Tutorial: Análisis de Sentimientos en Tweets y Clasificación

1. Configuración del Entorno

Para este proyecto, se utilizarán varias bibliotecas de Python relacionadas con el análisis de texto y modelado. Algunas de las bibliotecas principales que debes instalar incluyen:

- **nltk**: Para procesamiento de lenguaje natural.
- **emoji**: Para manejar y contar emojis.
- **textblob** y **vaderSentiment**: Para realizar el análisis de sentimientos.
- **pandas** y **numpy**: Para manipulación de datos.
- **matplotlib** y **seaborn**: Para visualización.
- **scikit-learn**: Para dividir los datos y entrenar modelos.

Instala las bibliotecas necesarias usando `pip` en tu entorno de trabajo.

---

2. Carga y Preprocesamiento de Datos

2.1 Cargando los Datos

En este proyecto, se trabaja con un conjunto de datos de tweets que contienen anotaciones de sentimientos (0 = negativo, 1 = positivo). Estos datos se cargarán desde un archivo CSV utilizando `pandas`.

2.2 Limpieza del Texto

Es importante limpiar los tweets eliminando elementos no esenciales como:

- **URLs**
- **Menciones a otros usuarios (@)**
- **Emojis**
- **Signos de puntuación y números**

Se debe escribir una función de preprocesamiento que elimine estos elementos y también convierta el texto a minúsculas para evitar duplicados.

	text	cleaned_text
0	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!	is upset that he cant update his facebook by texting it and might cry as a result school today also blah
1	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds	i dived many times for the ball managed to save 50 the rest go out of bounds
2	my whole body feels itchy and like its on fire	my whole body feels itchy and like its on fire
3	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.	no its not behaving at all im mad why am i here because i cant see you all over there
4	@Kwesidei not the whole crew	not the whole crew

2.3 Generación de Características

Una vez que el texto ha sido limpiado, se generan varias características adicionales a partir de los tweets. Algunas de las características que puedes implementar incluyen:

1. **Longitud del tweet**: Contar el número de caracteres.
2. **Conteo de emojis**: Contar cuántos emojis contiene cada tweet.
3. **Conteo de signos de exclamación e interrogación**: Estos pueden ser útiles para detectar emociones intensas.
4. **Proporción de palabras en mayúsculas**: Para identificar tweets en los que se hace "énfasis".
5. **Conteo de palabras**: Número total de palabras en el tweet.
6. **Conteo de stopwords**: Palabras comunes como "the" o "is" que no aportan mucho significado.
7. **Subjetividad**: Usar librerías como **TextBlob** o **VADER** para obtener la subjetividad del texto.
8. **Proporción de palabras repetidas**: Esta métrica puede ser útil para detectar patrones de repetición en el texto.
9. **Entropía del texto**: Una medida de la diversidad de caracteres en el texto.
10. **Sarcasmo del texto**: Usar librerías como **TextBlob** o **VADER** para obtener el sarcasmo del texto.

---

3. División del Conjunto de Datos

Una buena práctica es dividir los datos en conjuntos de:

- **Entrenamiento**: Para ajustar los modelos.
- **Prueba**: Para evaluar el desempeño del modelo.
- **Validación**: Para ajustar los hiperparámetros.

Se recomienda dividir los datos en un 70% para entrenamiento y 30% para prueba, y luego dividir el 30% restante en prueba y validación.

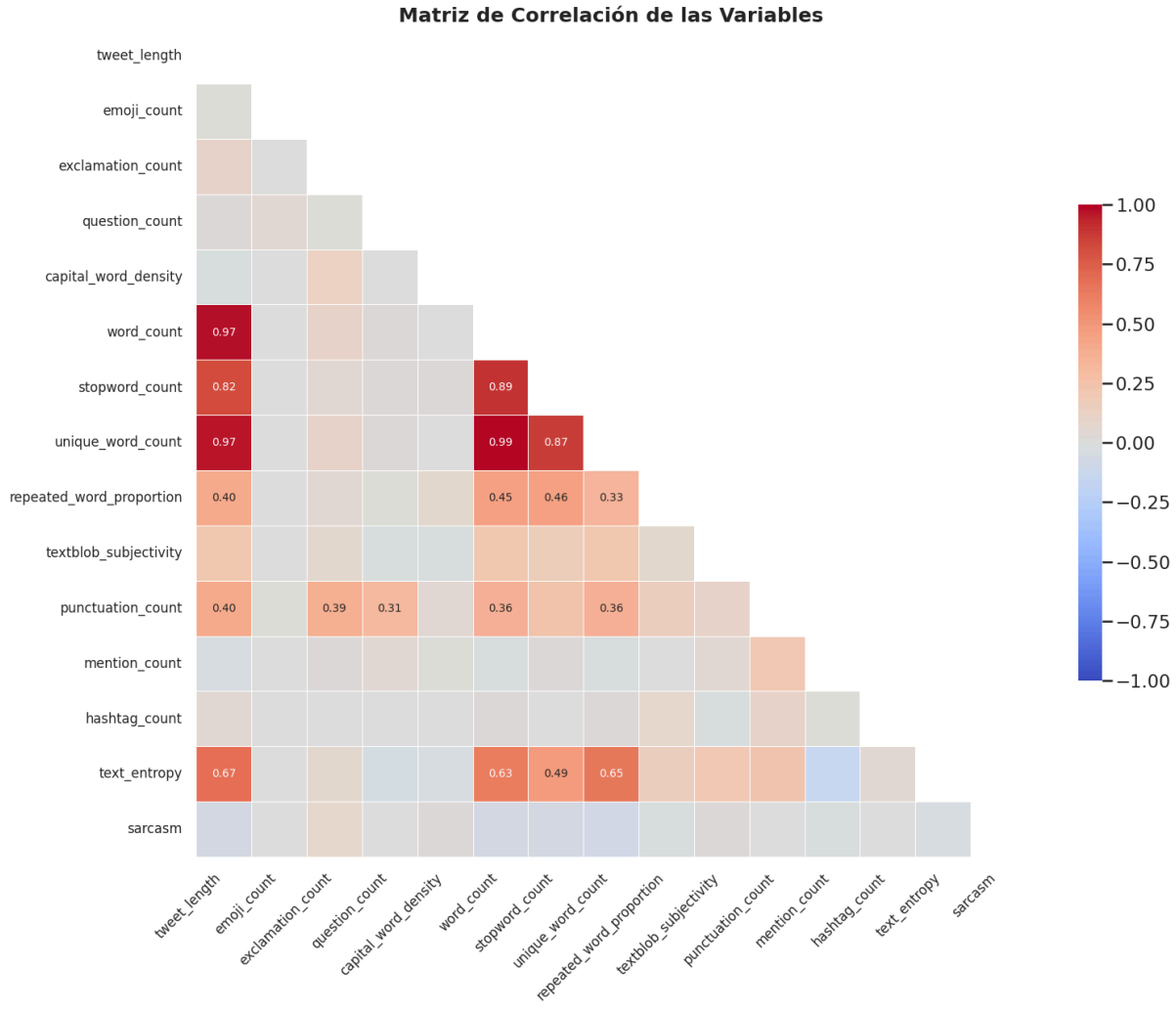
---

4. Visualización de los Datos

Antes de entrenar cualquier modelo, es útil visualizar la distribución de los datos. Algunos gráficos sugeridos:

- **Gráfico de barras** para visualizar la distribución de las clases (negativo, positivo).
- **Gráficos de densidad** para ver la dispersión de algunas de las características generadas, como la longitud del tweet o la polaridad del sentimiento.
- **Correlación de las variables** para poder eliminar variables que presentan correlación muy alta.

Esto ayudará a identificar si hay algún desbalance en las clases o si existen características podrían ser más relevantes para el modelo.



5. Modelado

Después de haber preprocesado los datos y generado las características, es momento de entrenar un modelo. Algunas opciones que puedes explorar son:

- **XGBoost**: Un potente modelo basado en árboles de decisión.
- **LightGBM:** Una opción más light que el XGBoost optimizando los tiempos

Al entrenar los modelos, asegúrate de:

- **Ajustar los hiperparámetros** utilizando validación cruzada y **Halving Grid Search** por su rápido procesamiento.
- **Evaluar el desempeño** del modelo en el conjunto de validación usando métricas como **accuracy**, **precision**, **recall** y **f1-score**.

```python
from sklearn.experimental import enable_halving_search_cv  # Import necesario para Halving
from sklearn.model_selection import HalvingGridSearchCV
import xgboost as xgb

# Definir el modelo base de XGBoost
xgb_model = xgb.XGBClassifier(eval_metric='')

# Definir los hiperparámetros a evaluar
param_grid = {
    'n_estimators': [],
    'max_depth': [],
    'learning_rate': [],
    'lambda': [],
    'colsample_bytree': [],
}

# Crear el Halving Grid Search con validación cruzada
halving_search = HalvingGridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    return_train_score=True,
    factor=,  # Reduce el número de evaluaciones en cada ronda
    random_state=42,
    cv=,  # Validación cruzada en 3 folds
    verbose=5
)

# Entrenar el modelo con Halving Grid Search
halving_search.fit(X_train, y_train)

# Ver los mejores hiperparámetros
print(f"Mejores hiperparámetros: {halving_search.best_params_}")
```

---

6. Evaluación del Modelo

Una vez entrenado el modelo, debes evaluar su desempeño en el conjunto de prueba. Las métricas sugeridas incluyen:

- **Accuracy**: Para ver qué porcentaje de las predicciones fueron correctas.
- **Precision y Recall**: Especialmente útiles si hay clases desbalanceadas.
- **Matriz de confusión**: Para visualizar qué tan bien el modelo predijo cada clase.

Comparar los rendimientos del XGboost como del LightGBM.

---

8. Conclusiones y Próximos Pasos

Este proyecto demuestra cómo el preprocesamiento de texto y la ingeniería de características pueden mejorar el rendimiento de los modelos de clasificación en un problema de análisis de sentimientos. Puedes probar tu modelo con tus propios tweets!.