

# Proyecto Fase 1

Juan Camilo Ibañez  
Leonardo Rueda  
Juan David Valencia

# 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   Título      56602 non-null  object
2   Descripción  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

```
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('NFD', word).encode('ascii', 'ignore').decode('utf-8', 'ignore')  
            new_words.append(new_word)  
    return new_words
```

[+ Código](#)[+ Texto](#)

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

```
[ ] 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
```

```
[ ] def replace_numbers(words):  
    """Replace all integer 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
```

```
[ ] 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)`.

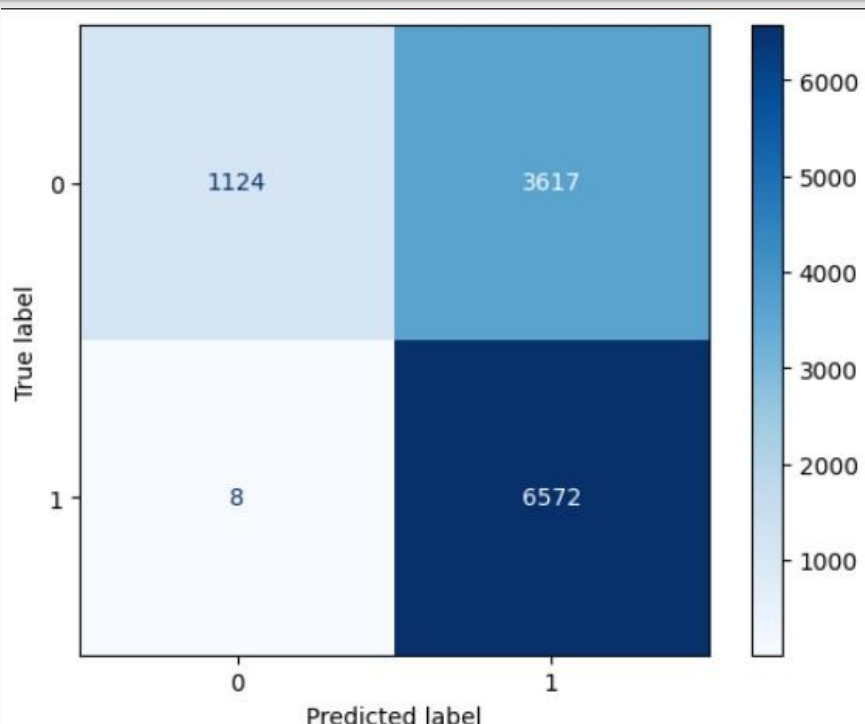
## **remove\_stopwords(words)**

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

## **preprocess\_text(words)**

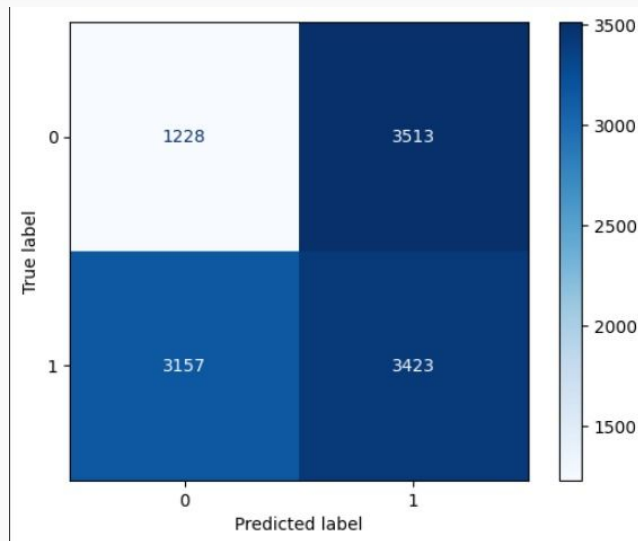
Aplica todas las funciones anteriores en secuencia para limpiar y normalizar los datos textuales.

# Confusion Matrix (Árbol de decisión)



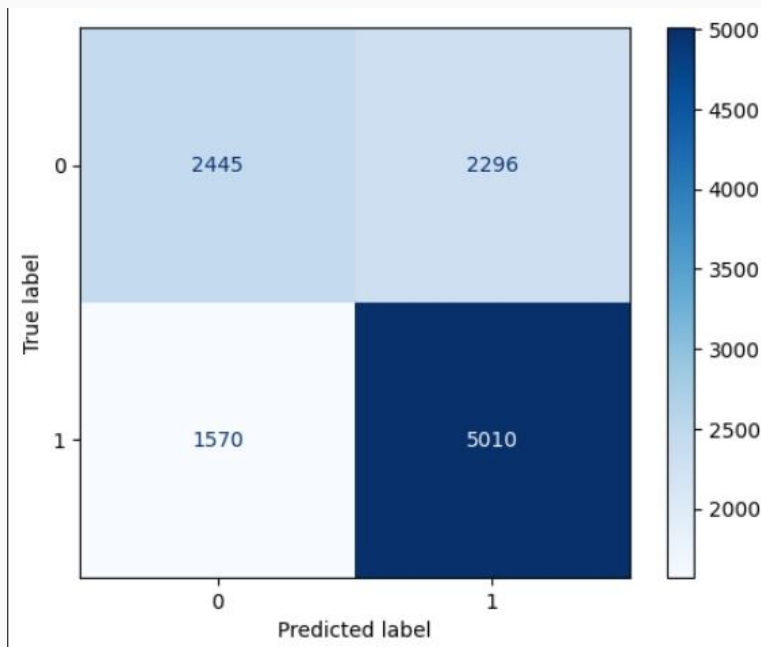
La matriz de confusión muestra que el modelo clasifica correctamente el 68% de los casos, con una alta sensibilidad (99.8%) pero baja especificidad (23.7%). Detecta bien los positivos (6572 aciertos y solo 8 falsos negativos), pero comete muchos errores con los negativos (3617 falsos positivos frente 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
```

# Explicación

Se muestran resultados tanto en los datos de entrenamiento ( $X_{\text{train}}$ ,  $Y_{\text{train}}$ ) como en los de prueba ( $X_{\text{test}}$ ,  $Y_{\text{test}}$ ). En ambos casos, la exactitud (accuracy) es del 68%, lo que indica que el modelo clasifica correctamente el 68% de las muestras. El recall es muy alto ( $\sim 0.999$ ), lo que significa que casi todos los ejemplos positivos fueron correctamente identificados. Sin embargo, la precisión es baja ( $\sim 0.645$ ), lo que sugiere que hay una cantidad considerable de falsos positivos. La puntuación F1, que equilibra precisión y recall, es aproximadamente 0.78, indicando un rendimiento moderado del modelo. La similitud entre los resultados en entrenamiento y prueba sugiere que no hay sobreajuste significativo.

