**Resumen**

El objetivo principal de este estudio fue desarrollar un modelo de clasificación capaz de identificar de manera precisa al autor de un texto dado. En otras palabras, buscamos construir una herramienta que, al analizar un texto, pueda determinar quién lo escribió entre un grupo de autores predefinidos. Utilizando un corpus de textos de diversos autores tomado de [**Victorian Era Authorship Attribution Data Set**](https://archive.ics.uci.edu/ml/datasets/Victorian+Era+Authorship+Attribution), se entrenaron y evaluaron diversos modelos de aprendizaje automático, incluyendo árboles de decisión, bosques aleatorios, máquinas de vectores de soporte, redes neuronales (MLP) y XGBoost. Los resultados mostraron que la elección del modelo óptimo depende de factores como el tamaño y características del corpus, así como de la complejidad del texto. El preprocesamiento del texto, la sintonización de hiperparámetros y el uso de modelos de conjunto resultaron cruciales para mejorar la precisión. En general, los modelos de clasificación sencillos, especialmente la Regresión Logística, demostraron ser muy poderosas para esta tarea, pero requieren mayores recursos computacionales.

**Descripción del corpus: Número y tipo de atributos, numero de instancias**

**## FALTA**

**Preprocesameinto de la información:**

La función presentada realiza una serie de transformaciones sobre un texto con el objetivo de prepararlo para su procesamiento por modelos de aprendizaje automático, específicamente para tareas de procesamiento de lenguaje natural (NLP).

**Pasos de Preprocesamiento:**

1. **Minusculización:** Convierte todo el texto a minúsculas para uniformizar y simplificar el procesamiento.
2. **Eliminación de acentos:** Normaliza el texto eliminando los acentos y caracteres especiales, facilitando la comparación entre palabras.
3. **Eliminación de puntuación:** Quita todos los signos de puntuación, ya que en muchos casos estos no son relevantes para el significado de las palabras.
4. **Remoción de números:** Elimina los números del texto, ya que pueden no ser relevantes para tareas como la clasificación de sentimientos o la temática de un documento.
5. **Tokenización:** Divide el texto en palabras individuales (tokens), que son las unidades básicas de análisis en NLP.
6. **Eliminación de stop words:** Quita las palabras vacías (stop words) como "the", "and", "of", que son muy frecuentes pero aportan poco significado al texto.

La función utilizada cumple con los requisitos de una buena limpieza y preprocesamiento de texto:

* **Pertinencia:** Las técnicas aplicadas son relevantes para preparar el texto para modelos de NLP.
* **Completitud:** Cubre los pasos de preprocesamiento más comunes y necesarios.
* **Eficiencia:** El código es conciso y eficiente.
* **Adaptabilidad:** La función puede ser fácilmente modificada para incluir otros pasos de preprocesamiento si fuera necesario.

**Estamos usando TFIDF porque:**

El TF-IDF (Frecuencia de Término-Frecuencia Inversa de Documento) es una técnica de ponderación de términos que ha demostrado ser sumamente útil en el procesamiento del lenguaje natural y en la recuperación de información. Su principal ventaja radica en su capacidad para asignar mayor relevancia a los términos que son más distintivos de un documento en comparación con el corpus completo.

Al ponderar los términos de esta manera, TF-IDF permite a los modelos de machine learning identificar las palabras clave más relevantes dentro de un texto. Esto resulta especialmente útil en tareas como la clasificación de documentos, la búsqueda de información y la extracción de características. Además, al considerar la frecuencia inversa de documento, TF-IDF ayuda a reducir el peso de términos muy frecuentes pero poco informativos, como las stop words.

En resumen, el uso de TF-IDF ofrece varias ventajas significativas:

* **Mayor relevancia de términos:** Al ponderar los términos en función de su frecuencia en un documento y en el corpus completo, TF-IDF permite identificar las palabras más distintivas y relevantes.
* **Reducción de ruido:** La frecuencia inversa de documento ayuda a reducir el peso de los términos muy frecuentes pero poco informativos, lo que mejora la calidad de la representación vectorial de los documentos.
* **Versatilidad:** TF-IDF puede ser utilizado en una amplia variedad de tareas de procesamiento del lenguaje natural, desde la clasificación de textos hasta la recuperación de información.
* **Interpretabilidad:** Los pesos asignados a los términos por TF-IDF pueden ser interpretados de manera intuitiva, lo que facilita la comprensión de los resultados de los modelos.

En conjunto, estas características hacen de TF-IDF una técnica fundamental en el campo del procesamiento del lenguaje natural.

**Proceso de optimización**

Se crea la función train\_cv\_models que implementa un proceso de búsqueda de hiperparámetros de dos etapas para entrenar un modelo de clasificación. Su objetivo principal es encontrar la mejor combinación de parámetros que maximice el rendimiento del modelo en un conjunto de datos dado.

**Primera Etapa:**

En la primera etapa, se crea un pipeline que combina varias etapas:

* **Vectorización TF-IDF:** Convierte el texto en una representación numérica utilizando la técnica TF-IDF, que pondera las palabras según su frecuencia en el documento y en el corpus completo.
* **Oversampling:** Corrige el desbalance de clases en los datos de entrenamiento, duplicando aleatoriamente las muestras de las clases minoritarias.
* **Clasificador:** Aplica el clasificador especificado (por ejemplo, una máquina de soporte vectorial, un bosque aleatorio) para realizar la clasificación.

A continuación, se realiza una búsqueda exhaustiva de hiperparámetros utilizando GridSearchCV con validación cruzada de 5 folds. Esto implica probar todas las posibles combinaciones de los valores de los parámetros especificados en param\_grid.

**Segunda Etapa:**

La segunda etapa se centra en afinar los hiperparámetros obtenidos en la primera etapa. Se crea un nuevo param\_grid donde los valores numéricos se ajustan en un rango del 10% alrededor de los valores óptimos encontrados en la primera etapa. Esto permite explorar un espacio de parámetros más reducido y encontrar una solución aún mejor.

**Función train\_final\_model:**

Una vez que se han encontrado los mejores hiperparámetros, la función train\_final\_model crea un modelo final utilizando estos parámetros. El modelo se entrena con todos los datos de entrenamiento y se evalúa en un conjunto de prueba independiente. Finalmente, se calculan métricas de evaluación como precisión, recall, F1-score y soporte para evaluar el rendimiento del modelo.

**Características clave:**

* **Búsqueda exhaustiva de hiperparámetros:** Garantiza que se explore un amplio espacio de parámetros para encontrar la mejor configuración.
* **Validación cruzada:** Permite evaluar el rendimiento del modelo en diferentes subconjuntos de los datos de entrenamiento, lo que reduce el riesgo de sobreajuste.
* **Oversampling:** Aborda el problema del desbalance de clases, mejorando el rendimiento del modelo en las clases minoritarias.
* **Múltiples etapas de búsqueda:** Permite afinar los hiperparámetros de manera más precisa.
* **Evaluación exhaustiva:** Calcula múltiples métricas de evaluación para obtener una visión completa del rendimiento del modelo.

Estas funciones proporcionan una forma sistemática de entrenar y evaluar modelos de clasificación, asegurando que se obtenga el mejor rendimiento posible en un conjunto de datos dado.

**Resultado de los modelos**

**Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja**

**FALTA COMPLETAR Y HACER ANALISIS**

**Conclusiones:**

**FALTA RESPONDER BASADO EN RESULTADOS**

1. **¿Cuáles modelos se desempeñaron mejor y peor?**
2. **¿Cuál modelo tuvo un comportamiento llamativo? ¿Por qué creen que sucedió esto?**
3. **De acuerdo con los resultados arrojados por su mejor modelo, ¿les parece que éste está listo para ser entregado para su uso en la organización?**
4. **¿Cuáles elementos adicionales vale la pena resaltar acerca del proceso, los resultados o los modelos?**

La evaluación de la disparidad en los datos es un paso fundamental en el desarrollo de modelos de machine learning, especialmente en el contexto de la clasificación de textos. Como se evidencia en las funciones de preprocesamiento y entrenamiento de modelos presentadas, existen varias etapas en las que esta disparidad puede influir significativamente en los resultados obtenidos.

**Consecuencias de la Desigualdad en los Datos**

* **Sesgos en los modelos:** Si los datos de entrenamiento están desbalanceados, es decir, si hay una representación desproporcionada de ciertas clases o categorías, el modelo tenderá a favorecer la predicción de la clase mayoritaria. Esto puede llevar a resultados sesgados y discriminatorios, especialmente en aplicaciones sensibles como la toma de decisiones en sistemas legales o de recursos humanos.
* **Pérdida de precisión:** La disparidad en los datos puede reducir la precisión del modelo en las clases minoritarias, ya que el modelo tiene menos ejemplos para aprender a reconocerlas. Esto puede ser un problema si estas clases son de particular interés para la aplicación en cuestión.
* **Dificultad en la generalización:** Modelos entrenados con datos sesgados pueden tener dificultades para generalizar a nuevos datos que no estén representados de manera adecuada en el conjunto de entrenamiento.

**Distribución de los Datos y Desempeño del Modelo**

La distribución de los datos influye directamente en el desempeño del modelo de varias maneras:

* **Tamaño de las clases:** Un desbalance en el número de muestras por clase puede dificultar el aprendizaje del modelo, especialmente para las clases minoritarias.
* **Características de las clases:** La distribución de las características dentro de cada clase puede afectar la capacidad del modelo para discriminar entre ellas. Por ejemplo, si las clases están muy solapadas, el modelo puede tener dificultades para encontrar una frontera de decisión clara.
* **Correlación entre características:** La presencia de correlaciones entre las características puede afectar la capacidad del modelo para aprender patrones relevantes y puede llevar a problemas de multicolinealidad.

**Análisis de Resultados y Causas**

Al analizar los resultados de un modelo, es importante considerar las siguientes causas posibles:

* **Selección del modelo:** La elección del modelo de clasificación puede tener un impacto significativo en el rendimiento. Algunos modelos son más adecuados para ciertos tipos de datos y problemas que otros.
* **Hiperparámetros:** Los valores de los hiperparámetros del modelo pueden influir en su capacidad de generalización y precisión.
* **Preprocesamiento de datos:** La calidad del preprocesamiento de los datos puede afectar significativamente el rendimiento del modelo.
* **Desbalance de clases:** Un desbalance en los datos de entrenamiento puede llevar a resultados sesgados.
* **Características de los datos:** La naturaleza de los datos, como la dimensionalidad y la presencia de ruido, puede afectar el rendimiento del modelo.

En resumen, la disparidad en los datos es un problema común en el aprendizaje automático que puede tener consecuencias importantes para el rendimiento y la fiabilidad de los modelos. Es fundamental realizar un análisis cuidadoso de los datos y seleccionar las técnicas de preprocesamiento y entrenamiento adecuadas para mitigar estos efectos y obtener modelos más justos y precisos.