

**UNIVERSIDAD DE LOS ANDES
DEPARTAMENTO DE INGENIERIA DE
SISTEMAS Y COMPUTACIÓN**



Proyecto 1: Analítica de textos

ISIS3304 – Inteligencia de negocios

María Del Pilar Villamisar

Grupo 29

Juan Pablo Castro 202222086

Manuel Gómez 202020415

Santiago Casasbuenas 202214932

2025-10

Construcción de pipeline:

1. Selección del Modelo

Durante la primera etapa del proyecto se evaluaron múltiples algoritmos de clasificación. Se seleccionó Regresión Logística como modelo final debido a que presentó los mejores resultados en métricas como accuracy, recall, f1-score y tiempo de entrenamiento.

2. Diseño del Pipeline

Para implementar un pipeline robusto y reutilizable, se crearon tres clases, cada una encargada de una etapa crítica en el procesamiento de datos. Estas clases se encuentran en el archivo `Prepipe.py`, ubicado en el path `proyecto1/parte2/back/Pipeline`.

3. Preprocesamiento de Texto (Clean)

La clase `Clean` realiza una limpieza del texto en las columnas `Titulo` y `Descripcion`. Las transformaciones aplicadas incluyen:

- Eliminación de caracteres no ASCII.
- Conversión de texto a minúsculas.
- Eliminación de signos de puntuación y números.
- Remoción de stopwords en español.
- Tokenización mediante `RegexpTokenizer`.
- Aplicación de stemming y lematización para normalización de palabras.

El resultado de esta clase son dos nuevas columnas: `Titulo_Normalizado` y `Descripcion_Normalizada`.

4. Vectorización de Texto (Vectorize)

La clase `Vectorize` convierte el texto normalizado en representaciones numéricas usando el método TF-IDF. Se utilizan dos vectorizadores separados: uno para el título y otro para la descripción. Ambas representaciones se combinan con `hstack`, generando una matriz dispersa optimizada para entrenamiento de modelos.

5. Entrenamiento del Modelo (Model)

La clase `Model` entrena el clasificador de regresión logística. Se realiza una partición de los datos con `train_test_split` (80% entrenamiento, 20% prueba), y se ajusta el modelo con los vectores generados. Posteriormente, se calculan las métricas de desempeño:

- Matriz de confusión.

- Accuracy.
- F1-score.
- Precision.
- Recall.
- Reporte de clasificación completo.

6. Construcción del Pipeline

El archivo Pipeline.py importa las tres clases anteriores, carga el dataset fake_news_spanish.csv, selecciona las columnas relevantes y ejecuta el pipeline completo. Finalmente, el modelo entrenado se guarda en el archivo model.joblib dentro de la carpeta assets utilizando joblib.

Construcción de API:

1. Selección del framework:

Se selecciona FastAPI gracias a su fácil implementación y su uso de PyDantic, el cual permite una mejor implementación al momento de usar los Data Frames de pandas usados por el modelo.

2. DataModel:

Usando PyDantic, se crea una estructura de datos los cuales se van a alimentar al modelo. En este se ponen los tipos de datos y qué datos se van a usar. Esto permite pasar la información de los JSON a los dataframes fácilmente.

3. Endpoints:

Se crean dos endpoints principales. El primero es el “/predict” el cual dado una noticia, retorna la predicción de dicha noticia dada por el modelo del .joblib. El segundo modelo es el de “/train” el cual recibe un csv, el cual se pasa a un JSON y luego a un df para entrenar al modelo usando el pipeline. Se retornan los scores del nuevo modelo con los datos dados.

Construcción de aplicación web:

1. Componentes Principales

La aplicación web React cuenta con dos componentes clave para el análisis de noticias: SingleArticleAnalysis y BatchAnalysis.

2. Análisis Individual (SingleArticleAnalysis)

Este componente permite al usuario ingresar manualmente el título, la descripción y la etiqueta (Label) de una noticia. Al enviar el formulario, los datos son enviados mediante una petición POST a la API (/predict). El resultado se muestra en pantalla, indicando la autenticidad de la noticia junto con la probabilidad asociada.

3. Análisis por Lote (BatchAnalysis)

Este componente permite subir un archivo .csv o .xlsx con múltiples noticias. El archivo es procesado localmente, validando que contenga los campos requeridos: ID, Label, Titulo, Descripción, Fecha. Luego, se transforma a un formato esperado por la API (/train) y se envía. El sistema muestra métricas como accuracy, f1-score, precision, recall, así como un reporte de clasificación y matriz de confusión.

Encuestas a usuarios:

Usuarios finales:

Los usuarios finales son los integrantes del grupo académico que quiere determinar que noticias son falsas y los factores que influyen en la veracidad de una noticia. Asimismo, una persona común que quiere saber si las noticias que lee son falsas o verdaderas es un usuario final. La aplicación sirve para que los académicos puedan enfocarse realizar otras herramientas para detectar noticias falsas, y a su vez sirve como herramienta para esto mismo. Esto hace que el proceso se automatice, y la persona no tenga que revisar cada noticia manualmente para revisar si es verdadera. Esto le ayuda muchísimo al usuario brindándole seguridad de que las noticias que está leyendo son verdaderas y le ahorra el tiempo de filtrar manualmente las noticias falsas.

Validación de Resultados:

Para validar el funcionamiento de la aplicación y probar la facilidad de uso de la aplicación, se realizó una encuesta a 8 personas que utilizaron la aplicación. Estos fueron los resultados:

1. ¿Como calificaría su experiencia con la aplicación?

[More details](#)



2. ¿Qué tanto cree que la predicción se asemeja a la realidad?

[More details](#)



3. ¿Siente que la aplicación es fácil de usar?

[More details](#)



4. ¿Siente que la aplicación es fácil de entender?

[More details](#)



Los resultados muestran que en general, la aplicación es fácil de usar, entender y provee valor al usuario, dando una predicción buena para cada usuario. Se evidencia también que por lo general los usuarios estuvieron satisfechos con la aplicación y se les proveyó valor con el algoritmo predictivo y la facilidad de uso.