Proyecto 1 etapa 1

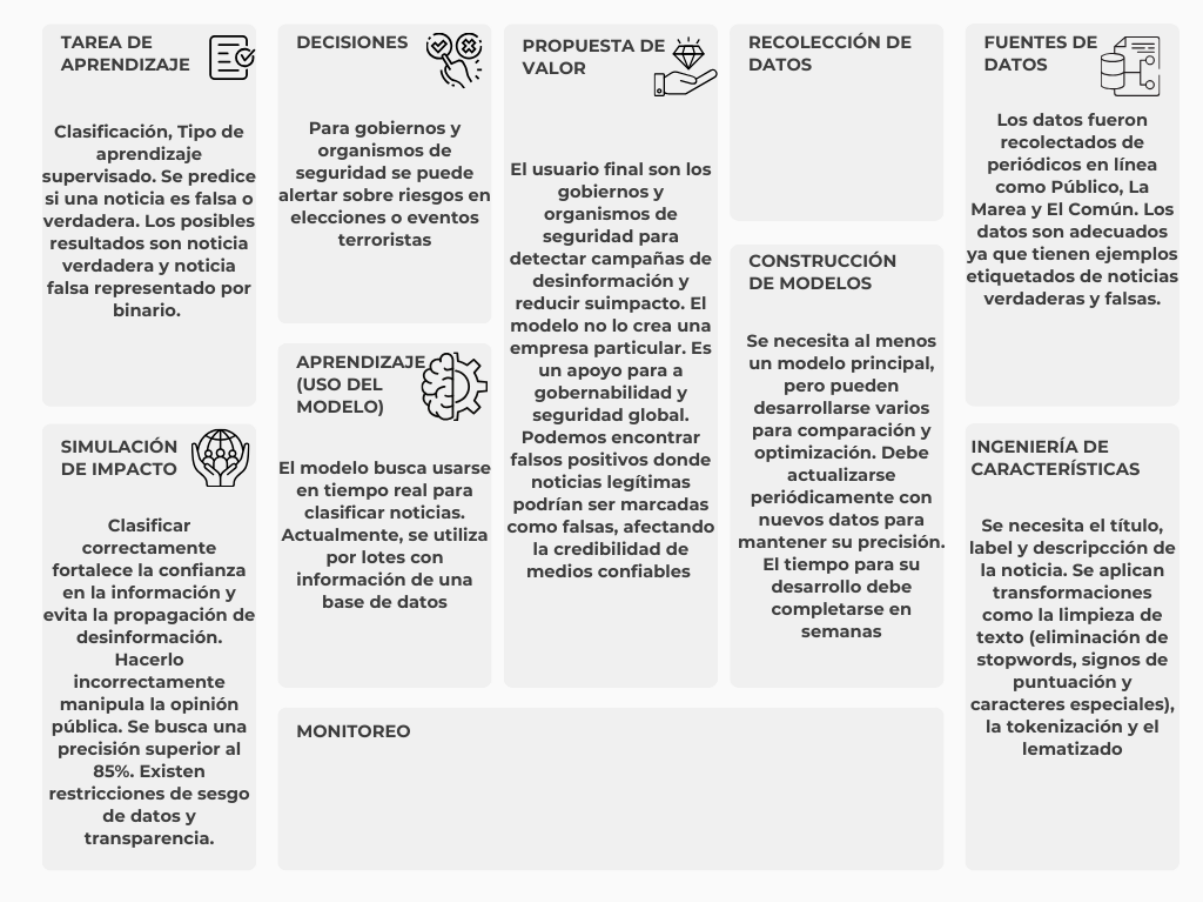
Grupo 24

Juan David Duarte - 202215070

Luisa Hernández - 202114093

Juan Sebastián Sánchez - 202121498

Documentación del proceso de aprendizaje automático



Entendimiento y preparación de los datos

Se realizó un análisis exploratorio de texto para evaluar la distribución de las palabras y la cantidad de registros en cada categoría (noticia real o falsa). Adicionalmente, se analiza la longitud promedio de los textos y la presencia de caracteres especiales o signos de puntuación que puedan afectar el procesamiento posterior.

En la preparación y transformación de los datos, se aplican técnicas de limpieza como conversión a minúsculas, eliminación de números, signos de puntuación y caracteres especiales. También se eliminan palabras irrelevantes (stopwords) y se normalizan los términos mediante lemmatización y stemming.

Modelado y evaluación

Se utilizaron tres algoritmos diferentes de clasificación (Árbol de decisión, KNN y Gradient Boosting). Se presentaron las métricas de evaluación respectivas junto a la matriz de confusión con el fin de determinar el mejor modelo para esta labor dados los datos.

Resultados

El primer modelo analizado es K-Nearest Neighbors (KNN), el cual arrojó los siguientes resultados. Los mejores hiperparámetros seleccionados fueron n_neighbors=7 y weights='uniform'. En el conjunto de prueba, el modelo obtuvo una exactitud del 54%, un recall del 49.05%, una precisión del 64.57% y un F1-score de 55.76%. Al analizar la matriz de confusión, se observó un alto número de falsos negativos, con 5108 muestras de la clase 1 clasificadas erróneamente como 0. Además, el modelo mostró una mejor precisión que recall, lo que indica que es más conservador al clasificar positivos, pero no detecta bien todos los casos de la clase minoritaria. En este sentido, KNN tiende a favorecer la precisión sobre el recall, lo que puede resultar menos efectivo si el objetivo es minimizar falsos negativos. Si se busca maximizar el recall, podría ser necesario ajustar los hiperparámetros o probar con otros modelos.

El segundo modelo analizado fue Árbol de Decisión, cuyos mejores hiperparámetros seleccionados fueron criterion='gini' y max_depth=10. En el conjunto de prueba, obtuvo una exactitud del 84%, un recall del 80.62%, una precisión del 88.29% y un F1-score de 81.89%. El análisis de la matriz de confusión mostró un número considerablemente menor de falsos negativos, con solo 98 muestras de la clase 1 clasificadas erróneamente como 0. En comparación con KNN, este modelo logra un mejor equilibrio entre precisión y recall, capturando la mayoría de los casos de la clase positiva sin comprometer demasiada precisión.

El tercer modelo analizado fue Gradient Boosting, con los mejores hiperparámetros seleccionados: max_depth=5 y n_estimators=100. En el conjunto de prueba, obtuvo una exactitud del 89%, un recall del 87.19%, una precisión del 91.02% y un F1-score de 88.28%. Al analizar la matriz de confusión, se observó un número muy bajo de falsos negativos, con solo 187 muestras de la clase 1 clasificadas erróneamente como 0. Este modelo logra un buen equilibrio entre precisión y recall, permitiendo detectar la mayoría de los casos positivos sin comprometer la precisión.

En conclusión, si el objetivo es maximizar la capacidad de detección de la clase positiva sin perder precisión, Gradient Boosting es la mejor opción. Si se prefiere un modelo más interpretable con un rendimiento aceptable, el Árbol de Decisión puede ser una

alternativa. Sin embargo, KNN no es recomendable en este caso, debido a su bajo desempeño en la detección de la clase minoritaria.

Enlace del video

Enlace público que redirige al video:

https://www.canva.com/design/DAGf2pZ7cjQ/8BA-cvnp03U7Lv8yo_HK8A/watch?utm_content=DAGf2pZ7cjQ&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=h59e0669c87

En el desarrollo del proyecto, nuestro equipo estuvo conformado por tres integrantes: Juan Sebastián Sánchez, Juan David Duarte y Luisa Hernandez. A continuación, se detallan los roles asumidos, las tareas realizadas por cada integrante, los tiempos dedicados, los retos enfrentados y la manera en que se resolvieron, así como el uso de ChatGPT en el proceso.

Roles y responsabilidades

- **Juan Sebastián (Líder de proyecto)**
  - Definió el cronograma del equipo y organizó las reuniones.
  - Asignó tareas asegurando una distribución equitativa del trabajo.
  - Subió la entrega final al sistema de la universidad.
  - Horas dedicadas: 20 horas.
- **Juan David (Líder de datos)**
  - Gestionó la recopilación, limpieza y preprocesamiento de los datos.
  - Aseguró que los datos estuvieran disponibles en el repositorio de Git.
  - Validó la calidad y coherencia de los datos antes del análisis.
  - Horas dedicadas: 25 horas.
- **Luisa (Líder de analítica)**
  - Seleccionó los modelos de análisis y evaluó su rendimiento.
  - Comparó distintos modelos para encontrar la mejor solución.
  - Verificó que los entregables cumplieran con los requerimientos.
  - Horas dedicadas: 22 horas.

Retos enfrentados y soluciones

1. **Manejo de datos inconsistentes:** Se encontraron valores duplicados y datos desbalanceados. Se resolvió aplicando técnicas de imputación y balanceo de clases.

2. **Dificultad en la interpretación de resultados:** Se realizaron reuniones adicionales para mejorar la comprensión de los modelos y ajustar hiperparámetros.
3. **Compatibilidad de librerías:** Durante la implementación del modelo, encontramos diferencias en las versiones de librerías como scikit-learn, pandas y scipy, lo que generaba errores al ejecutar el código en diferentes entornos. Lo solucionamos actualizando las librerías a su última versión.
4. **Implementación de tokenización:** La tokenización inicial generaba una matriz dispersa muy grande, afectando el rendimiento del modelo. Optimizamos la tokenización reduciendo el número de características con max_features, filtrando palabras con min_df y max_df

Uso de ChatGPT

ChatGPT fue utilizado en diversas etapas del proyecto:

- Corrección de funciones para el preprocesamiento de datos y limpieza.
- Depuración de errores en código Python.
- Solución de errores en el uso de las librerías

Distribución de puntos y mejoras para la próxima entrega

Juan Sebastián	33
Juan David	34
Luisa	33

Puntos por mejorar para la siguiente entrega

1. **Mejor documentación del código**: Se sugiere agregar comentarios más detallados en el código para facilitar su comprensión y futura reutilización.
2. **Automatización de tareas**: Se explorará el uso de pipelines en scikit-learn para automatizar el preprocesamiento de datos.
3. **Mayor control de versiones**: Se recomienda utilizar git branches para evitar conflictos en la integración del código.