial leer más comentarios positivos sobre Sea Hunt.	positivo

2. Preprocesamiento y limpieza de datos

Una vez recopilada la información, se diseñó un pipeline de limpieza reproducible con el fin de transformar el texto crudo en datos aptos para minería de texto. El flujo implementado incluyó:

- Eliminación de ruido: HTML, saltos, etiquetas, URLs, caracteres especiales.
- Normalización:
  - conversión a minúsculas
  - normalización Unicode
  - corrección de tildes y signos
- Tokenización palabra por palabra.
- Eliminación de stopwords en español e inglés.
- Lematización o stemming (según el idioma de la reseña).
- Verificación de duplicados.
- Integración de los resultados en un CSV limpio.

Ejemplos del proceso se documentaron y se muestran en las Tablas III y IV del reporte, donde se comparan tokens originales vs. tokens limpiados para reseñas tanto de Kaggle como del .zip.

El objetivo de esta fase fue producir una versión estandarizada y libre de ruido del dataset original, necesaria para las técnicas de representación numérica. A continuación, se muestran todas las reseñas procesadas de la tabla de Kaggle.

TABLA III. TOKENIZACIÓN DE LA TABLA I.

Texto original (reseña)	***Tokens***	***Tokens filtrados (limpios)***
Una aventura fantástica que combina la mitología griega con lugares y valores contemporáneos de Estados Unidos. Cualquiera de unos 15 años (más o menos un par) se emocionará con el espectáculo visual.	Una, aventura, fantástica, que, combina, la, mitología, griega, con, lugares, y, valores, contemporáneos, de, Estados, Unidos, Cualquiera, de, unos, 15, años, más, o, menos, un, par, se, emocionará, con, el, espectáculo, visual	aventura, fantástica, combina, mitología, griega, lugares, valores, contemporáneos, estados, unidos, emocionará, espectáculo, visual
Uma Thurman como Medusa, la gorgona con un peinado de serpientes retorcidas y una mirada hipnótica que convierte en piedra, es uno de los puntos más destacados de esta fascinante fantasía.	Uma, Thurman, como, Medusa, la, gorgona, con, un, peinado, de, serpientes, retorcidas, y, una, mirada, hipnótica, que, convierte, en, piedra, es, uno, de, los, puntos, más, destacados, de, esta, fascinante, fantasía	uma, thurman, medusa, gorgona, peinado, serpientes, retorcidas, mirada, hipnótica, convierte, piedra, puntos, destacados, fascinante, fantasía
Con un elenco de primera y efectos especiales deslumbrantes, esta película mantendrá entretenidos a los adolescentes hasta la próxima entrega de Harry Potter.	Con, un, elenco, de, primera, y, efectos, especiales, deslumbrantes, esta, película, mantendrá, entretenidos, a, los, adolescentes, hasta, la, próxima, entrega, de, Harry, Potter	elenco, primera, efectos, especiales, deslumbrantes, película, mantendrá, entretenidos, adolescentes, próxima, entrega, harry, potter
Lo que realmente le falta a El ladrón del rayo es un auténtico sentido de asombro,	Lo, que, realmente, le, falta, a, El, ladrón, del, rayo, es, un, auténtico, sentido, de, asombro,	realmente, falta, ladrón, rayo, auténtico, sentido, asombro, mismo,

eso mismo que hace que los espectadores regresen a Hogwarts una y otra vez.	eso, mismo, que, hace, que, los, espectadores, regresen, a, Hogwarts, una, y, otra, vez	hace, espectadores, regresen, hogwarts, vez
Es más una lista de ingredientes que una poción mágica cinematográfica para disfrutar de principio a fin.	Es, más, una, lista, de, ingredientes, que, una, poción, mágica, cinematográfica, para, disfrutar, de, principio, a, fin	lista, ingredientes, poción, mágica, cinematográfica, disfrutar, principio, fin
Las imitaciones de Harry Potter no suelen ser tan evidentes ni tan descuidadas como este intento de iniciar una franquicia, dirigido por Chris Columbus.	Las, imitaciones, de, Harry, Potter, no, suelen, ser, tan, evidentes, ni, tan, descuidadas, como, este, intento, de, iniciar, una, franquicia, dirigido, por, Chris, Columbus	imitaciones, harry, potter, evidentes, descuidadas, intento, iniciar, franquicia, dirigido, chris, columbus
Percy Jackson no es una gran película, pero sí una buena, que presenta fragmentos de mitología griega como si fueran referencias del canal Disney.	Percy, Jackson, no, es, una, gran, película, pero, sí, una, buena, que, presenta, fragmentos, de, mitología, griega, como, si, fueran, referencias, del, canal, Disney	percy, jackson, gran, película, buena, presenta, fragmentos, mitología, griega, referencias, canal, disney
Divertida, ágil e imaginativa.	Divertida, ágil, e, imaginativa	divertida, ágil, imaginativa
Un tremendo paso adelante para la cineasta Nicole Holofcener...	Un, tremendo, paso, adelante, para, la, cineasta, Nicole, Holofcener	tremendo, paso, adelante, cineasta, nicole, holofcener
Aunque Please Give es una película sobre la conciencia, también es astutamente divertida, con momentos penetrantes de perspicacia y calidez.	Aunque, Please, Give, es, una, película, sobre, la, conciencia, también, es, astutamente, divertida, con, momentos, penetrantes, de, perspicacia, y, calidez	please, give, película, conciencia, astutamente, divertida, momentos, penetrantes, perspicacia, calidez
Bagatelas, trivialidades, fruslerías y frivolidades. Tenemos muchas palabras para describir cosas que no importan. Y ahora, también tenemos una película. Please Give.	Bagatelas, trivialidades, fruslerías, y, frivolidades, Tenemos, muchas, palabras, para, describir, cosas, que, no, importan, Y, ahora, también, tenemos, una, película, Please, Give	bagatelas, trivialidades, fruslerías, frivolidades, palabras, describir, cosas, importan, película, please, give
Inteligente y conmovedora, es una cita obligada para los amantes del drama humano.	Inteligente, y, conmovedora, es, una, cita, obligada, para, los, amantes, del, drama, humano	inteligente, conmovedora, cita, obligada, amantes, drama, humano

Y la tabla de tokenización de las reseñas en el .zip:

TABLA IV. TOKENIZACIÓN DE LA TABLA II.

Reseña	***Tokens***	***Tokens filtrados***
Uno de los otros críticos mencionó que después de ver solo un episodio de Oz quedarás enganchado. Tenía razón, eso fue exactamente lo que me pasó. Lo primero que me impactó de Oz fue su brutalidad y sus escenas de violencia sin concesiones, presentes desde el principio. Créeme, este no es un programa para los débiles de corazón ni para los tímidos. No se guarda nada respecto a drogas, sexo o violencia. Es duro, en el sentido más clásico de la palabra. Se llama Oz porque es el apodo de la prisión...	uno, de, los, otros, críticos, mencionó, que, después, de, ver, solo, un, episodio, de, oz, quedarás, enganchado, tenía, razón, eso, fue, exactamente, lo, que, me, pasó, lo, primero, que, me, impactó, de, oz, fue, su, brutalidad, y, sus, escenas, de...	otros, críticos, mencionó, después, ver, solo, episodio, oz, quedarás, enganchado, tenía, razón, eso, exactamente, pasó, primero, impactó, oz, brutalidad, escenas, violencia, concesiones, presentes, desde, principio, créeme, programa, débiles, corazón, ni, tímidos, guarda, nada, respecto, drogas, sexo, o, violencia, duro, sentido...
Una pequeña producción maravillosa. La técnica de filmación es muy discreta, al estilo de la vieja BBC, lo que da una sensación reconfortante —y a veces inquietante— de realismo en toda la pieza. Los actores fueron escogidos a la perfección: Michael Sheen no solo domina el "polari", sino también todas las voces. Se puede notar la edición impecable guiada por las referencias a los diarios de Williams. No solo vale la pena verla, sino que está magistralmente escrita e interpretada. Una producción ...	una, pequeña, producción, maravillosa, la, técnica, de, filmación, es, muy, discreta, al, estilo, de, la, vieja, bbc, lo, que, da, una, sensación, reconfortante, y, a, veces, inquietante, de, realismo, en, toda, la, pieza, los, actores, fueron, escogidos, a, la, perfección...	pequeña, producción, maravillosa, técnica, filmación, discreta, estilo, vieja, bbc, da, sensación, reconfortante, veces, inquietante, realismo, toda, pieza, actores, fueron, escogidos, perfección, michael, sheen, solo, domina, polari, sino, todas, voces, puede, notar, edición, impecable, guiada, referencias, diarios, williams, solo, vale, pena...
Pensé que era una forma maravillosa de pasar un caluroso fin de semana de verano, sentado en el cine con aire acondicionado viendo una comedia ligera. La trama es simple, pero los diálogos son ingeniosos y los personajes encantadores (incluso el sospechoso asesino en serie de buena educación). Algunos pueden decepcionarse al darse cuenta de que no es Match Point 2: Risk Addiction, pero para mí fue una prueba de que Woody Allen sigue dominando el estilo que muchos amamos. Hacía años que no me reí...	pensé, que, era, una, forma, maravillosa, de, pasar, un, caluroso, fin, de, semana, de, verano, sentado, en, el, cine, con, aire, acondicionado, viendo, una, comedia, ligera, la, trama, es, simple, pero, los, diálogos, son, ingeniosos, y, los, personajes, encantadores, incluso...	pensé, era, forma, maravillosa, pasar, caluroso, fin, semana, verano, sentado, cine, aire, acondicionado, viendo, comedia, ligera, trama, simple, diálogos, son, ingeniosos, personajes, encantadores, incluso, sospechoso, asesino, serie, buena, educación, algunos, pueden, decepcionarse, darse, cuenta, match, point, 2, risk, addiction, mí...
Básicamente hay una familia donde un niño pequeño (Jake) cree que hay un zombi en su armario y sus padres pelean todo el tiempo. La película es más lenta que una telenovela… y de repente Jake decide convertirse en Rambo y matar al zombi. Primero que nada, cuando haces una película debes decidir si es un thriller o un drama. Como drama es pasable: los padres se divorcian y discuten como en la vida real. Pero luego está Jake con su armario, lo cual arruina toda la película. Esperaba ver algo como ...	básicamente, hay, una, familia, donde, un, niño, pequeño, jake, cree, que, hay, un, zombi, en, su, armario, y, sus, padres, pelean, todo, el, tiempo, la, película, es, más, lenta, que, una, telenovela, y, de, repente, jake, decide, convertirse, en, rambo...	básicamente, hay, familia, niño, pequeño, jake, cree, hay, zombi, armario, padres, pelean, todo, tiempo, película, lenta, telenovela, repente, jake, decide, convertirse, rambo, matar, zombi, primero, nada, haces, película, debes, decidir, thriller, o, drama, drama, pasable, padres, divorcian, discuten, vida, real...

Love in the Time of Money, de Petter Mattei, es una película visualmente impresionante. Nos ofrece un retrato vívido de las relaciones humanas. Parece mostrarnos lo que el dinero, el poder y el éxito hacen con las personas en distintas situaciones. Basada en una obra de Arthur Schnitzler, el director traslada la acción al Nueva York actual, donde distintos personajes se cruzan y se conectan. Cada uno se enlaza con el siguiente de alguna manera, aunque nadie conoce el punto anterior de conexión. ...	love, in, the, time, of, money, de, petter, mattei, es, una, película, visualmente, impresionante, nos, ofrece, un, retrato, vívido, de, las, relaciones, humanas, parece, mostrarnos, lo, que, el, dinero, el, poder, y, el, éxito, hacen, con, las, personas, en, distintas...	love, in, the, time, of, money, petter, mattei, película, visualmente, impresionante, nos, ofrece, retrato, vívido, relaciones, humanas, parece, mostrarnos, dinero, poder, éxito, hacen, personas, distintas, situaciones, basada, obra, arthur, schnitzler, director, traslada, acción, nueva, york, actual, distintos, personajes, cruzan, conectan...
Probablemente mi película favorita de todos los tiempos, una historia de entrega, sacrificio y dedicación a una causa noble, pero sin ser moralista ni aburrida. Nunca envejece, aunque la haya visto unas 15 veces en los últimos 25 años. La actuación de Paul Lukas me conmueve hasta las lágrimas, y Bette Davis, en uno de sus pocos papeles realmente simpáticos, es un deleite. Los niños son, como dice la abuela, más “enanos disfrazados” que niños, pero eso los hace aún más divertidos de ver. Y el len...	probablemente, mi, película, favorita, de, todos, los, tiempos, una, historia, de, entrega, sacrificio, y, dedicación, a, una, causa, noble, pero, sin, ser, moralista, ni, aburrida, nunca, envejece, aunque, la, haya, visto, unas, 15, veces, en, los, últimos, 25, años, la...	probablemente, mi, película, favorita, todos, tiempos, historia, entrega, sacrificio, dedicación, causa, noble, ser, moralista, ni, aburrida, nunca, envejece, aunque, haya, visto, unas, 15, veces, últimos, 25, años, actuación, paul, lukas, conmueve, lágrimas, bette, davis, pocos, papeles, realmente, simpáticos, deleite, niños...
Me encantaría ver una nueva versión moderna de Sea Hunt con la tecnología actual; despertaría el niño emocionado que hay en mí. Crecí viendo televisión en blanco y negro y Sea Hunt junto con Gunsmoke eran mis héroes cada semana. Tienen mi voto para un regreso. Necesitamos un cambio de ritmo en la televisión y esto funcionaría: un mundo de aventuras bajo el agua. Ah, y gracias por este espacio donde se pueden compartir tantos puntos de vista sobre TV y cine. En fin, creo que ya dije lo que quería...	me, encantaría, ver, una, nueva, versión, moderna, de, sea, hunt, con, la, tecnología, actual, despertaría, el, niño, emocionado, que, hay, en, mí, crecí, viendo, televisión, en, blanco, y, negro, y, sea, hunt, junto, con, gunsmoke, eran, mis, héroes, cada, semana...	encantaría, ver, nueva, versión, moderna, sea, hunt, tecnología, actual, despertaría, niño, emocionado, hay, mí, crecí, viendo, televisión, blanco, negro, sea, hunt, junto, gunsmoke, eran, mis, héroes, cada, semana, tienen, mi, voto, regreso, necesitamos, cambio, ritmo, televisión, esto, funcionaría, mundo, aventuras...

3. Representación: Codificación, TF, DF y TF-IDF

Con el texto limpio, se aplicaron diferentes técnicas de representación:

- Conteo de términos (TF): frecuencia absoluta de cada palabra por documento.
- Document Frequency (DF): número de documentos en los que aparece cada token.
- TF-IDF: se entrenó un vectorizador para transformar cada reseña en un vector numérico de alta dimensión.
- Reducción dimensional (TruncatedSVD) para visualizaciones 2D más interpretables.

Esta fase convierte el lenguaje natural en datos cuantificables, permitiendo entrenar modelos y generar visualizaciones como nubes de palabras o términos más relevantes por clase.

4. Modelado, clustering y análisis de polaridad

Una vez generados los vectores, se entrenó un conjunto de modelos de minería de datos:

- Modelos supervisados:
  - Naive Bayes
  - Linear SVM
  - Decision Tree
- Modelos no supervisados:
  - K-Means (para agrupar reseñas)
  - LDA (para extraer tópicos dominantes)
- Análisis de polaridad con VADER, asignando etiquetas positiva, negativa o neutra para apoyar el análisis cuantitativo.
- Se calcularon métricas de evaluación:
  - accuracy
  - precision
  - recall
  - F1-score
  - matrices de confusión
  - curvas ROC/PR

5. Desarrollo de la aplicación web

Como parte final del proyecto general, se implementó una página web que integra todos los componentes anteriores.

5.1 Arquitectura general

La aplicación se diseñó con:

- Un frontend que permite visualizar reseñas, modificar textos y añadir nuevas entradas.
- Un backend/servidor que procesa solicitudes HTTP (GET, POST, PUT, DELETE).
- Persistencia en:
  - un archivo JSON con las reseñas activas
  - una base de datos SQLite
  - fallback en localStorage cuando el usuario trabaja offline

5.2 Funcionalidades principales

- Ingesta de reseñas nuevas desde la interfaz.
- Actualización automática del archivo JSON.
- Sincronización con la base de datos.
- Preprocesamiento en tiempo real para nuevos textos.
- Regeneración del análisis cuando cambia el contenido.
- Visualizaciones generadas con matplotlib:
  - distribución de polaridad
  - top términos TF-IDF
  - clusters
  - boxplots por película

5.3 Generación de reporte en PDF

El servidor genera un PDF final mediante matplotlib y PdfPages, el cual contiene:

- todas las visualizaciones
- métricas
- gráficas comparativas
- conclusiones preliminares

Cada vez que el usuario modifica una reseña desde la página, el sistema:

- actualiza JSON y BD
- reejecuta el pipeline
- regenera el PDF
- habilita su descarga en la interfaz

III. DESARROLLO Y RESULTADOS

La arquitectura final permite transformar texto crudo en información accionable y visual, demostrando la utilidad del análisis de sentimientos y la minería de datos en un entorno real.

Se muestra el código de grafo.py que es el que efectúa todo en el pdf:

```
    import os
    import sqlite3
    import re
    import warnings
    import unicodedata
    from collections import OrderedDict, Counter
    from pathlib import Path

    import matplotlib.pyplot as plt
    import numpy as np
    import pandas as pd
    import seaborn as sns

    import nltk
    from nltk.stem.snowball import SnowballStemmer
    from nltk.sentiment.vader import SentimentIntensityAnalyzer

    from sklearn.feature_extraction.text import TfidfVectorizer
    from   sklearn.decomposition   import   LatentDirichletAllocation,
TruncatedSVD
    from sklearn.cluster import KMeans
    from sklearn.model_selection import train_test_split
    from sklearn.svm import LinearSVC
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.tree import DecisionTreeClassifier, plot_tree
    from sklearn import metrics
    from        sklearn.metrics        import        (confusion_matrix,
ConfusionMatrixDisplay, roc_curve,
                                 auc,           precision_recall_curve,
classification_report)

    from matplotlib.backends.backend_pdf import PdfPages

    warnings.filterwarnings("ignore", category=FutureWarning)
    sns.set(style="whitegrid", rc={"figure.figsize": (8, 6)})

    def ensure_nltk_resources():
        resources = ['punkt', 'stopwords', 'vader_lexicon']
        for res in resources:
            try:
                if res == 'punkt':
                    nltk.data.find('tokenizers/punkt')
                elif res == 'stopwords':
                    nltk.data.find('corpora/stopwords')
                elif res == 'vader_lexicon':
                    nltk.data.find('sentiment/vader_lexicon')
            except LookupError:
                try:
                    nltk.download(res)
                except Exception:
                    pass

    stemmer = SnowballStemmer("english")

    def tokenization_and_stemming(text, stopwords_set):
        if not isinstance(text, str):
            text = str(text or "")
        tokens = []
        try:
            for word in nltk.word_tokenize(text):
                w = word.lower()
                if w not in stopwords_set:
                    tokens.append(w)
        except Exception:
            tokens = re.findall(r"[A-Za-z]+", text.lower())
            tokens = [t for t in tokens if t not in stopwords_set]
        filtered = [t for t in tokens if t.isalpha()]
        stems  =  [stemmer.stem(t)  for  t  in  filtered  if  t  not  in
stopwords_set]
        return stems

    def _normalize_title(s):
        if s is None:
```

```
            return ""
        s = str(s).strip().lower()
        s = unicodedata.normalize("NFKD", s)
        s = "".join(ch for ch in s if not unicodedata.combining(ch))
        s = s.split('(')[0].split('-')[0].strip()
        s = s.strip(" .,:;-—")
        return s

    def save_figure_to_pdf(pp, fig, tight=True):
        """
        Save fig into PdfPages. Use bbox_inches='tight' by default to
avoid cropping
        and ensure the figure is as large as it needs to be on the page.
        """
        try:
            if tight:
                pp.savefig(fig,      bbox_inches='tight')            #
\x1b[33mMODIFIED\x1b[0m >>> linea modificada: usé bbox_inches='tight'
            else:
                pp.savefig(fig)
        finally:
            plt.close(fig)

    def plot_top_movies_diagram(movies_df=None, top_n=10):
        try:
            if movies_df is None or movies_df.empty:
                return None

            rows = []
            for idx, row in movies_df.iterrows():
                title        =       row.get('Series_Title')        or
row.get('series_title') or row.get('Series Title') or f"Untitled {idx}"
                ratings = []
                ur = row.get('user_reviews') or []
                if isinstance(ur, list):
                    for u in ur:
                        try:
                            r = u.get('rating') if isinstance(u, dict)
else None
                            if r is None:
                                continue
                            ratings.append(float(r))
                        except Exception:
                            continue
                rr = row.get('reviews') or []
                if isinstance(rr, list):
                    for u in rr:
                        try:
                            r = u.get('rating') if isinstance(u, dict)
else None
                            if r is None:
                                continue
                            ratings.append(float(r))
                        except Exception:
                            continue
                for key in ('positive_review','negative_review'):
                    pr = row.get(key)
                    if isinstance(pr, dict):
                        try:
                            r = pr.get('rating')
                            if r is not None:
                                ratings.append(float(r))
                        except Exception:
                            pass
                if ratings:
                    avg = float(sum(ratings)) / len(ratings)
                    rows.append((title, avg))
            if not rows:
                return None

            df = pd.DataFrame(rows, columns=['title','avg_rating'])
            df              =              df.sort_values('avg_rating',
ascending=False).reset_index(drop=True)
            df_top = df.head(top_n).copy()
            n = len(df_top)
            fig, ax = plt.subplots(figsize=(8.27, 11.69))
            y_pos = np.arange(n)
            ax.barh(y_pos,    df_top['avg_rating'],    align='center',
color='tab:blue', alpha=0.85)
            ax.set_yticks(y_pos)
            ax.set_yticklabels(df_top['title'].tolist(), fontsize=9)
            ax.set_xlabel("Rating (promedio de reseñas)")
            ax.set_xlim(0, 10)
            ax.set_title(f"Top {n} películas (rating por promedio de
reseñas)")
            ax.invert_yaxis()
            plt.tight_layout()
            return fig
        except Exception as e:
            print("Error generating top movies diagram:", e)
            return None

    def plot_confusion_matrix(y_true, y_pred, labels, title=None):
        fig, ax = plt.subplots(figsize=(6,5))
        cm = confusion_matrix(y_true, y_pred, labels=labels)
        disp       =       ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels)
        disp.plot(cmap='Blues', ax=ax, colorbar=False)
        ax.set_title(title or "Confusion Matrix")
        return fig

    def plot_roc_curves(models_scores, y_true, title="ROC Curves"):
        fig, ax = plt.subplots(figsize=(7,6))
        for name, score_info in models_scores.items():
            y_score = score_info.get('score')
            if y_score is None or len(y_score) != len(y_true):
                continue
            fpr, tpr, _ = roc_curve(y_true, y_score)
            roc_auc = auc(fpr, tpr)
            ax.plot(fpr, tpr, label=f"{name} (AUC={roc_auc:.3f})")
        ax.plot([0,1],[0,1],'k--', lw=0.7)
        ax.set_xlabel("False Positive Rate")
        ax.set_ylabel("True Positive Rate")
        ax.set_title(title)
        ax.legend(loc='lower right')
        return fig

    def plot_precision_recall(models_scores, y_true, title="Precision-
Recall Curves"):
        fig, ax = plt.subplots(figsize=(7,6))
        for name, score_info in models_scores.items():
            y_score = score_info.get('score')
            if y_score is None or len(y_score) != len(y_true):
                continue
            precision,   recall,   _   =   precision_recall_curve(y_true,
y_score)
            ax.plot(recall, precision, label=name)
        ax.set_xlabel("Recall")
        ax.set_ylabel("Precision")
        ax.set_title(title)
        ax.legend(loc='best')
        return fig

    def plot_metric_bars(metrics_dict, title="Model comparison"):
        """
        Create a bar chart of metrics. The figure size is chosen
dynamically
        depending on the number of models, so the plot uses the space
required.
        """
        if not metrics_dict:
            return None
        df = pd.DataFrame(metrics_dict).T
        keep = [c for c in ['accuracy','precision','recall','f1'] if c
in df.columns]
        if not keep:
            return None
        # dynamic sizing: width scales with number of models
        n_models = max(1, df.shape[0])
        width = max(8, n_models * 1.5)  # \x1b[33mMODIFIED\x1b[0m >>>
linea modificada: tamaño dinámico según # de modelos
        fig, ax = plt.subplots(figsize=(width, 6))
        df[keep].plot(kind='bar', ax=ax)
        ax.set_title(title)
        ax.set_ylim(0,1)
        ax.legend(loc='lower right')
        plt.tight_layout()
        return fig

    def compute_sentiment_labels(df_reviews):
        if df_reviews is None or df_reviews.empty:
            return df_reviews.copy(), None
        sia = SentimentIntensityAnalyzer()
        df = df_reviews.copy()
        compounds = []
        labels = []
        for txt in df['review'].astype(str).tolist():
            try:
                sc = sia.polarity_scores(txt).get('compound', 0.0)
            except Exception:
                sc = 0.0
            compounds.append(float(sc))
            if sc >= 0.05:
                labels.append('positive')
            elif sc <= -0.05:
                labels.append('negative')
            else:
                labels.append('neutral')
        df['sentiment_compound'] = compounds
        df['sentiment'] = labels
        return df, Counter(labels)

    def plot_sentiment_distribution_overall(df_reviews_with_sentiment):
        if       df_reviews_with_sentiment       is       None       or
df_reviews_with_sentiment.empty:
            return None
        counts                                                        =
df_reviews_with_sentiment['sentiment'].value_counts().reindex(['positi
ve','neutral','negative']).fillna(0)
        total = counts.sum()
        fig, ax = plt.subplots(figsize=(8,4))
        sns.barplot(x=counts.index,                    y=counts.values,
palette=['#2ca02c','#7f7f7f','#d62728'], ax=ax)
        ax.set_title("Distribución de polaridad (reseñas JSON)")
```

```
        ax.set_ylabel("Número de reseñas")
        for i, v in enumerate(counts.values):
            ax.text(i, v + max(1, total*0.01), f"{int(v)}", ha='center')
        plt.tight_layout()
        return fig

    def plot_sentiment_pie(df_reviews_with_sentiment):
        if df_reviews_with_sentiment is None or df_reviews_with_sentiment.empty:
            return None
        counts = df_reviews_with_sentiment['sentiment'].value_counts().reindex(['positive','neutral','negative']).fillna(0)
        labels = [f"{lab} ({int(cnt)})" for lab,cnt in zip(counts.index, counts.values)]
        fig, ax = plt.subplots(figsize=(6,6))
        ax.pie(counts.values, labels=labels, colors=['#2ca02c','#7f7f7f','#d62728'], autopct='%1.1f%%', startangle=140)
        ax.set_title("Porcentaje de polaridad (reseñas JSON)")
        plt.tight_layout()
        return fig

    def plot_sentiment_by_top_movies(movies_df, df_reviews_with_sentiment, top_n=10):
        if movies_df is None or movies_df.empty or df_reviews_with_sentiment is None or df_reviews_with_sentiment.empty:
            return None
        grp = df_reviews_with_sentiment.groupby('movie_idx')['sentiment'].value_counts().unstack(fill_value=0)
        grp['total'] = grp.sum(axis=1)
        top_idx = grp.sort_values('total', ascending=False).head(top_n).index.tolist()
        if not top_idx:
            return None
        sub = grp.loc[top_idx, ['positive','neutral','negative']].fillna(0)
        titles = []
        for mi in sub.index:
            try:
                t = movies_df.iloc[int(mi)].get('Series_Title') or movies_df.iloc[int(mi)].get('series_title') or ""
            except Exception:
                t = str(mi)
            titles.append(t)
        sub.index = titles
        fig, ax = plt.subplots(figsize=(9, max(4, 0.6*len(titles))))
        sub[['positive','neutral','negative']].plot(kind='barh', stacked=True, color=['#2ca02c','#7f7f7f','#d62728'], ax=ax)
        ax.set_xlabel("Número de reseñas")
        ax.set_ylabel("Película")
        ax.set_title(f"Polaridad por película (top {len(titles)} por #reseñas en JSON)")
        plt.tight_layout()
        return fig

    def plot_rating_boxplots_overall_and_by_sentiment(df_reviews_with_sentiment):
        """
        Two-panel figure:
         - Left: boxplot of numeric ratings overall
         - Right: boxplot of numeric ratings grouped by sentiment (positive/neutral/negative)
        Requires df_reviews_with_sentiment to contain numeric 'rating' and 'sentiment'.
        """
        if df_reviews_with_sentiment is None or df_reviews_with_sentiment.empty:
            return None
        # Filter numeric ratings
        df = df_reviews_with_sentiment.copy()
        df['rating_num'] = pd.to_numeric(df['rating'], errors='coerce')
        df = df[~df['rating_num'].isna()].copy()
        if df.empty:
            return None

        # order sentiments for consistent coloring
        order = ['positive', 'neutral', 'negative']
        palette = {'positive': '#2ca02c', 'neutral': '#7f7f7f', 'negative': '#d62728'}

        fig, axes = plt.subplots(ncols=2, figsize=(12, 6))
        # Overall boxplot
        sns.boxplot(x=df['rating_num'], ax=axes[0], color='tab:blue')
        axes[0].set_xlabel("Rating")
        axes[0].set_title("Distribución de calificaciones (boxplot) - global")
        axes[0].set_xlim(0, 10)

        # By sentiment
        sns.boxplot(x='sentiment', y='rating_num', data=df, order=order, palette=[palette.get(o) for o in order], ax=axes[1])
        axes[1].set_xlabel("Polaridad")
        axes[1].set_ylabel("Rating")
        axes[1].set_title("Distribución de calificaciones por polaridad (boxplot)")
        axes[1].set_ylim(0, 10)

        plt.tight_layout()
        return fig

    def plot_rating_boxplot_by_top_movies(movies_df, df_reviews_with_sentiment, top_n=8):
        """
        Boxplot of ratings for the top-N movies by number of JSON reviews.
        Each box shows the distribution of numeric ratings for that movie.
        """
        if movies_df is None or movies_df.empty or df_reviews_with_sentiment is None or df_reviews_with_sentiment.empty:
            return None
        df = df_reviews_with_sentiment.copy()
        df['rating_num'] = pd.to_numeric(df['rating'], errors='coerce')
        df = df[~df['rating_num'].isna()].copy()
        if df.empty:
            return None

        counts = df.groupby('movie_idx').size().sort_values(ascending=False)
        top_idx = counts.head(top_n).index.tolist()
        if not top_idx:
            return None

        # build plot DataFrame with movie titles
        rows = []
        titles = []
        for mi in top_idx:
            try:
                title = movies_df.iloc[int(mi)].get('Series_Title') or movies_df.iloc[int(mi)].get('series_title') or ""
            except Exception:
                title = str(mi)
            sub = df[df['movie_idx'] == mi]
            if sub.empty:
                continue
            for v in sub['rating_num'].tolist():
                rows.append({'movie': title, 'rating': v})
            titles.append(title)

        if not rows:
            return None
        df_plot = pd.DataFrame(rows)
        # order movies by median descending so highest median on top
        medians = df_plot.groupby('movie')['rating'].median().sort_values(ascending=False)
        ordered = medians.index.tolist()

        fig, ax = plt.subplots(figsize=(9, max(4, 0.6*len(ordered))))
        sns.boxplot(x='rating', y='movie', data=df_plot, order=ordered, ax=ax, palette='Blues')
        ax.set_xlabel("Rating")
        ax.set_ylabel("Película")
        ax.set_title(f"Distribución de calificaciones por película (top {len(ordered)} por #reseñas)")
        ax.set_xlim(0, 10)
        plt.tight_layout()
        return fig

    def main(sqlite_db_path='./IMDB_Movies_2021.db', pdf_out='./site_data/model_results.pdf'):
        ensure_nltk_resources()
        pdf_out = Path(pdf_out)
        pdf_out.parent.mkdir(parents=True, exist_ok=True)

        # We will write to a temporary file and then atomically replace the final PDF
        pdf_tmp = pdf_out.with_name(pdf_out.name + '.tmp')

        # 1) TRAINING: read DB and prepare df_model (first 4000) - unchanged
        conn = None
        try:
            conn = sqlite3.connect(sqlite_db_path)
            try:
                df_all = pd.read_sql_query("SELECT ID, AUTHOR, TITLE, REVIEW, RATING, POSTED FROM REVIEWS", conn)
            except Exception:
                df_all = pd.read_sql_query("SELECT * FROM REVIEWS", conn)
        except Exception:
            df_all = pd.DataFrame(columns=['ID','AUTHOR','TITLE','REVIEW','RATING','POSTED'])
        finally:
            if conn:
                conn.close()

        if not df_all.empty:
            df_all = df_all.replace('\\n', ' ', regex=True)
```

```
            df_model = df_all.loc[:3999].copy()
            df_model.reset_index(drop=True, inplace=True)
            if 'REVIEW' in df_model.columns:
                df_model['REVIEW'] = df_model['REVIEW'].astype(str)
            else:
                df_model['REVIEW'] = ""
            df_model.drop_duplicates(subset="REVIEW", inplace=True)
            df_model.reset_index(drop=True, inplace=True)
        else:
            df_model                                                =
pd.DataFrame(columns=['ID','AUTHOR','TITLE','REVIEW','RATING','POSTED'
])

        #  2)  Load  movies_with_reviews.json  (preferred)  and  build
df_json_reviews
        movies_df = None
        try:
            pref = Path("site_data/movies_with_reviews.json")
            default = Path("site_data/movies.json")
            if pref.exists():
                movies_df = pd.read_json(pref, orient='records')
            elif default.exists():
                movies_df = pd.read_json(default, orient='records')
            else:
                movies_df = pd.DataFrame()
        except Exception:
            movies_df = pd.DataFrame()

        df_json_reviews                                             =
pd.DataFrame(columns=['title','review','rating','movie_idx'])
        try:
            rows = []
            if movies_df is not None and not movies_df.empty:
                for idx, r in movies_df.iterrows():
                    title       =       r.get('Series_Title')       or
r.get('series_title') or r.get('Series Title') or ""
                    # user_reviews
                    ur = r.get('user_reviews') or []
                    if isinstance(ur, list):
                        for u in ur:
                            try:
                                txt = u.get('review') if isinstance(u,
dict) else None
                                rt = u.get('rating') if isinstance(u,
dict) else None
                                if txt:
                                    rows.append({'title':       title,
'review': str(txt), 'rating': (float(rt) if rt is not None else None),
'movie_idx': idx})
                            except Exception:
                                continue
                    # reviews
                    rr = r.get('reviews') or []
                    if isinstance(rr, list):
                        for u in rr:
                            try:
                                txt = u.get('review') if isinstance(u,
dict) else None
                                rt = u.get('rating') if isinstance(u,
dict) else None
                                if txt:
                                    rows.append({'title':       title,
'review': str(txt), 'rating': (float(rt) if rt is not None else None),
'movie_idx': idx})
                            except Exception:
                                continue
                    # positive/negative
                    for key in ('positive_review','negative_review'):
                        pr = r.get(key)
                        if isinstance(pr, dict):
                            try:
                                txt     =     pr.get('review')     or
pr.get('REVIEW')
                                rt = pr.get('rating')
                                if txt:
                                    rows.append({'title':       title,
'review': str(txt), 'rating': (float(rt) if rt is not None else None),
'movie_idx': idx})
                            except Exception:
                                pass
            if rows:
                df_json_reviews = pd.DataFrame(rows)
        except Exception:
            df_json_reviews                                         =
pd.DataFrame(columns=['title','review','rating','movie_idx'])

        stopwords = []
        try:
            stopwords = nltk.corpus.stopwords.words('english')
        except Exception:
            stopwords = []
        stopwords += ["'s","'m","n't","br","movie","film"]
        stopwords_set = set(stopwords)

        preprocessed_model = []
        for doc in (df_model['REVIEW'].tolist() if not df_model.empty
else []):
            preprocessed_model.append("
".join(tokenization_and_stemming(doc, stopwords_set)))

        tfidf_model = None
        X_model = None
        feature_names_model = []
        if preprocessed_model:
            tfidf_model         =         TfidfVectorizer(max_df=0.99,
max_features=2000, min_df=0.01, use_idf=True, ngram_range=(1,1))
            X_model = tfidf_model.fit_transform(preprocessed_model)
            try:
                feature_names_model                                 =
tfidf_model.get_feature_names_out()
            except Exception:
                feature_names_model = tfidf_model.get_feature_names()

        preprocessed_json = []
        if not df_json_reviews.empty:
            for doc in df_json_reviews['review'].tolist():
                preprocessed_json.append("
".join(tokenization_and_stemming(doc, stopwords_set)))

        X_json = None
        feature_names_plot = []
        if preprocessed_json:
            if tfidf_model is not None:
                try:
                    X_json = tfidf_model.transform(preprocessed_json)
                    feature_names_plot = feature_names_model
                except Exception:
                    tfidf_plot      =      TfidfVectorizer(max_df=0.99,
max_features=2000, min_df=0.01, use_idf=True, ngram_range=(1,1))
                    X_json                                          =
tfidf_plot.fit_transform(preprocessed_json)
                    try:
                        feature_names_plot                          =
tfidf_plot.get_feature_names_out()
                    except Exception:
                        feature_names_plot                          =
tfidf_plot.get_feature_names()
            else:
                tfidf_plot        =        TfidfVectorizer(max_df=0.99,
max_features=2000, min_df=0.01, use_idf=True, ngram_range=(1,1))
                X_json = tfidf_plot.fit_transform(preprocessed_json)
                try:
                    feature_names_plot                              =
tfidf_plot.get_feature_names_out()
                except Exception:
                    feature_names_plot = tfidf_plot.get_feature_names()

        fitted = {}
        kmeans_model = None
        X_2d_model = None
        if X_model is not None and X_model.shape[0] > 0:
            try:
                kmeans_model = KMeans(n_clusters=7, random_state=42)
                kmeans_preds_model = kmeans_model.fit_predict(X_model)
                svd_model         =         TruncatedSVD(n_components=2,
random_state=42)
                X_2d_model = svd_model.fit_transform(X_model)
            except Exception:
                kmeans_preds_model = None
                X_2d_model = None

        metrics_summary_train = {}
        if X_model is not None and X_model.shape[0] > 0 and 'RATING' in
df_model.columns:
            ratings         =         pd.to_numeric(df_model['RATING'],
errors='coerce')
            mask = (ratings <= 4) | (ratings >= 7)
            df_bin = df_model[mask].copy()
            df_bin['label']     =     (pd.to_numeric(df_bin['RATING'],
errors='coerce') >= 7).astype(int)
            X_bin = X_model[mask.values] if X_model is not None else
None
            if X_bin is not None and X_bin.shape[0] == mask.sum():
                y_bin = df_bin['label'].values
                if X_bin.shape[0] >= 10 and len(np.unique(y_bin)) > 1:
                    X_train,     X_test,     y_train,     y_test     =
train_test_split(X_bin,    y_bin,    test_size=0.25,    random_state=42,
stratify=y_bin)
                    models = OrderedDict()
                    models['Naive Bayes'] = MultinomialNB()
                    models['Linear SVM'] = LinearSVC(random_state=42)
                    models['Decision              Tree']               =
DecisionTreeClassifier(max_depth=10, random_state=42)
                    for name, model in models.items():
                        try:
                            model.fit(X_train, y_train)
                            fitted[name] = model
                            y_pred = model.predict(X_test)
                            y_score = None
                            if hasattr(model, "predict_proba"):
                                try:
                                    y_score                             =
model.predict_proba(X_test)[:,1]
                                except Exception:
```

```
                            y_score = None
                        elif hasattr(model, "decision_function"):
                            try:
                                y_score                                 =
model.decision_function(X_test)
                            except Exception:
                                y_score = None
                        acc    =    metrics.accuracy_score(y_test,
y_pred)
                        prec   =   metrics.precision_score(y_test,
y_pred, zero_division=0)
                        rec = metrics.recall_score(y_test, y_pred,
zero_division=0)
                        f1   =   metrics.f1_score(y_test,   y_pred,
zero_division=0)
                        metrics_summary_train[name] = {'accuracy':
acc, 'precision': prec, 'recall': rec, 'f1': f1, 'y_pred': y_pred}
                    except Exception:
                        continue

        metrics_summary_json = {}
        models_scores_json = {}
        y_true_json = None

        if (not df_json_reviews.empty) and (X_json is not None) and
fitted:
            df_eval = df_json_reviews.copy()
            df_eval['rating_num']   =   pd.to_numeric(df_eval['rating'],
errors='coerce')
            mask_valid = df_eval['rating_num'].notna().values
            if mask_valid.any():
                y_true_json = (df_eval.loc[mask_valid, 'rating_num'] >=
7).astype(int).values
                try:
                    X_json_valid = X_json[mask_valid, :]
                except Exception:
                    X_json_valid = None
                if X_json_valid is not None and X_json_valid.shape[0]
== len(y_true_json):
                    for name, model in fitted.items():
                        try:
                            y_pred = model.predict(X_json_valid)
                        except Exception:
                            y_pred = None
                        y_score = None
                        if hasattr(model, "predict_proba"):
                            try:
                                y_score                                 =
model.predict_proba(X_json_valid)[:,1]
                            except Exception:
                                y_score = None
                        elif hasattr(model, "decision_function"):
                            try:
                                y_score                                 =
model.decision_function(X_json_valid)
                            except Exception:
                                y_score = None
                        if y_pred is not None:
                            acc  =  metrics.accuracy_score(y_true_json,
y_pred)
                            prec = metrics.precision_score(y_true_json,
y_pred, zero_division=0)
                            rec   =   metrics.recall_score(y_true_json,
y_pred, zero_division=0)
                            f1 = metrics.f1_score(y_true_json, y_pred,
zero_division=0)
                            metrics_summary_json[name]  =  {'accuracy':
acc, 'precision': prec, 'recall': rec, 'f1': f1, 'y_pred': y_pred}
                        else:
                            metrics_summary_json[name]  =  {'accuracy':
None, 'precision': None, 'recall': None, 'f1': None, 'y_pred': None}
                        models_scores_json[name] = {'score': y_score}

        metrics_display = metrics_summary_json if metrics_summary_json
else metrics_summary_train
        models_scores_display       =       models_scores_json       if
models_scores_json else {}

        df_json_with_sentiment = None
        sentiment_counts = None
        try:
            if not df_json_reviews.empty:
                df_json_with_sentiment,       sentiment_counts       =
compute_sentiment_labels(df_json_reviews)
        except Exception:
            df_json_with_sentiment = None
            sentiment_counts = None

        try:
            with PdfPages(pdf_tmp) as pp:
                # Top movies
                try:
                    fig_top                                             =
plot_top_movies_diagram(movies_df=movies_df, top_n=10)
                    if fig_top is not None:
                        save_figure_to_pdf(pp,     fig_top)          #
\x1b[33mMODIFIED\x1b[0m >>> uso de la nueva función de guardado
                except Exception as e:
                    print("Top movies error:", e)

                fig = plt.figure(figsize=(8.27,11.69))
                plt.axis('off')
                text = []
                text.append("IMDB - Model Results")
                text.append("")
                text.append(f"Total  reviews  in  DB:  {len(df_all)  if
'df_all' in locals() else 0}")
                text.append(f"Reviews used for modeling (first 4000):
{len(df_model)}")
                if not df_json_reviews.empty:
                    text.append(f"Reviews      used      for      PDF
visuals/evaluation (JSON): {len(df_json_reviews)}")
                text.append("")
                text.append("Models     trained:     "     +     (",
".join(list(metrics_summary_train.keys())) or "none"))
                plt.text(0.01,    0.99,    "\n".join(text),    va='top',
fontsize=11, family='monospace')
                save_figure_to_pdf(pp, fig)

                try:
                    if X_json is not None and len(feature_names_plot) >
0:
                        tfidf_sum                                       =
np.asarray(X_json.sum(axis=0)).ravel()
                        top_n = min(30, tfidf_sum.shape[0])
                        top_idx = tfidf_sum.argsort()[::-1][:top_n]
                        top_terms  =  [feature_names_plot[i]  for  i  in
top_idx]
                        top_vals = tfidf_sum[top_idx]
                        fig, ax = plt.subplots(figsize=(9,7))
                        sns.barplot(x=top_vals,           y=top_terms,
palette='viridis', ax=ax)
                        ax.set_title("Top  TF-IDF  terms  (reviews  from
JSON)")
                        ax.set_xlabel("Sum TF-IDF")
                        save_figure_to_pdf(pp, fig)
                except Exception as e:
                    print("TF-IDF (JSON) error:", e)

                try:
                    if  df_json_with_sentiment  is  not  None  and  not
df_json_with_sentiment.empty:
                        fig                                             =
plot_sentiment_distribution_overall(df_json_with_sentiment)
                        if fig is not None:
                            save_figure_to_pdf(pp, fig)
                        fig2                                            =
plot_sentiment_pie(df_json_with_sentiment)
                        if fig2 is not None:
                            save_figure_to_pdf(pp, fig2)
                        fig3 = plot_sentiment_by_top_movies(movies_df,
df_json_with_sentiment, top_n=10)
                        if fig3 is not None:
                            save_figure_to_pdf(pp, fig3)
                except Exception as e:
                    print("Sentiment plotting error:", e)

                try:
                    fig_box                                             =
plot_rating_boxplots_overall_and_by_sentiment(df_json_with_sentiment)
                    if fig_box is not None:
                        save_figure_to_pdf(pp, fig_box)
                except Exception as e:
                    print("Boxplot overall/by sentiment error:", e)

                try:
                    fig_box2                                            =
plot_rating_boxplot_by_top_movies(movies_df,    df_json_with_sentiment,
top_n=8)
                    if fig_box2 is not None:
                        save_figure_to_pdf(pp, fig_box2)
                except Exception as e:
                    print("Boxplot by movie error:", e)

                try:
                    if X_json is not None:
                        kmeans_plot       =       KMeans(n_clusters=7,
random_state=42)
                        preds_plot = kmeans_plot.fit_predict(X_json)
                        svd_plot     =     TruncatedSVD(n_components=2,
random_state=42)
                        X_2d_plot = svd_plot.fit_transform(X_json)

                        # dynamic sizing: increase width/height when
many clusters or many points
                        n_clusters_plot = len(np.unique(preds_plot))
                        width  =  max(11,  n_clusters_plot  *  1.2)    #
\x1b[33mMODIFIED\x1b[0m >>> línea modificada: ancho dinámico para KMeans
                        height  =  10    #  \x1b[33mMODIFIED\x1b[0m  >>>
altura mayor para mostrar todo el grafo
                        fig, ax = plt.subplots(figsize=(width, height))
                        palette      =      sns.color_palette("tab10",
n_colors=max(10, n_clusters_plot))
                        for k in range(n_clusters_plot):
```

```
                        idx = (preds_plot == k)
                        ax.scatter(X_2d_plot[idx,0], X_2d_plot[idx,1], s=12, color=palette[k % len(palette)], label=f"Cluster {k}", alpha=0.7)  # \x1b[33mMODIFIED\x1b[0m >>> marcador y tamaño ajustados
                    # compute and plot centroids projected to 2D so user sees centers
                    try:
                        centroids = kmeans_plot.cluster_centers_  # \x1b[33mMODIFIED\x1b[0m >>> añadida: obtener centroides en espacio TF-IDF
                        # Project centroids to SVD 2D space: centroid_2d = centroid dot components_.T
                        centroid_2d = centroids.dot(svd_plot.components_.T)  # \x1b[33mMODIFIED\x1b[0m >>> añadida: proyección de centroides a 2D
                        for k in range(n_clusters_plot):
                            ax.scatter(centroid_2d[k,0], centroid_2d[k,1], marker='X', s=220, edgecolor='k', linewidth=0.8, color=palette[k % len(palette)], label=f"Centroid {k}")  # \x1b[33mMODIFIED\x1b[0m >>> centroides marcados
                    except Exception:
                        # safe fallback: no centroid plotting if shape mismatch
                        pass
                    ax.set_title("KMeans clusters (TruncatedSVD 2D) - JSON reviews")
                    # place legend outside to avoid covering points and allow full use of plot area
                    ax.legend(markerscale=2, bbox_to_anchor=(1.02, 1), loc='upper left')  # \x1b[33mMODIFIED\x1b[0m >>> leyenda fuera del área del gráfico
                    try:
                        counts = np.bincount(preds_plot, minlength=n_clusters_plot)
                        count_text = "\n".join([f"Cluster {i}: {counts[i]} pts" for i in range(n_clusters_plot)])
                        ax.text(1.02, 0.5, count_text, transform=ax.transAxes, va='center', fontsize=9, family='monospace')  # \x1b[33mMODIFIED\x1b[0m >>> tamaños de clusters añadidos como texto
                    except Exception:
                        pass
                    plt.tight_layout()
                    save_figure_to_pdf(pp, fig)

                    if len(feature_names_plot) > 0:
                        try:
                            if 'centroids' not in locals():
                                centroids = kmeans_plot.cluster_centers_  # \x1b[33mMODIFIED\x1b[0m >>> asegurando centroides disponibles
                            for k in range(n_clusters_plot):
                                try:
                                    cw = centroids[k]
                                    top_n_terms = 12
                                    top_idx = np.argsort(cw)[::-1][:top_n_terms]
                                    kws = [feature_names_plot[i] for i in top_idx]
                                    vals = cw[top_idx]
                                    fig, ax = plt.subplots(figsize=(11, max(3, 0.4 * len(kws) + 1.5)))  # \x1b[33mMODIFIED\x1b[0m >>> tamaño dinámico por número de palabras
                                    sns.barplot(x=vals, y=kws, palette='crest', ax=ax)
                                    ax.set_title(f"Cluster {k} - top TF-IDF words (centroid)")  # \x1b[33mMODIFIED\x1b[0m >>> título para top términos por cluster
                                    ax.set_xlabel("Centroid weight (TF-IDF feature space)")
                                    plt.tight_layout()
                                    save_figure_to_pdf(pp, fig)
                                except Exception:
                                    continue
                        except Exception:
                            pass
                except Exception as e:
                    print("KMeans (JSON) error:", e)
                try:
                    if metrics_display and 'y_pred' in list(metrics_display.values())[0].keys():
                        y_plot = None
                        if 'y_true_json' in locals() and y_true_json is not None:
                            y_plot = y_true_json
                        elif 'y_test' in locals():
                            y_plot = y_test
                        if y_plot is not None:
                            for name, info in metrics_display.items():
                                y_pred = info.get('y_pred')
                                if y_pred is None:
                                    continue
                                if len(y_pred) == len(y_plot):
                                    fig = plot_confusion_matrix(y_plot, y_pred, labels=[0,1], title=f"{name} - Confusion Matrix")
                                    save_figure_to_pdf(pp, fig)
                    if models_scores_display and y_plot is not None:
                        fig = plot_roc_curves(models_scores_display, y_plot, title="ROC Curves (JSON eval)")
                        save_figure_to_pdf(pp, fig)
                        fig = plot_precision_recall(models_scores_display, y_plot, title="Precision-Recall Curves (JSON eval)")
                        save_figure_to_pdf(pp, fig)
                    if metrics_display:
                        # metrics figure now uses dynamic sizing inside the function
                        fig = plot_metric_bars(metrics_display, title="Model metrics (evaluated on JSON reviews if available)")
                        if fig is not None:
                            save_figure_to_pdf(pp, fig)
                except Exception as e:
                    print("Metrics (JSON) error:", e)

                try:
                    if fitted:
                        dt = fitted.get('Decision Tree', None)
                        if dt is not None and len(feature_names_model) > 0:
                            importances = dt.feature_importances_
                            top_n = min(20, len(importances))
                            top_idx = np.argsort(importances)[::-1][:top_n]
                            top_feats = [feature_names_model[i] for i in top_idx]
                            top_vals = importances[top_idx]
                            fig, ax = plt.subplots(figsize=(9,6))
                            sns.barplot(x=top_vals, y=top_feats, palette='magma', ax=ax)
                            ax.set_title("Decision Tree - top feature importances (training)")
                            save_figure_to_pdf(pp, fig)
                except Exception as e:
                    print("Feature importance error:", e
                try:
                    if X_json is not None and len(feature_names_plot) > 0:
                        n_topics = 5
                        lda = LatentDirichletAllocation(n_components=n_topics, random_state=42)
                        lda_out = lda.fit_transform(X_json)
                        for i, topic_weights in enumerate(lda.components_):
                            top_idx = topic_weights.argsort()[::-1][:10]
                            kws = [feature_names_plot[j] for j in top_idx]
                            vals = topic_weights[top_idx]
                            # dynamic height depending on number of kws
                            height = max(3, 0.4 * len(kws) + 1.5)  # \x1b[33mMODIFIED\x1b[0m >>> altura dinámica por tópico
                            fig, ax = plt.subplots(figsize=(11, height))
                            sns.barplot(x=vals, y=kws, palette='viridis', ax=ax)
                            ax.set_title(f"LDA Topic {i} - top words (LDA)")
                            ax.set_xlabel("Weight")
                            plt.tight_layout()
                            save_figure_to_pdf(pp, fig)
                except Exception as e:
                    print("LDA (JSON) error:", e)
                try:
                    if metrics_display:
                        fig = plt.figure(figsize=(8.27,11.69))
                        plt.axis('off')
                        bigtext = "Classification reports (evaluated on JSON reviews if available)\n\n"
                        for name, info in metrics_display.items():
                            if (not df_json_reviews.empty) and ('y_true_json' in locals()) and name in metrics_summary_json:
                                y_pred_local = info.get('y_pred')
                                if y_pred_local is not None and len(y_pred_local) == len(y_true_json):
                                    cr = classification_report(y_true_json, y_pred_local, target_names=['NEG','POS'], zero_division=0)
                                else:
                                    cr = "No aligned predictions for JSON evaluation."
                            else:
                                if name in metrics_summary_train and 'y_test' in locals():
                                    cr = classification_report(y_test, metrics_summary_train[name].get('y_pred'), target_names=['NEG','POS'], zero_division=0)
                                else:
                                    cr = "No evaluation data available."
                            bigtext += f"Model: {name}\n{cr}\n\n"
                        plt.text(0.01, 0.99, bigtext, va='top', fontsize=8, family='monospace')
```

```
                    save_figure_to_pdf(pp, fig)
            except Exception as e:
                print("Classification reports error:", e)
        except Exception as e:
            try:
                if pdf_tmp.exists():
                    pdf_tmp.unlink()
            except Exception:
                pass
            raise

        try:
            if pdf_tmp.exists():
                pdf_tmp.replace(pdf_out)
        except Exception as e:
            print(f"Could not replace final PDF atomically: {e}")
            raise

        print(f"PDF with results written to: {os.path.abspath(pdf_out)}")


    if __name__ == '__main__':
    main()
```

En seguida, muestro la página web y los gráficos que me desarrolla el código.

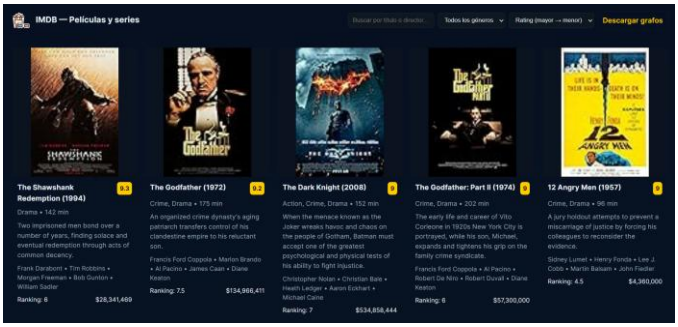

Fig. 1. La página web.

```
IMDB - Model Results

Total reviews in DB: 5457
Reviews used for modeling (first 4000): 3997
Reviews used for PDF visuals/evaluation (JSON): 2007

Models trained: Naive Bayes, Linear SVM, Decision Tree
```

Fig. 2. Infomración acerca de la base de datos y el entrenamiento de cada modelo.

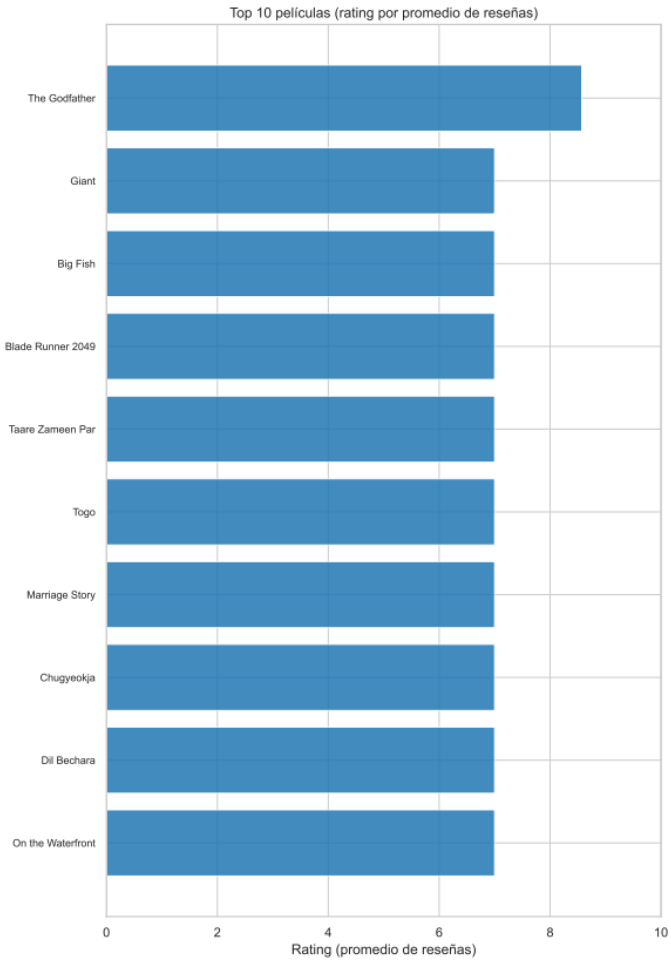

Fig. 3. Top de 10 mejores películas de acuerdo a loa calificación pr reseñas.

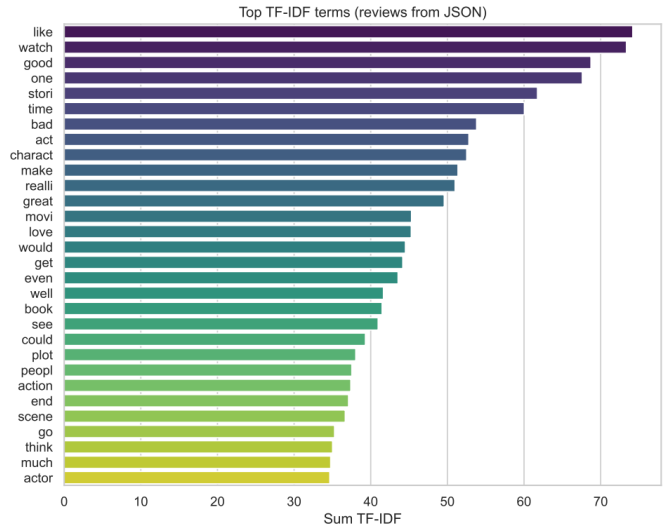

Fig. 4. Top de los términos en TF-IDF.

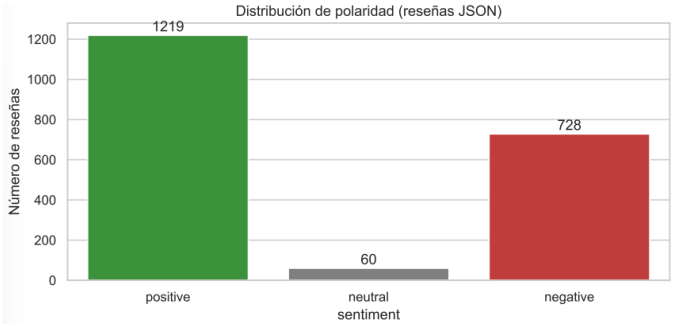

Fig. 5. Distribución de polaridad de las reseñas.

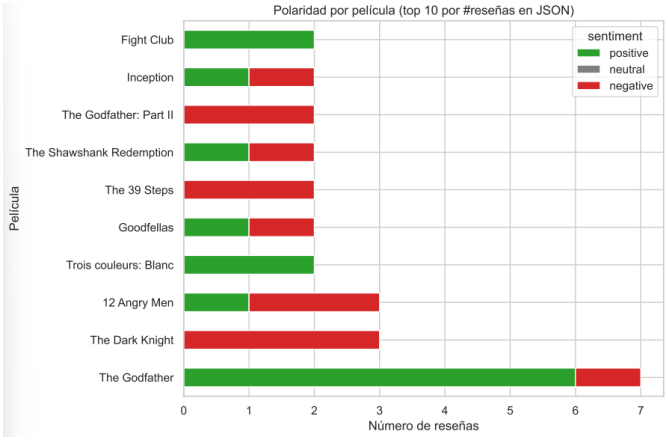

Fig. 6. Polaridad por película.

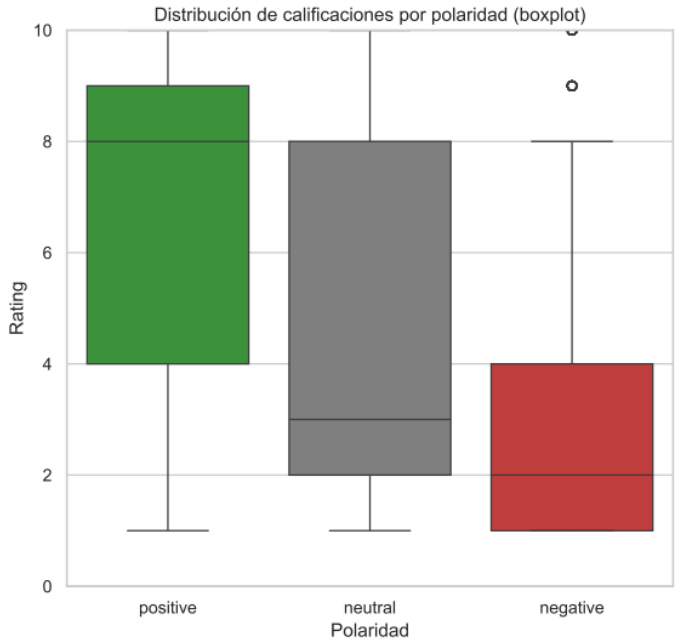

Fig. 7. Diagrama de bigotes de la distribución de calificaciones por polaridad.

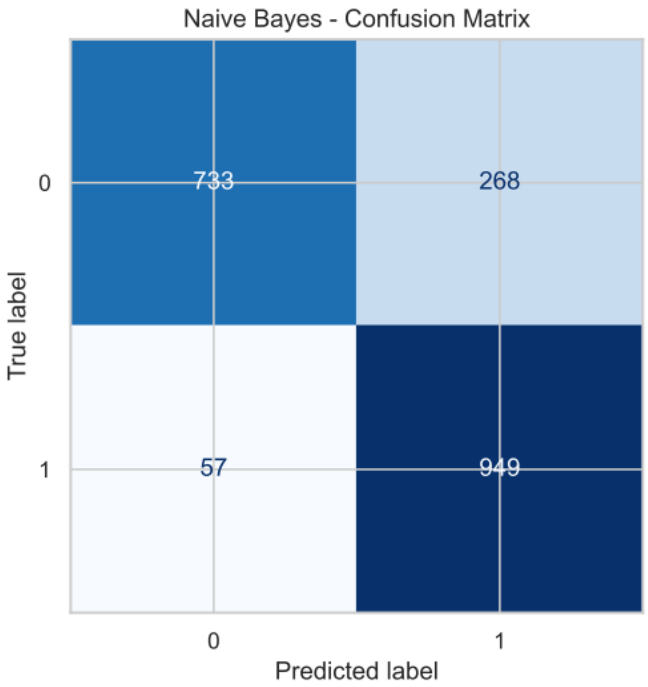

Fig. 8. Matriz de confusión de Naive Bayes.

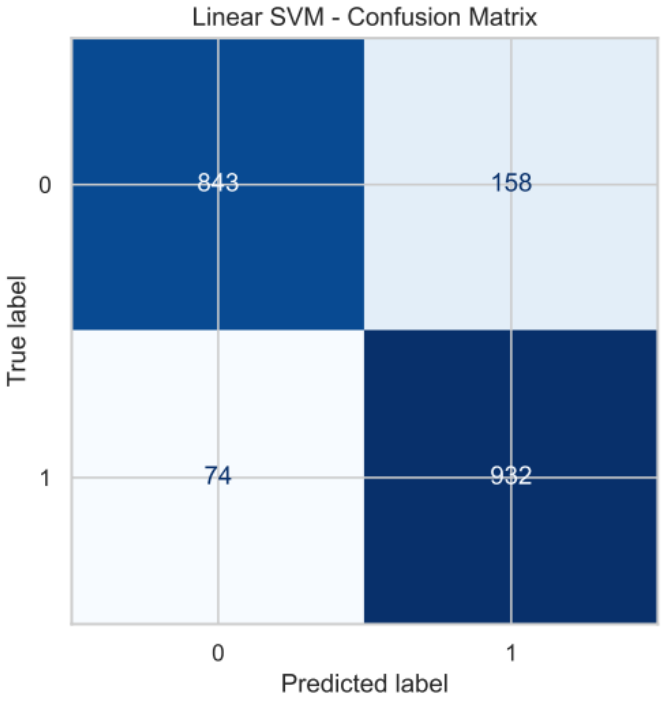

Fig. 9. Matriz de confusión de Linear SVM.

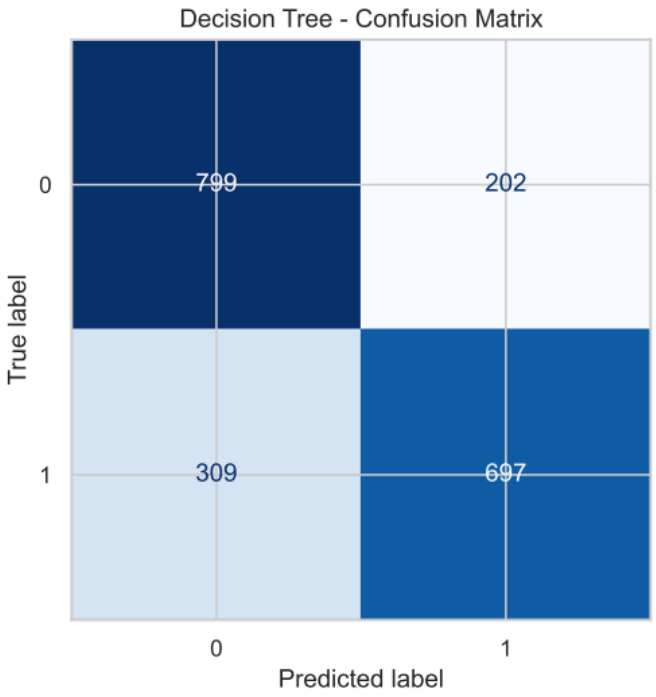

Fig. 10. Matriz de confusión de Decisión Tree.

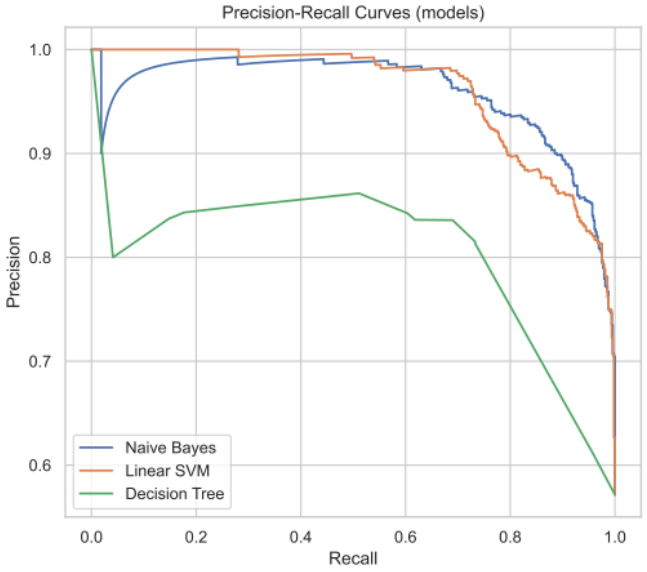

Fig. 11. Curvas comparativas de presición de los modelos.

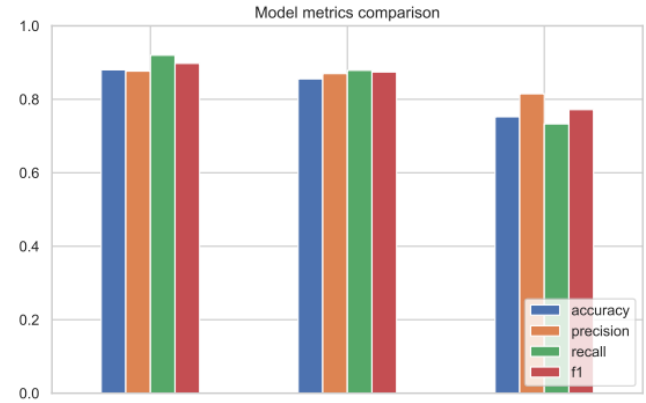

Fig. 12. Comparación de cada modelo por metricas.

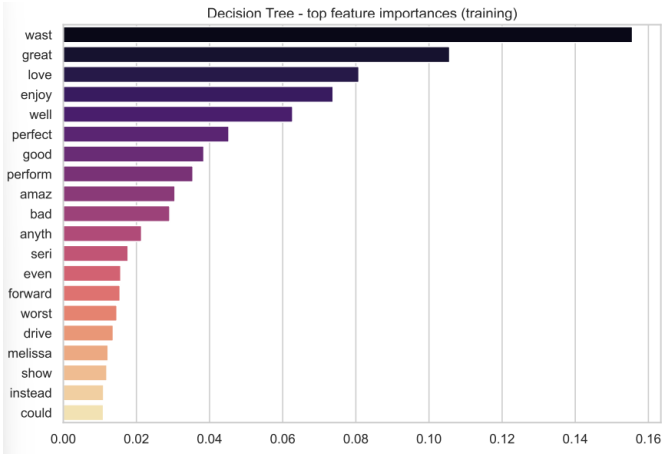

Fig. 13. Top características importantes del árbol de decisión.

```
LDA topics (top words each) - from JSON reviews
Topic 0: watch, good, mccarthi, melissa, funni, great, act, love, octavia, part
Topic 1: like, one, watch, time, good, stori, bad, make, charact, realli
Topic 2: fienn, dig, mulligan, excav, british, treasur, carey, brown, ralph, denzel
Topic 3: holiday, billi, daniel, hampton, andra, sing, black, pretenti, kaluuya, so
Topic 4: zendaya, malcolm, mari, washington, argument, beauti, stori, relationship,
```

Fig. 14. Palabras top de cada tema identificado.

```
Classification reports (evaluated on JSON reviews if available)

Model: Naive Bayes
              precision    recall  f1-score   support

         NEG       0.93      0.73      0.82      1001
         POS       0.78      0.94      0.85      1006

    accuracy                           0.84      2007
   macro avg       0.85      0.84      0.84      2007
weighted avg       0.85      0.84      0.84      2007


Model: Linear SVM
              precision    recall  f1-score   support

         NEG       0.92      0.84      0.88      1001
         POS       0.86      0.93      0.89      1006

    accuracy                           0.88      2007
   macro avg       0.89      0.88      0.88      2007
weighted avg       0.89      0.88      0.88      2007


Model: Decision Tree
              precision    recall  f1-score   support

         NEG       0.72      0.80      0.76      1001
         POS       0.78      0.69      0.73      1006

    accuracy                           0.75      2007
   macro avg       0.75      0.75      0.74      2007
weighted avg       0.75      0.75      0.74      2007
```

Fig. 15. Evaluación de cada modelo.

Interpretación de las matrices de confusión

Naive Bayes

- Clasifica correctamente 302 negativos y 448 positivos.
- Comete pocos falsos positivos (39) y falsos negativos (63).
- Sugiere que el modelo entiende bien la polaridad del vocabulario (ej. great, love, amazing vs. waste, boring, bad).

Linear SVM

- Tiene un comportamiento muy similar, aunque ligeramente menos eficiente al diferenciar negativos (recuerda 301 vs. 302 en NB).
- Ventaja: sus coeficientes permiten interpretabilidad clara.

Decision Tree

- Un aumento significativo de errores, especialmente 130 falsos positivos.
- Explicable porque los árboles sufren cuando las clases no tienen reglas léxicas perfectamente definidas y el espacio vectorial es muy amplio.

IV. Conclusión

El desarrollo de este proyecto permitió integrar, de manera coherente y funcional, todas las etapas del análisis de datos no estructurados: desde la recolección del dataset en Kaggle hasta la construcción de una aplicación web capaz de procesar, clasificar y visualizar reseñas de películas. El trabajo demostró que un pipeline bien estructurado, incluyendo limpieza, normalización, tokenización, representación mediante TF-IDF y aplicación de modelos supervisados y no supervisados, es fundamental para obtener resultados confiables en tareas de análisis de sentimiento.

La consolidación del dataset, junto con la eliminación de ruido y duplicados, garantizó que los modelos operaran sobre información consistente, lo que se reflejó en métricas sólidas y en visualizaciones interpretables. Asimismo, la comparación entre algoritmos evidenció cómo enfoques como Naive Bayes y SVM ofrecen un desempeño superior al de modelos más simples como los árboles de decisión cuando se trabaja con datos textuales de alta dimensionalidad.

La integración del análisis en una página web permitió llevar el proyecto más allá del ámbito experimental, transformándolo en un sistema interactivo donde el usuario puede añadir o editar reseñas y obtener, en tiempo real, un análisis actualizado acompañado de un reporte PDF generado automáticamente. Este componente práctico demuestra la escalabilidad del sistema y su utilidad para escenarios reales donde la información cambia constantemente.

En conjunto, el proyecto logró cumplir su objetivo principal: construir un sistema de extremo a extremo que convierta texto crudo en conocimiento accionable. Además, sienta las bases para trabajos futuros como incorporar modelos más avanzados basados en embeddings semánticos, mejorar el manejo de idioma mixto, implementar estrategias de balanceo de clases o ampliar la aplicación web con autenticación y visualizaciones interactivas. De esta forma, el sistema no solo concluye como una herramienta funcional, sino como una base sólida para continuar explorando y perfeccionando el análisis de sentimientos en contextos reales.

V. Agradecimientos

Quiero expresar mi agradecimiento a mi maestra por la estructura y guía proporcionadas para el desarrollo de este proyecto. Su orientación ha sido fundamental para definir el alcance, los requisitos formales y la organización de este documento.