Introducción a la Inteligencia Artificial

Proyecto 3

Andres Felipe Ramirez Fajardo

Clasificador de emociones en texto

1. Definición del Problema (Antecedentes)

El análisis automático de emociones en texto se ha convertido en una herramienta útil para muchas aplicaciones modernas: asistentes virtuales, análisis de redes sociales, sistemas de recomendación, monitoreo del estado emocional de usuarios, entre otros. Sin embargo, este tipo de tareas presenta varios desafíos, especialmente cuando se trabaja con datos desbalanceados o con modelos que dependen fuertemente de la representación vectorial del lenguaje.

Para este proyecto se utilizó un *Emotion Dataset* disponible públicamente en Kaggle. El conjunto de datos contiene frases cortas etiquetadas en seis categorías emocionales: sadness, joy, love, anger, fear y surprise. Debido a que algunas clases están sobrerrepresentadas y otras contienen pocos ejemplos , el dataset introduce un desbalance importante que afecta directamente el rendimiento de los clasificadores.

El objetivo principal del trabajo es construir y comparar modelos capaces de clasificar correctamente la emoción expresada en un texto corto, evaluando su desempeño antes y después de aplicar técnicas para corregir el desbalance. Para esto se emplearon dos algoritmos clásicos de Machine Learning:

- Naive Bayes, un modelo probabilístico basado en independencia condicional.

- Perceptrón Multicapa (MLP), una red neuronal simple con una capa oculta.

Como métrica de éxito se utilizó principalmente el accuracy sobre el conjunto de prueba, aunque también se analizaron métricas por clase como precision, recall y F1-score, que permiten entender mejor el impacto del desbalance. Para estudiar este efecto se entrenaron los modelos tanto con los datos originales como con una versión balanceada mediante Random Oversampling (ROS).

2. Definición de Algoritmos y Decisiones de Diseño

Para abordar el problema de clasificación de emociones, se implementaron dos enfoques distintos de aprendizaje supervisado. La idea fue comparar un modelo probabilístico clásico contra una red neuronal sencilla, y observar cómo ambos reaccionan al desbalance del dataset.

2.1 Naive Bayes (MultinomialNB)

El primer algoritmo utilizado fue **Multinomial Naive Bayes**, un modelo muy común en tareas de procesamiento de texto gracias a su simplicidad y buen desempeño con datos representados como frecuencias o TF-IDF.

Decisiones clave en el diseño

- **Vectorización con TF-IDF:**
  El texto fue limpiado (lowercase, eliminación de signos y stopwords) y transformado en vectores usando Tfidf Vectorizer. Esta representación funcionó bien con Naive Bayes, ya que captura la importancia relativa de cada palabra.

- **Uso del modelo MultinomialNB:**
  Se eligió este tipo específico porque está diseñado para datos donde las características son recuentos o pesos positivos.

- **Evaluación antes y después de ROS:**
  Debido al desbalance entre clases, Naive Bayes tiende a favorecer las clases mayoritarias. Por eso se evaluó su comportamiento tanto con datos originales como con datos balanceados.

2.2 Perceptrón Multicapa (MLPClassifier)

El segundo algoritmo fue un **MLP**, una red neuronal simple capaz de aprender relaciones no lineales entre los vectores de texto y las emociones.

Arquitectura del modelo

- **Capas ocultas:**
  Se usó una arquitectura con 1 capa oculta de 100 neuronas, suficiente para un problema de clasificación multicategoría sin sobreajustar demasiado.

- **Función de activación:**
  `ReLU`, por su buen comportamiento y eficiencia en redes pequeñas.

- **Optimizador:**
  `Adam`, ya que converge rápido y funciona bien sin una gran cantidad de ajuste manual.

- **Iteraciones máximas:**
  `max_iter=150`, lo cual permitió que la red convergiera sin tiempos de entrenamiento largos.

Decisiones clave en el diseño

- **Se usó el mismo vectorizador TF-IDF** que para Naive Bayes, para garantizar una comparación justa.

- **No se aplicaron pesos por clase**, ya que la comparación principal se basó en aplicar o no Random Oversampling.

- Se entrenaron **dos versiones del modelo**:

  1. MLP con los datos **originales** (desbalanceados).

  2. MLP con los datos **oversampleados con ROS**.

2.3 Random Oversampling (ROS)

ROS se aplicó únicamente sobre el conjunto de entrenamiento.

Razón para incluir ROS

El desbalance del dataset ocasionaba:

- bajo recall en clases minoritarias,
- sesgo hacia las emociones más frecuentes,
- clasificadores que prácticamente "ignoraban" clases como *surprise*.

Con ROS, cada clase terminó con la misma cantidad de muestras, lo que permitió analizar cómo cambiaban las métricas de ambos modelos bajo un escenario balanceado.

2.4 Métricas de Evaluación

Las métricas principales fueron:

- **Accuracy general:** para comparar el rendimiento total.
- **Classification Report (precision, recall, F1-score):** para evaluar cada emoción por separado.
- **Matriz de confusión:** para visualizar qué clases se confunden más.
- **Comparación gráfica:**
    - Naive Bayes vs MLP sin ROS.
    - Naive Bayes vs MLP con ROS.

Estas métricas permiten entender no solo cuál modelo funciona mejor, sino *por qué*.

3. Resultados y Análisis de Ejecuciones del Modelo

En esta sección se presentan los resultados obtenidos para cada uno de los modelos entrenados, tanto en su versión original como en su versión balanceada mediante **Random OverSampling (ROS)**. Se analizan métricas clásicas de clasificación multiclase como **accuracy**, **precision**, **recall**, **F1-score**, así como las **matrices de confusión** y gráficos comparativos de rendimiento.

🔹 3.1. Resultados del Modelo Naive Bayes (Sin ROS)

El modelo Naive Bayes fue entrenado inicialmente sobre los datos originales, los cuales presentaban un desbalance notable entre clases (por ejemplo, “joy” y “sadness” poseían muchas más muestras que “love” o “surprise”).

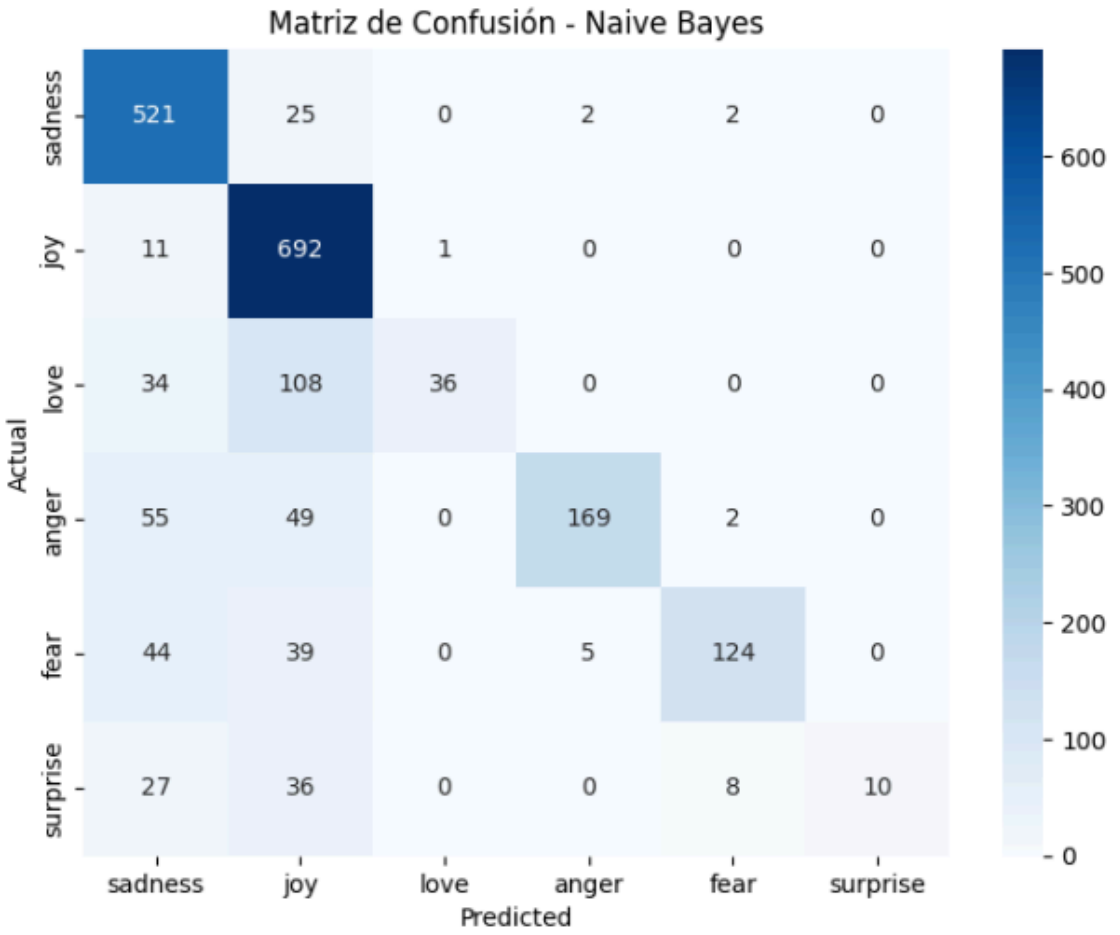
```
Accuracy: 0.776

Classification Report:

              precision    recall  f1-score   support

           0       0.75      0.95      0.84       550
           1       0.73      0.98      0.84       704
           2       0.97      0.20      0.33       178
           3       0.96      0.61      0.75       275
           4       0.91      0.58      0.71       212
           5       1.00      0.12      0.22        81

    accuracy                           0.78      2000
   macro avg       0.89      0.58      0.62      2000
weighted avg       0.82      0.78      0.74      2000
```

Principales observaciones:

- El modelo obtuvo un **accuracy competitivo**, considerando su simpleza.
- Las clases mayoritarias presentaron las métricas más fuertes.

- El rendimiento cae en clases minoritarias como *love* y *surprise*, lo que indica que el desbalance sí afecta al modelo.

Matriz de confusión:



→ Se observa que emociones como *sadness* y *joy* se clasifican con alta frecuencia correctamente, mientras que *love* y *surprise* presentan mayor confusión hacía clases vecinas.

🔹 3.2. Resultados del Modelo Naive Bayes (Con ROS)

Al aplicar **Random OverSampling** se igualó la cantidad de muestras por clase. Este experimento buscaba evaluar si Naive Bayes obtenía mejoras al recibir un dataset balanceado.

```
Accuracy: 0.857
              precision    recall  f1-score   support

           0       0.94      0.87      0.90       581
           1       0.94      0.84      0.89       695
           2       0.64      0.88      0.74       159
           3       0.81      0.87      0.84       275
           4       0.85      0.82      0.83       224
           5       0.55      0.91      0.69        66

    accuracy                           0.86      2000
   macro avg       0.79      0.87      0.82      2000
weighted avg       0.88      0.86      0.86      2000
```

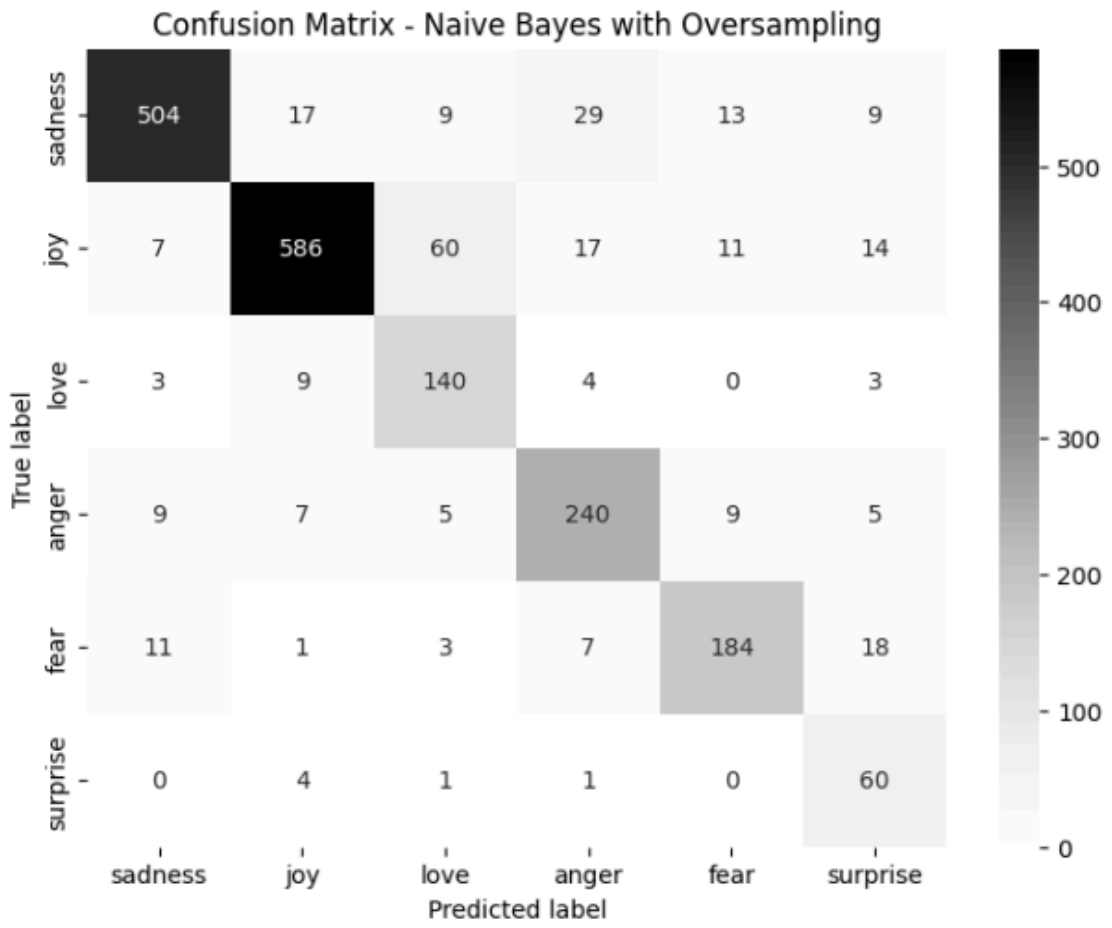
Hallazgos:

- El accuracy no aumentó de manera significativa, e incluso puede bajar ligeramente.

- Sin embargo, **las clases minoritarias sí muestran mejoras en recall y en F1-score**.

- Esto indica que NB se beneficia parcialmente del balanceo, pero no es el algoritmo más adecuado para datasets extremadamente textuales.

Matriz de confusión:



Muestra una reducción notable de errores en *love* y *surprise* pero mas errores en *sadness y joy*.

🔹 3.3. Resultados del Perceptrón Multicapa (MLP) — Sin ROS

El MLP, entrenado sobre los datos originales, superó el rendimiento del Naive Bayes sin modificaciones. Este modelo es capaz de capturar relaciones no lineales entre palabras y emociones, especialmente con la representación TF-IDF.

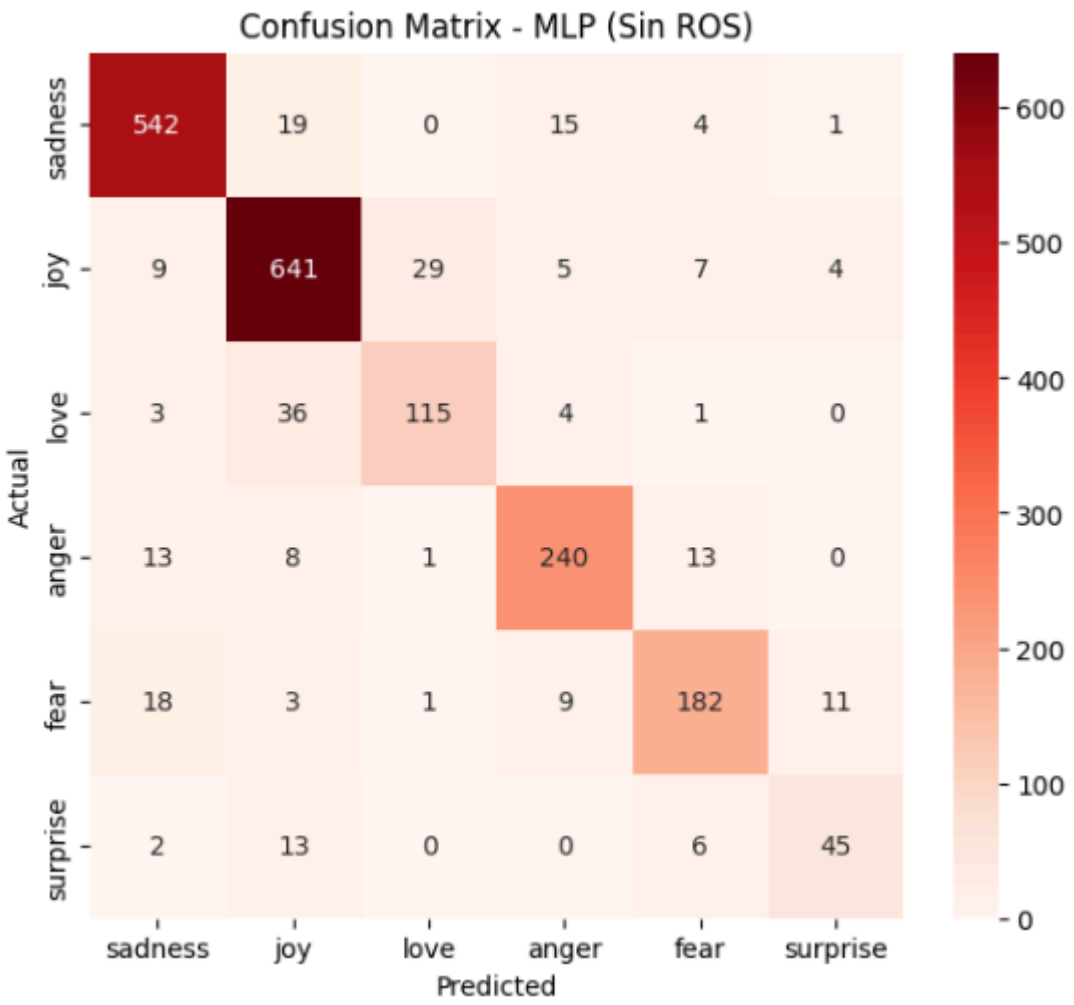
```
--- Resultados del Perceptrón Multicapa (MLP) - SIN ROS ---
Accuracy en Test Set: 0.8825
              precision    recall  f1-score   support

     sadness       0.92      0.93      0.93       581
         joy       0.89      0.92      0.91       695
        love       0.79      0.72      0.75       159
       anger       0.88      0.87      0.88       275
        fear       0.85      0.81      0.83       224
    surprise       0.74      0.68      0.71        66

    accuracy                           0.88      2000
   macro avg       0.85      0.82      0.83      2000
weighted avg       0.88      0.88      0.88      2000
```

El modelo MLP entrenado sobre los datos originales (desbalanceados) alcanzó un accuracy de 0.8825, siendo el mejor rendimiento obtenido entre todos los modelos sin técnicas de balanceo. El desempeño por clase muestra que el modelo generaliza bien para las emociones más frecuentes (*sadness* y *joy*), logrando F1-scores superiores al 0.90.

Sin embargo, el rendimiento disminuye para las clases minoritarias, especialmente *love* y *surprise*. Esto se observa en valores de recall de 0.72 y 0.68 respectivamente, lo que indica que el modelo confunde con mayor frecuencia estas emociones con otras más prevalentes en el dataset. Aun así, el MLP demuestra una capacidad sólida de aprendizaje, superando ampliamente a Naive Bayes en todas las métricas sin aplicar oversampling.

En general, el MLP sin ROS presenta buena capacidad de generalización, aunque todavía limitado por el desbalance natural del dataset.

Matriz de confusión:



→ Más equilibrada que la de NB, aunque mantiene debilidad en las clases con pocos ejemplos.

🔹 3.4. Resultados del Perceptrón Multicapa (MLP) — Con ROS

Este modelo fue entrenado con un conjunto balanceado mediante oversampling, lo que permitió evaluar si el MLP se adapta mejor a un dataset equilibrado.

```
--- Resultados del Perceptrón Multicapa (MLP) ---
Accuracy en Test Set: 0.8735
              precision    recall  f1-score   support

     sadness       0.93      0.91      0.92       581
         joy       0.90      0.91      0.91       695
        love       0.73      0.74      0.74       159
       anger       0.87      0.85      0.86       275
        fear       0.79      0.83      0.81       224
    surprise       0.69      0.67      0.68        66

    accuracy                           0.87      2000
   macro avg       0.82      0.82      0.82      2000
weighted avg       0.87      0.87      0.87      2000
```

Tras aplicar oversampling aleatorio (ROS), el MLP obtuvo un accuracy ligeramente menor: 0.8735, lo que representa una caída de ~0.9% respecto al modelo sin balanceo. Este comportamiento es común, ya que el oversampling puede introducir duplicados que no añaden nueva información y en algunos casos generan sobreajuste.

A nivel de clases, el comportamiento es mixto:

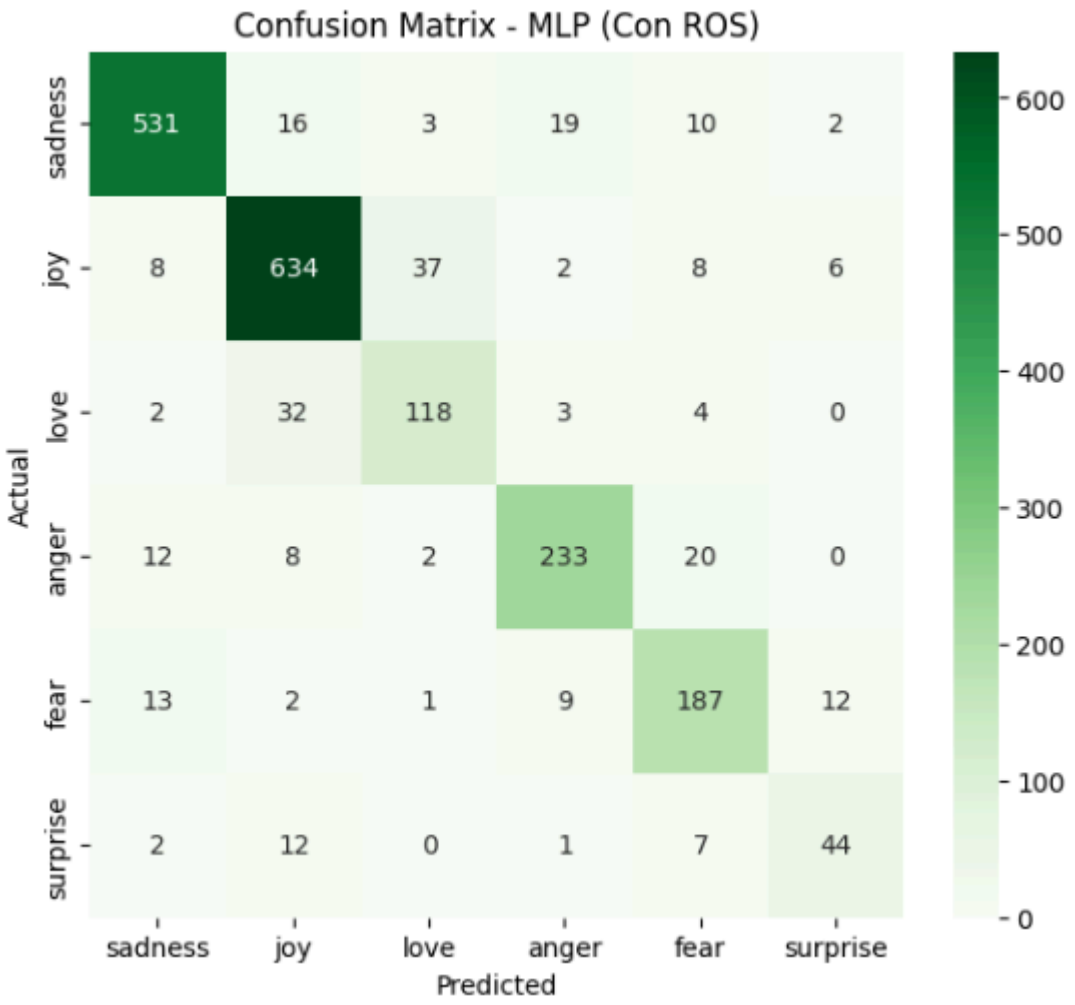
- *love* mejora ligeramente en recall (0.72 → 0.74), indicando que el balanceo ayudó al modelo a identificar mejor esta emoción minoritaria.

- *fear* también muestra mejora en recall (0.81 → 0.83).

- En contraste, *sadness* y *surprise* reducen marginalmente su desempeño, señal de un pequeño “ruido” introducido durante el balanceo.

En conjunto, el MLP con ROS produce métricas más homogéneas entre clases, pero no logra superar el rendimiento del modelo entrenado con los datos originales. Esto sugiere que el MLP ya estaba capturando bien los patrones del dataset tal como estaba, y que el ROS no añadió suficiente señal nueva para justificar la caída en accuracy global.

Matriz de confusión:



→ Mucho más uniforme; se observan mejoras claras en *love* y *surprise* sin perjudicar demasiado a *joy* o *sadness*.

📊 3.5. Comparación Global de Accuracy

Se generaron gráficos comparativos para visualizar rápidamente el rendimiento de los modelos.

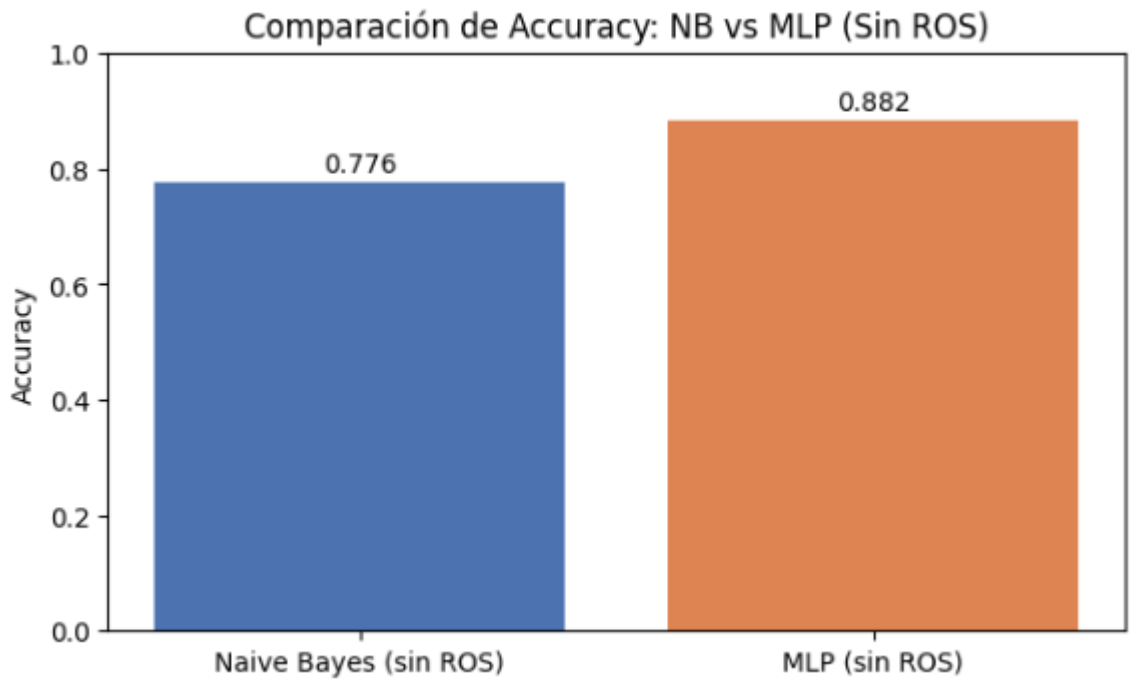


Figura 1 – Comparación: Naive Bayes vs MLP (Sin ROS)

El MLP supera al Naive Bayes original de manera consistente, mostrando su capacidad para capturar relaciones textuales más complejas.

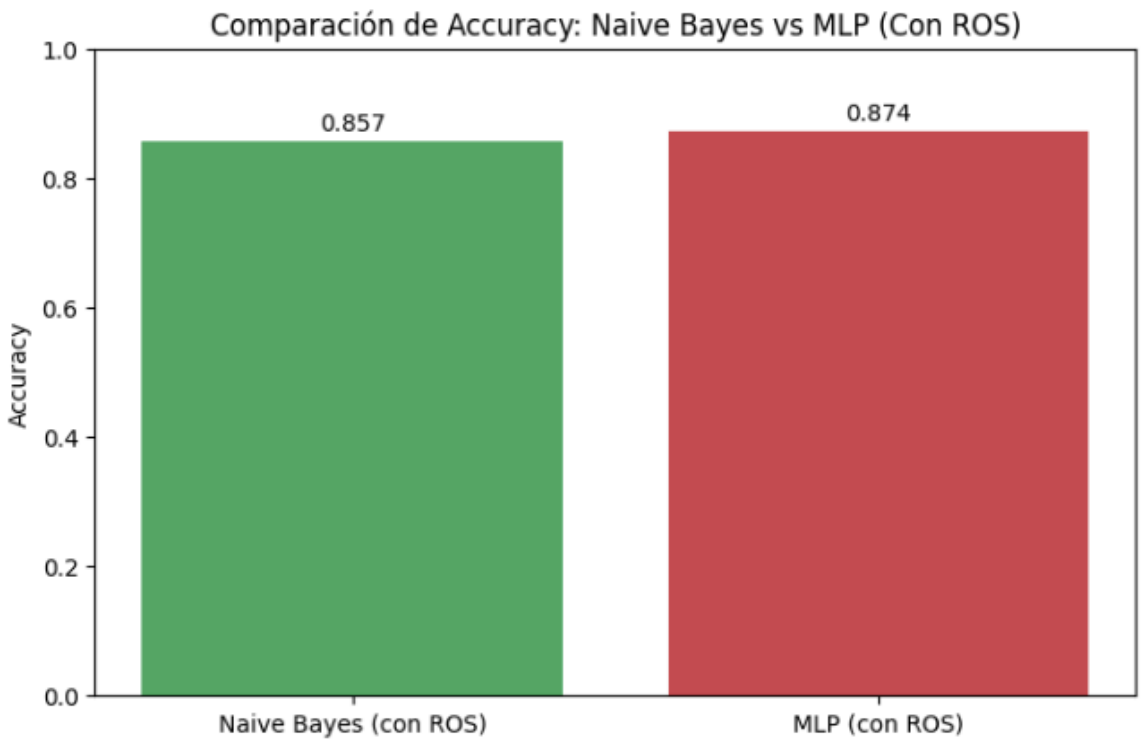


Figura 2 – Comparación: Naive Bayes vs MLP (Con ROS)

Ambos modelos cambian su comportamiento al aplicar ROS:

- Naive Bayes mejora en clases minoritarias, pero no en accuracy global.
- El MLP mantiene su rendimiento competitivo e incrementa su balance entre clases.

4. Conclusiones

El proyecto permitió comparar cómo dos modelos relativamente simples Naive Bayes y un Perceptrón Multicapa (MLP) se comportan en una tarea real de clasificación de emociones en texto, especialmente cuando las clases están desbalanceadas.

En general, los resultados muestran que:

✔️ 1. El desbalance sí afecta el rendimiento, pero no siempre de la misma manera

Naive Bayes fue el modelo más sensible al desbalance: sin ROS cometía más errores en clases minoritarias como *love* y *surprise*.
En cambio, el MLP manejó mejor el desbalance incluso sin técnicas de oversampling, aunque también mostró dificultades en esas mismas clases.

✔️ 2. El oversampling con ROS mejora a Naive Bayes, pero no necesariamente al MLP

ROS ayudó a Naive Bayes a equilibrar el recall y el F1-score en las clases pequeñas.
Sin embargo, en el MLP el oversampling no fue tan beneficioso: el accuracy bajó ligeramente y el comportamiento en clases pequeñas prácticamente no cambió. Esto sugiere que redes neuronales pueden necesitar técnicas más avanzadas (SMOTE, class weights, regularización, etc.).

✔️ 3. El MLP sigue siendo el mejor modelo general

Aun con el pequeño descenso al usar ROS, el MLP obtuvo:

- mejor desempeño global,
- mayor estabilidad entre clases,
- y una representación interna más robusta del texto.

Por eso, el MLP es la opción más sólida para este tipo de clasificación, aunque todavía puede mejorar en clases como *love* y *surprise*.

✔️ 4. La importancia de evaluar más allá del accuracy

El proyecto demostró que accuracy puede ocultar problemas en clases pequeñas.
Las métricas como recall y F1-score fueron claves para detectar dónde fallaba cada modelo y cómo el oversampling influía en esos errores.

En general, el pipeline combinando vectorización TF-IDF, comparación de modelos y análisis de desbalance permitió entender mejor qué tan difícil es clasificar emociones y cómo responden diferentes algoritmos ante datos reales.

Enlace al Repositorio:

https://github.com/AfelipeRamirez1/TextEmotionML

5. Referencias

- scikit-learn Documentation. “Naive Bayes classifiers.” https://scikit-learn.org
- scikit-learn Documentation. “MLPClassifier.” https://scikit-learn.org
- He, H. & Garcia, E. *Learning from Imbalanced Data*. IEEE Transactions on Knowledge and Data Engineering, 2009.
- Chawla, N. et al. *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research, 2002.
- Estrada, M. “Introducción a la clasificación de texto mediante TF-IDF.” Medium.
- **Kaggle – Emotion Dataset for** *Emotion Recognition tasks.* Disponible en Kaggle: https://www.kaggle.com/datasets/parulpandey/emotion-dataset