Proyecto de Clase: Clasificación de Textos para Problemas Reales

Descripción del Proyecto:

Abordar un problema práctico de clasificación de textos de su interés. Este proyecto incluye la definición del problema, exploración de datos, implementación de modelos de clasificación y análisis de resultados.

Requerimientos:

1. **Selección y planteamiento del problema:**
    - Cada equipo debe proponer un problema de clasificación de textos. Ejemplos sugeridos:
        - Determinar si una publicación en redes sociales contiene opiniones negativas.
        - Clasificar correos electrónicos como spam o no spam.
        - Detectar noticias falsas.
        - Identificar emociones en textos (alegría, tristeza, enojo, etc.).
    - El equipo debe justificar su elección, explicando:
        - Relevancia del problema.
        - Potenciales aplicaciones prácticas.
    - Definir claramente las clases a predecir (binarias o múltiples) y las métricas para evaluar el modelo.

2. **Análisis descriptivo:**
    - Realizar un análisis exploratorio del dataset, incluyendo:
        - Estadísticas descriptivas (longitud de textos, frecuencia de palabras, etc.).
        - Visualización de datos (nubes de palabras, histogramas de clases).
        - Patrones relevantes o hallazgos iniciales.

3. **Búsqueda y selección del dataset:**
    - Deben encontrar un dataset relacionado con el problema seleccionado. Fuentes sugeridas:
        - Kaggle.
        - UCI Machine Learning Repository.
        - Conjuntos de datos públicos extraídos de APIs (por ejemplo, Twitter, Reddit).

- Limpiar y preprocesar el dataset (tokenización, eliminación de stopwords, etc.).

4. **Implementación de dos modelos distintos:**

   - Seleccionar dos modelos de clasificación de textos. Ejemplos:
     - Naive Bayes y Logistic Regression.
     - Random Forest y Support Vector Machine.
     - Un modelo clásico y un modelo basado en embeddings (e.g., usando Word2Vec o TF-IDF con Logistic Regression).
   - Los modelos deben implementarse como clases en Python, siguiendo un estilo estructurado y bien documentado.

5. **Explicación teórica:**

   - Incluir una breve explicación teórica de cada modelo en el reporte, citando referencias confiables (libros, artículos, capítulos).
   - Relacionar la implementación en Python con la teoría descrita.

6. **Evaluación de los modelos:**

   - Entrenar ambos modelos con el dataset seleccionado.
   - Evaluarlos usando métricas como precisión, recall, F1-score, y matriz de confusión.
   - Comparar los resultados de los modelos y discutir sus fortalezas y limitaciones.

7. **Entregables:**

   - **Análisis descriptivo:** Visualizaciones y conclusiones sobre el dataset.
   - **Implementación en Python:** Código bien documentado y funcional de las clases para ambos modelos.
   - **Reporte final:** Documento que incluya:
     - Descripción del problema.
     - Análisis descriptivo y exploratorio.
     - Explicación de los modelos seleccionados.
     - Comparación de resultados y conclusiones.
   - **Presentación oral:** Resumen del proyecto, duración máxima: 20 minutos.

Evaluación:

- **Planteamiento del problema y justificación (10%)**
- **Análisis descriptivo y exploración de datos (20%)**
- **Calidad de las implementaciones (30%)**
- **Reporte final (20%)**
- **Presentación oral (20%)**

Fecha de Entrega: segunda semana después de regresar de vacaciones

Recursos de apoyo:

- An Introduction to Statistical Learning para modelos clásicos.
- Artículos sobre clasificación de textos en arXiv.
- Guías de preprocesamiento de textos y modelado en Python (e.g., Scikit-learn, NLTK, SpaCy).