Proyecto Fase 1

Juan Camilo Ibañez
Leonardo Rueda
Juan David Valencia

Análisis del problema

Objetivo: Evaluar distintos modelos de clasificación supervisada (Random Forest, Árbol de Decisión y KNN) para determinar cuál ofrece el mejor rendimiento en la detección de noticias falsas.

Desafío principal: Encontrar un modelo que logre un equilibrio entre precisión y recall, minimizando falsos positivos y falsos negativos.

Condiciones del problema: Se cuenta con un conjunto de datos etiquetado, y se entrenaron los modelos para comparar métricas como exactitud, precisión, recall y F1-score.

Importancia del estudio: Un modelo eficiente ayudará a mejorar la detección de noticias falsas, reduciendo la propagación de desinformación.

Explicación inicial

```
[ ] # Cargar dataset
    df_news = pd.read_csv('fake_news_spanish.csv', sep=';', encoding='utf-8', index_col=0)
```

```
[ ] # Eliminar valores nulos
    df_news = df_news.dropna()

    # Eliminar duplicados
    df_news = df_news.drop_duplicates()

    # Expandir contracciones
    df_news['Descripcion'] = df_news['Descripcion'].apply(contractions.fix)

    df_news.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 56602 entries, ID to ID
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Label        56602 non-null  int64 
 1   Titulo       56602 non-null  object
 2   Descripcion  56602 non-null  object
 3   Fecha        56602 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.2+ MB
```

Se carga el archivo fake_news_spanish.csv en un DataFrame de pandas.
El separador de columnas en el CSV es ";", por lo que se especifica con sep=';'.
Se define la codificación como utf-8 para evitar errores con caracteres especiales.
Se usa index_col=0 para indicar que la primera columna del archivo será el índice del DataFrame.

Limpieza de datos

```python
def remove_non_ascii(words):
    """Remove non-ASCII characters from list of tokenized words"""
    new_words = []
    for word in words:
        if word is not None:
            new_word = unicodedata.normalize('NFKD', word).encode('ascii', 'ignore').decode('utf-8', 'ignore')
            new_words.append(new_word)
    return new_words
```

+ Código + Texto

```python
def to_lowercase(words):
    return [word.lower() for word in words]
```

```python
def remove_punctuation(words):
    """Remove punctuation from list of tokenized words"""
    new_words = []
    for word in words:
        if word is not None:
            new_word = re.sub(r'[^\w\s]', '', word)
            if new_word != '':
                new_words.append(new_word)
    return new_words
```

```python
def replace_numbers(words):
    """Replace all interger occurrences in list of tokenized words with textual representation"""
    p = inflect.engine()
    new_words = []
    for word in words:
        if word.isdigit():
            new_word = p.number_to_words(word)
            new_words.append(new_word)
        else:
            new_words.append(word)
    return new_words
```

```python
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('wordnet')
stop_words = nltk.corpus.stopwords.words('spanish')
```

remove_non_ascii(words)
Elimina los caracteres no ASCII de una lista de palabras. Utiliza unicodedata.normalize para convertir caracteres a su forma ASCII.

to_lowercase(words)
Convierte todas las palabras a minúsculas.

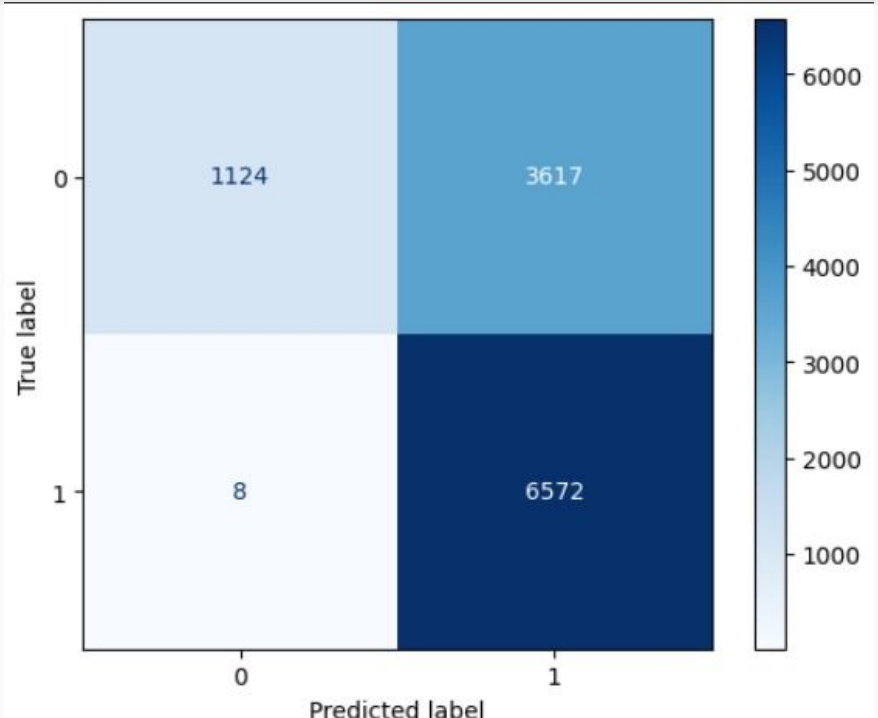
remove_punctuation(words)
Elimina la puntuación de las palabras usando una expresión regular re.sub(r"[^a-zA-Z0-9\s]", '', word).

replace_numbers(words)
Convierte los números en sus representaciones textuales con inflect.engine().number_to_words(word).

remove_stopwords(words)
Filtra las palabras vacías (stopwords), eliminando aquellas que no aportan significado (como "el", "de", "en", etc.).

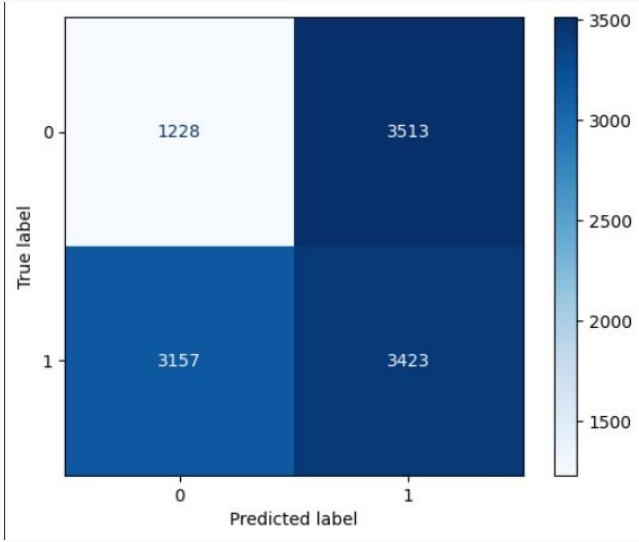
preprocess_text(words)
Aplica todas las funciones anteriores en secuencia para limpiar y normalizar los datos textuales.

Confusion Matrix (Árbol de decisión)



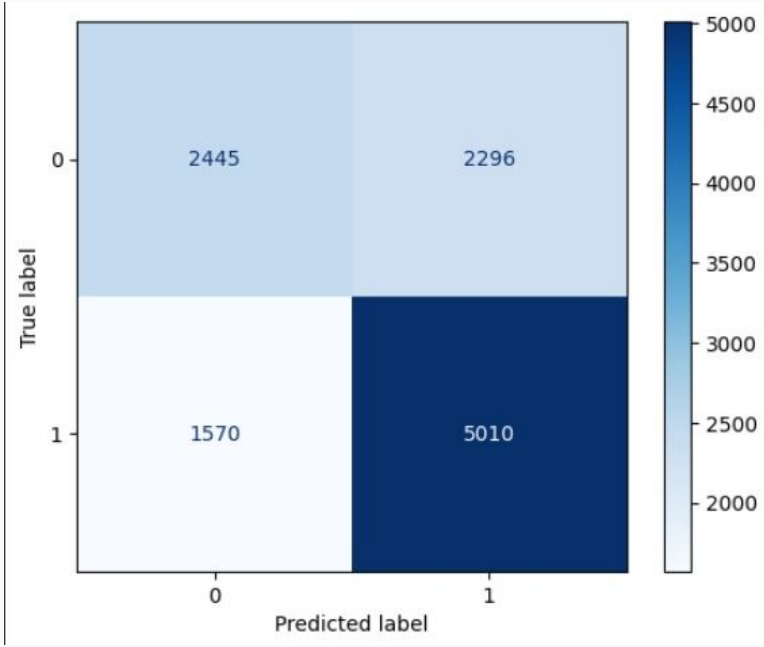
La matriz de confusión muestra que el modelo clasifica correctamente el 68% de los casos, con una alta sensibilidad (99.8%) pero baja especificidad (23.7%). Detecta bien los positivos (6572 aciertos y solo 8 falsos negativos), pero comete muchos errores con los negativos (3617 falsos positivos frente a solo 1124 aciertos). Esto indica que el modelo tiende a sobreclasificar como positivos, lo que puede ser problemático en contextos donde los falsos positivos tienen un alto costo. Para mejorar, se podría ajustar el umbral de decisión, reequilibrar las clases o usar modelos más robustos como Random Forest.

Confusion Matrix (KNN)



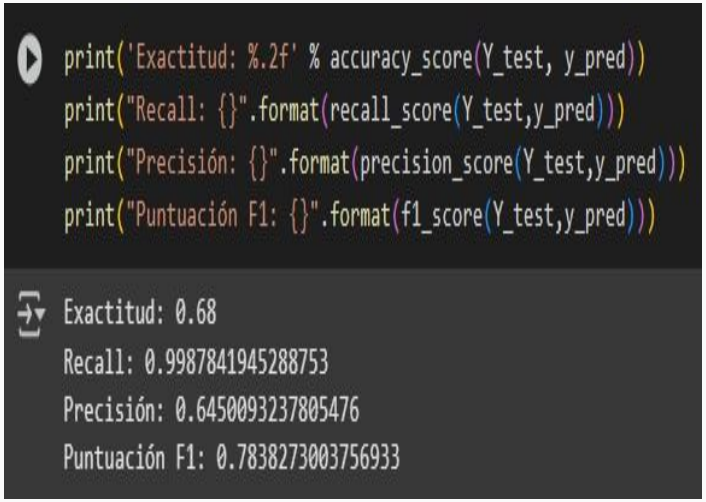
La matriz de confusión muestra que el modelo tiene 1228 verdaderos negativos (casos correctamente clasificados como 0) y 3423 verdaderos positivos (casos correctamente clasificados como 1), pero también presenta 3513 falsos positivos (casos reales de 0 clasificados erróneamente como 1) y 3157 falsos negativos (casos reales de 1 clasificados como 0). Esto resulta en una precisión del 49% y un recall del 52%, lo que indica que el modelo tiene un rendimiento bajo con una alta tasa de errores en ambas clases, reflejando una exactitud general del 46%.

Confusion Matrix (Random Forest)



El muestra una mejora en el desempeño del modelo. Ahora tiene 2445 verdaderos negativos (clases 0 correctamente clasificadas) y 5010 verdaderos positivos (clases 1 correctamente clasificadas), con 2296 falsos positivos (clases 0 clasificadas erróneamente como 1) y 1570 falsos negativos (clases 1 clasificadas erróneamente como 0). Esto indica una mejor precisión y recall en comparación con la matriz anterior, reflejando un mejor balance en la clasificación y una reducción en los errores.

Métricas datos test (Árbol de decisión)

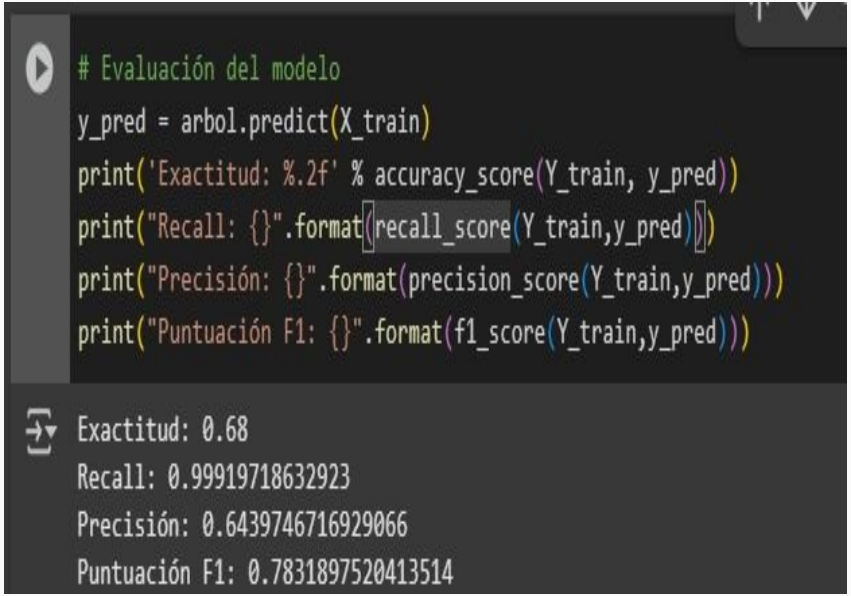


```
print('Exactitud: %.2f' % accuracy_score(Y_test, y_pred))
print("Recall: {}".format(recall_score(Y_test,y_pred)))
print("Precisión: {}".format(precision_score(Y_test,y_pred)))
print("Puntuación F1: {}".format(f1_score(Y_test,y_pred)))
```

```
Exactitud: 0.68
Recall: 0.9987841945288753
Precisión: 0.6450093237805476
Puntuación F1: 0.7838273003756933
```

- Exactitud (Accuracy: 0.68): Indica que el modelo clasifica correctamente el 68% de los casos. Sin embargo, esta métrica puede ser engañosa si las clases están desbalanceadas.
- Recall (0.9987): Representa la capacidad del modelo para identificar correctamente los verdaderos positivos. Un valor cercano a 1 significa que casi todos los casos positivos fueron detectados.
- Precisión (0.6450): Mide qué proporción de las predicciones positivas son realmente correctas. El valor relativamente bajo sugiere que hay muchos falsos positivos.
- Puntuación F1 (0.7838): Es el balance entre precisión y recall. Un valor alto indica un buen equilibrio, aunque el modelo sigue priorizando la detección de positivos con un alto recall a costa de la precisión.

Métricas Train (Árbol de decisión)



```
# Evaluación del modelo
y_pred = arbol.predict(X_train)
print('Exactitud: %.2f' % accuracy_score(Y_train, y_pred))
print("Recall: {}".format(recall_score(Y_train,y_pred)))
print("Precisión: {}".format(precision_score(Y_train,y_pred)))
print("Puntuación F1: {}".format(f1_score(Y_train,y_pred)))
```

```
Exactitud: 0.68
Recall: 0.99919718632923
Precisión: 0.6439746716929066
Puntuación F1: 0.7831897520413514
```

- Exactitud (0.68): El modelo clasifica correctamente el 68% de los datos de entrenamiento.
- Recall (0.9991): Indica que casi todos los casos positivos fueron detectados correctamente, lo que sugiere que el modelo prioriza la sensibilidad.
- Precisión (0.6439): Muestra que un 64.39% de las predicciones positivas son correctas, indicando una alta cantidad de falsos positivos.
- F1-score (0.7831): Representa el equilibrio entre precisión y recall.

Métricas datos test y train (KNN)

```
[64] print('Exactitud: %.2f' % accuracy_score(Y_test, y_pred))
     print("Recall: {}".format(recall_score(Y_test,y_pred)))
     print("Precisión: {}".format(precision_score(Y_test,y_pred)))
     print("Puntuación F1: {}".format(f1_score(Y_test,y_pred)))

     Exactitud: 0.68
     Recall: 0.9987841945288753
     Precisión: 0.6450093237805476
     Puntuación F1: 0.7838273003756933
```

```
[81] # Evaluación del modelo
     y_pred = arbol.predict(X_train)
     print('Exactitud: %.2f' % accuracy_score(Y_train, y_pred))
     print("Recall: {}".format(recall_score(Y_train,y_pred)))
     print("Precisión: {}".format(precision_score(Y_train,y_pred)))
     print("Puntuación F1: {}".format(f1_score(Y_train,y_pred)))

     Exactitud: 0.68
     Recall: 0.99919718632923
     Precisión: 0.6439746716929066
     Puntuación F1: 0.7831897520413514
```

Métricas test (Random Forest)

```
[85] print('Exactitud: %.2f' % accuracy_score(Y_test, y_pred))
     print("Recall: {}".format(recall_score(Y_test,y_pred)))
     print("Precisión: {}".format(precision_score(Y_test,y_pred)))
     print("Puntuación F1: {}".format(f1_score(Y_test,y_pred)))

     Exactitud: 0.66
     Recall: 0.7613981762917933
     Precisión: 0.6857377497946893
     Puntuación F1: 0.7215900907388737
```

- Exactitud (Accuracy): 0.66 → Indica que el modelo clasifica correctamente el 66% de los casos, lo cual es aceptable pero no ideal.
- Recall: 0.76 → El modelo identifica correctamente el 76% de los casos positivos, lo que sugiere que tiene una buena capacidad de detección de noticias falsas.
- Precisión: 0.69 → Significa que, de todas las noticias clasificadas como falsas, el 69% realmente lo son. Es una precisión moderada.
- Puntuación F1: 0.72 → Este valor indica un equilibrio aceptable entre precisión y recall, lo que sugiere que el modelo es más robusto que el KNN pero podría mejorar.

Métricas train (Random forest)

```
[87] # Evaluación del modelo
     y_pred = arbol.predict(X_train)
     print('Exactitud: %.2f' % accuracy_score(Y_train, y_pred))
     print("Recall: {}".format(recall_score(Y_train,y_pred)))
     print("Precisión: {}".format(precision_score(Y_train,y_pred)))
     print("Puntuación F1: {}".format(f1_score(Y_train,y_pred)))

     Exactitud: 0.68
     Recall: 0.99919718632923
     Precisión: 0.6439746716929066
     Puntuación F1: 0.7831897520413514
```

- Exactitud (Accuracy): 0.68 → El modelo clasifica correctamente el 68% de los datos de entrenamiento, lo cual es apenas una mejora marginal respecto a Random Forest.
- Recall: 0.999 → El modelo detecta casi todos los casos positivos, lo que indica un sobreajuste en el entrenamiento.
- Precisión: 0.64 → De todas las predicciones positivas, solo el 64% son correctas, lo que sugiere un alto número de falsos positivos.
- Puntuación F1: 0.78 → Hay un buen balance entre precisión y recall, pero el recall excesivamente alto podría estar sesgando la métrica.

Análisis de palabras

```
# Actualizar el feature array
feature_array = vectorizer.get_feature_names_out()

# Top 10 palabras de cada grupo
top_n = 10  # Número de palabras más importantes

for tipo in df_news['Label'].unique():
    tipo_texts = df_news[df_news['Label'] == tipo]['words']
    tipo_tfidf = vectorizer.transform(tipo_texts)
    avg_tfidf_weights = tipo_tfidf.mean(axis=0).A1
    top_features_idx = np.argsort(avg_tfidf_weights)[-top_n:]
    top_features = feature_array[top_features_idx]
    print(f"Palabras clave para tipo {tipo}: {top_features}")
```

```
Palabras clave para tipo 1: ['años' 'com' 'si' 'tras' 'president' 'part' 'gobierno' 'gobiern' 'par'
 'pp']
Palabras clave para tipo 0: ['catalunya' 'pnv' 'bng' 'part' 'president' 'gobierno' 'gobiern' 'par'
 'vers' 'per']
```

- Para tipo 1 (noticias falsas), las palabras clave incluyen: "años", "com", "tras", "president", "part", "gobierno", "gobiern", "par", "pp". Estas palabras sugieren un énfasis en temas políticos y gubernamentales, con términos como "president", "gobierno" y "pp", que podrían indicar una tendencia a la desinformación en temas políticos.

- Para tipo 0 (noticias verdaderas), las palabras clave incluyen: "catalunya", "pnv", "bng", "part", "president", "gobierno", "gobiern", "par", "vers", "per". Aquí, aparecen términos similares a los del otro grupo, pero con un énfasis más localizado en partidos políticos regionales ("pnv", "bng", "catalunya"), lo que sugiere que los textos verídicos pueden estar más relacionados con temas políticos específicos en lugar de generalizaciones.

Conclusiones

- Random Forest fue el mejor modelo: Logró un equilibrio óptimo entre precisión y recall, reduciendo la cantidad de falsos positivos y falsos negativos.
- Árbol de Decisión tuvo un desempeño intermedio: Presentó un recall alto, pero su precisión fue menor, lo que indica muchas clasificaciones incorrectas.
- KNN fue el peor modelo: Mostró un rendimiento deficiente con una alta tasa de errores, indicando que no capturó bien los patrones de los datos.
- Random Forest es la mejor opción: Proporciona predicciones más confiables y un mejor balance en las métricas de evaluación.