

RECUPERACIÓN AVANZADA DE LA INFORMACIÓN 2024.

CLASIFICACIÓN DE DOCUMENTOS CON REDUCCIÓN DIMENSIONAL

Marta de Castro Leira



Departamento de Informática y Automática
Universidad de Salamanca

Resumen (Español)

Este informe aborda el problema de clasificación automática de documentos mediante el uso de diversas técnicas de aprendizaje automático. A partir de una colección de 18.000 documentos extraídos de las conferencias CLEF, se aplicaron métodos de preprocesamiento como la conversión a minúsculas, eliminación de palabras vacías y lematización con Snowball Stemmer en español. El objetivo principal fue comparar el rendimiento de varios algoritmos de clasificación (Naive Bayes, Rocchio, k-NN y SVM), evaluando su precisión, recall y f1-score. Además, se probó la reducción dimensional utilizando métodos como Chi cuadrado, información mutua y ANOVA F-text para determinar si mejora los resultados en términos de precisión y tiempo de cómputo. Los resultados obtenidos mostraron que los métodos de clasificación aplicados lograron una precisión promedio de alrededor del 92%, destacando el buen rendimiento en categorías como Política, Deportes y Economía.

Summary (English)

This report addresses the problem of automatic document classification using various machine learning techniques. A collection of 18,000 documents from the CLEF conferences was used, and preprocessing methods such as lowercase conversion, stopword removal, and Snowball Stemmer lemmatization in Spanish were applied. The main objective was to compare the performance of several classification algorithms (Naive Bayes, Rocchio, k-NN, and SVM), evaluating metrics like accuracy, recall, and f1-score. Additionally, dimensionality reduction methods like Chi-square, mutual information, and ANOVA F-text were tested to determine whether they improve results in terms of accuracy and computation time. The results showed that the applied classification methods achieved an average accuracy of around 92%, with strong performance in categories such as Politics, Sports, and Economics.

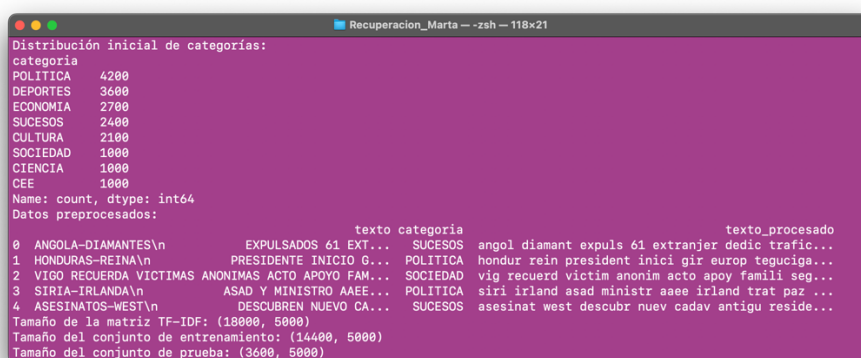
ÍNDICE

1. Introducción	5
2. Metodología	5
2.2 Preprocesamiento de los Datos	5
2.2. Representación TF-IDF y División de Datos.....	6
2.3. Reducción Dimensional	6
2.4. Clasificadores Utilizados	6
2.5. Métricas y Evaluación	6
2.6. Ejecución del Programa	7
3. Resultados	7
3.1. Comparación de las Matrices de Confusión.....	8
3.2. Análisis de la Precisión y el Tiempo Datos	10
3.3. Resultados Generados y Mejor Valor de k.....	13
4. Conclusiones.....	14
ANEXO: Código	15

1. INTRODUCCIÓN

El objetivo de este informe es clasificar automáticamente una colección de 18,000 documentos extraídos de conferencias CLEF, utilizando técnicas de aprendizaje automático y evaluando el impacto de métodos de reducción dimensional como Chi cuadrado (χ^2), información mutua y ANOVA F-text. Se busca analizar la precisión y eficiencia computacional de los clasificadores Naive Bayes (Multinomial y Bernoulli), Rocchio, k-NN y SVM.

El preprocesamiento textual incluye conversión a minúsculas, eliminación de palabras vacías y lematización con Snowball Stemmer en español. Los textos se vectorizan mediante TF-IDF y se dividen en conjuntos de entrenamiento (80%) y prueba (20%), empleando validación cruzada para garantizar robustez. Los resultados son evaluados a través de métricas como precisión, recall y F1-score, con el objetivo de identificar el método más eficaz para la clasificación textual.



```
Recuperacion_Marta -- zsh -- 118x21
Distribución inicial de categorías:
categoria      4200
POLITICA       3600
DEPORTES       2700
ECONOMIA       2400
SUCECOS        2100
CULTURA       1000
SOCIEDAD       1000
CIENCIA        1000
CEE            1000
Name: count, dtype: int64
Datos preprocesados:
   texto categoria      texto_procesado
0  ANGOLA-DIAMANTES\n  EXPULSADOS 61 EXT...  SUCECOS  angol diamant expuls 61 extranjer dedic trafic...
1  HONDURAS-REINA\n    PRESIDENTE INICIO G...  POLITICA  hondur rein president inici gir europ teguciga...
2  VIGO RECUERDA VICTIMAS ANONIMAS ACTO APOYO FAM...  SOCIEDAD  vig recuerd victim anonim acto apoy famili seg...
3  SIRIA-IRLANDA\n    ASAD Y MINISTRO AAEE...  POLITICA  siri irland asad ministr aeee irland trat paz ...
4  ASESINATOS-WEST\n    DESCUBREN NUEVO CA...  SUCECOS  asesinat west descubr nuev cadav antigu reside...
Tamaño de la matriz TF-IDF: (18000, 5000)
Tamaño del conjunto de entrenamiento: (14400, 5000)
Tamaño del conjunto de prueba: (3600, 5000)
```

2. METODOLOGÍA

Este trabajo sigue un enfoque estructurado para abordar la clasificación automática de documentos. A continuación, se detallan las etapas principales:

2.1. Preprocesamiento de los Datos

El preprocesamiento es crucial para limpiar y preparar los textos para su análisis. Las etapas realizadas fueron:

- **Conversión a minúsculas:** Uniformiza el texto para evitar distinciones entre mayúsculas y minúsculas.
- **Eliminación de caracteres no alfanuméricos:** Elimina puntuación y símbolos innecesarios.
- **Tokenización:** Divide los textos en palabras (tokens) usando espacios como delimitadores.
- **Eliminación de stopwords:** Retira palabras vacías comunes que no aportan significado al análisis (ej.: "el", "de", "y").
- **Lematización con Snowball Stemmer:** Reduce palabras a su raíz común, unificando variantes como "corriendo" y "correr".

Este preprocesamiento redujo el ruido en los datos, facilitando el enfoque en las características relevantes.

2.2. Representación TF-IDF y División de Datos

Los textos procesados se representaron numéricamente utilizando **TF-IDF** (Term Frequency-Inverse Document Frequency), asignando un peso a cada palabra basado en su frecuencia en el documento y su relevancia en la colección completa.

- **Limitación a 5000 características más relevantes:** Mejora la eficiencia al centrarse en términos con mayor impacto predictivo.
- **División del conjunto de datos:** Se dividieron en:
 - **Entrenamiento (80%)** para ajustar los modelos.
 - **Prueba (20%)** para evaluar su rendimiento.
- **Validación cruzada (Stratified K-Fold):** Utilizando 5 pliegues, garantiza robustez manteniendo proporciones originales de categorías en cada iteración.

2.3. Reducción Dimensional

Para mejorar la eficiencia computacional y reducir el espacio de características, se aplicaron tres técnicas de reducción dimensional, seleccionando las 1000 características más relevantes:

1. **Chi cuadrado (χ^2):** Evalúa la relación estadística entre palabras y categorías.
2. **Información mutua:** Mide la cantidad de información aportada por cada palabra para predecir una categoría.
3. **ANOVA F-text:** Identifica palabras que maximizan la variación entre categorías.

2.4. Clasificadores Utilizados

Se implementaron cuatro algoritmos de clasificación ampliamente utilizados en tareas de texto, seleccionados por su idoneidad:

- **Naive Bayes (Multinomial y Bernoulli):** Adecuados para texto; el primero modela frecuencias, mientras que el segundo evalúa presencia/ausencia de términos.
- **k-NN (k-Nearest Neighbors):** Clasifica según las categorías de los **k** vecinos más cercanos. Se optimizó el valor de **k** (1-20) mediante validación cruzada, determinándose que **k=1** es el óptimo.
- **Rocchio:** Calcula vectores centroides para cada categoría y clasifica según cercanía.
- **SVM (Support Vector Machine):** Encuentra un hiperplano óptimo de separación para las clases, destacando por su robustez ante ruido.

2.5. Métricas y Evaluación

Los modelos fueron evaluados con métricas estándar:

- **Precisión (Accuracy):** Proporción de documentos correctamente clasificados.
- **Matriz de confusión:** Analiza errores de clasificación entre categorías.
- **Tiempo de ejecución:** Se registró el tiempo necesario para entrenar y probar cada modelo, con y sin reducción dimensional.

2.6. Ejecución del Programa

El programa está diseñado para ser autónomo y fácil de usar en cualquier sistema operativo.

Estructura del Proyecto

El proyecto incluye los siguientes archivos:

Recuperacion_Marta/

recuperacion2.py	# Código fuente del proyecto
categorias18000.xml	# Archivo de datos con los documentos
requirements.txt	# Dependencias necesarias
setup.sh	# Script de automatización para Linux/macOS.
setup.bat	# Script de automatización para Windows.
Readme.md	# Instrucciones del proyecto.
informe.pdf	

Instrucciones de Ejecución

- **Windows:**

1. Navega al directorio del proyecto.
2. Ejecuta setup.bat.
3. El script crea un entorno virtual, instala dependencias y ejecuta el programa.
4. Los resultados (gráficos y CSV) estarán disponibles en el directorio.

- **Linux/macOS:**

1. Navega al directorio del proyecto.
2. Ejecuta `chmod +x setup.sh` y luego `./setup.sh`.
3. El script realiza los mismos pasos que en Windows.
4. Los resultados se generan automáticamente en el directorio del proyecto.

3. RESULTADOS

Antes de analizar los resultados, es importante aclarar el esquema de codificación utilizado para representar las distintas categorías de documentos en el conjunto de datos. Cada categoría se ha identificado con un número específico, asignado de la siguiente manera:

0. POLÍTICA
1. DEPORTES
2. ECONOMÍA
3. CULTURA
4. TECNOLOGÍA
5. SOCIEDAD
6. CIENCIA
7. CEE

Esta codificación ha sido implementada para simplificar la interpretación de los resultados, permitiendo representar cada categoría de manera concisa durante el proceso de clasificación automática y la visualización de métricas.

En esta sección se presentan y analizan los resultados obtenidos a través de los clasificadores, considerando tres dimensiones clave: las matrices de confusión, la precisión en validación y prueba, y los tiempos de ejecución. Se comparan los resultados obtenidos con y sin reducción dimensional para evaluar la eficacia y la eficiencia de los modelos en diferentes escenarios.

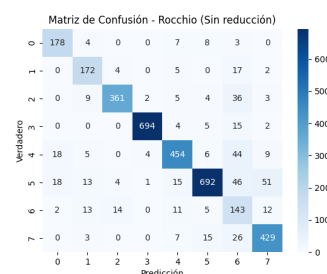
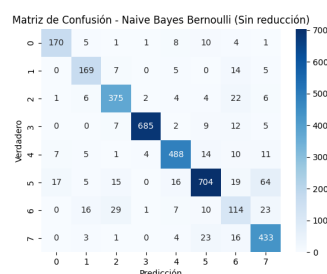
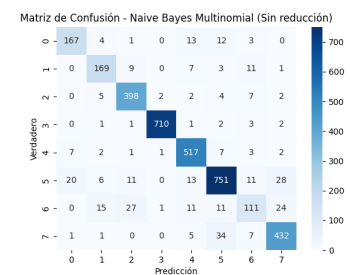
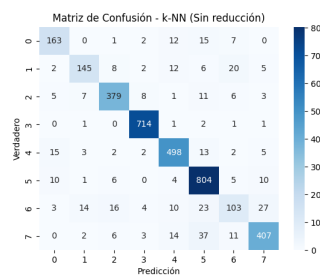
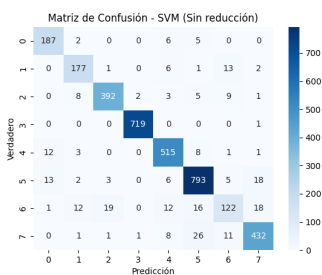
3.1. Comparación de las Matrices de Confusión

La comparación de las matrices de confusión para los distintos modelos y métodos de reducción dimensional permite analizar el rendimiento de cada enfoque en la clasificación de las categorías de documentos.

Sin Reducción Dimensional

En términos generales, SVM sobresale como el modelo más preciso y confiable, logrando clasificar correctamente en la mayoría de las categorías con pocos errores. Los métodos Naive Bayes, tanto Multinomial como Bernoulli, también mostraron un desempeño sólido, especialmente en categorías más frecuentes, aunque con ligeros errores en las menos representadas.

Por otro lado, k-NN y Rocchio, aunque efectivos en categorías grandes, presentaron mayores dificultades para clasificar correctamente documentos de categorías menos frecuentes, reflejando una sensibilidad mayor a las distribuciones desbalanceadas.



Reducción Dimensional - Chi²

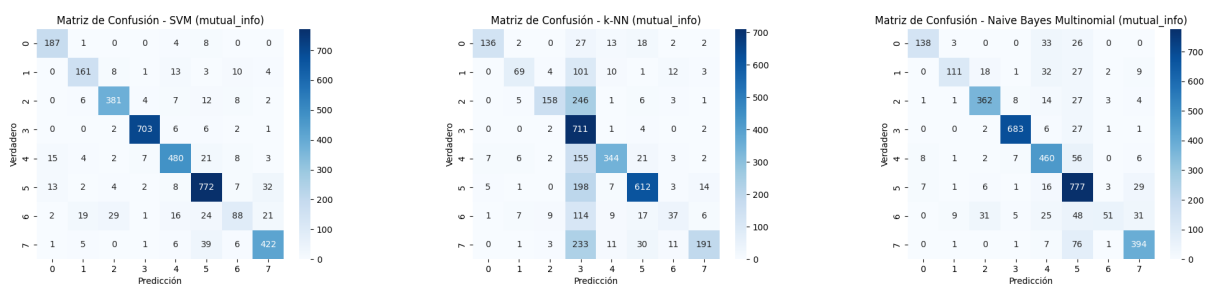
SVM mantiene un desempeño casi idéntico al caso sin reducción, mostrando pocos errores incluso en categorías menos representadas. Los modelos Naive

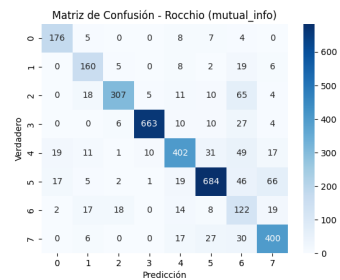
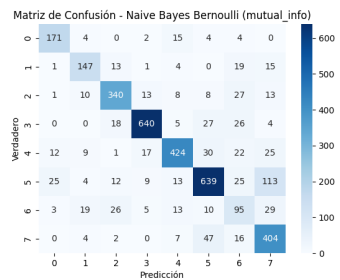
Bayes (Multinomial y Bernoulli) muestran consistencia en las categorías principales, con Bernoulli logrando un mejor balance entre falsos positivos y negativos y ligeras mejoras en categorías como CULTURA (3) y SOCIEDAD (5). Mientras, k-NN experimenta un desempeño más débil en categorías pequeñas, aunque presenta una leve mejora en categorías grandes como "DEPORTES" (1). Finalmente, Rocchio mejora su precisión en categorías principales, pero sigue mostrando dificultades en las categorías menos frecuentes.



Reducción Dimensional - Mutual Info

SVM sigue siendo el modelo más confiable, aunque experimenta un leve incremento de errores en categorías menores como CEE (7). Naive Bayes (Multinomial y Bernoulli) muestra una ligera disminución de precisión en comparación con Chi² y ANOVA, aunque Bernoulli mantiene consistencia en categorías. En contraste, k-NN presenta el peor desempeño entre todas las reducciones, acumulando numerosos errores en categorías menos representadas. Rocchio, por su parte, registra el desempeño más bajo en general, con altos niveles de confusión en las categorías menores, lo que limita su efectividad en este contexto.

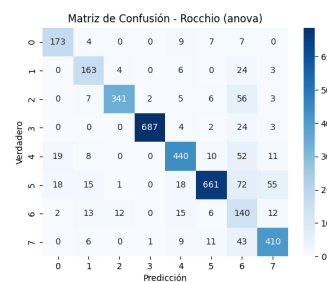
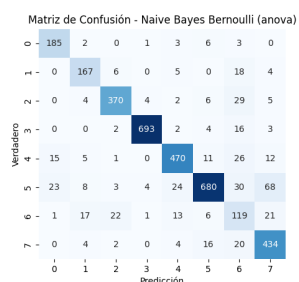
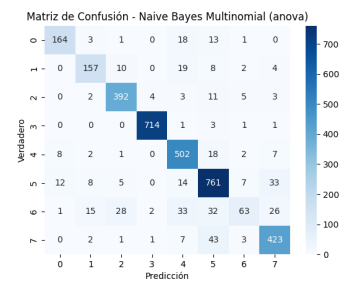
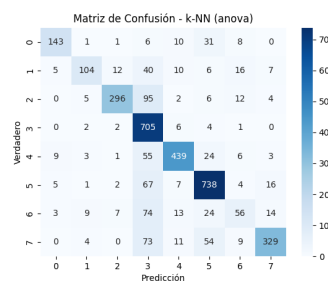
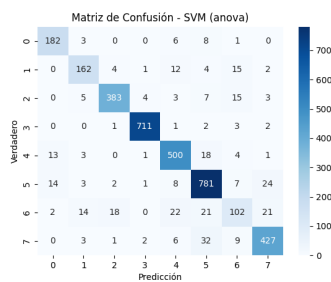




Reducción Dimensional – ANOVA

SVM mantiene un rendimiento robusto y consistente, similar a χ^2 , destacándose en todas las categorías. Naive Bayes Multinomial también muestra un buen desempeño. Bernoulli presenta mejoras destacadas en CULTURA (3), aunque los problemas persisten en categorías pequeñas como CEE (7). k-NN muestra un rendimiento ligeramente mejor que con χ^2 , pero sigue acumulando errores significativos en categorías menos representadas.

Finalmente, Rocchio tiene un rendimiento comparable al de χ^2 , con ligeras mejoras en las categorías más grandes.



3.2. Análisis de la Precisión y el Tiempo

La evaluación de precisión y tiempos de ejecución permite identificar la eficacia y eficiencia de los modelos de clasificación en diferentes escenarios, considerando configuraciones con y sin reducción dimensional.

Sin Reducción Dimensional

SVM destacó como el modelo más preciso (92.69% en prueba), aunque a un alto costo computacional (566.85 segundos). Naive Bayes Multinomial mostró un

equilibrio ideal entre precisión (90.42%) y eficiencia, con un tiempo mínimo de ejecución (0.18 segundos). Su variante Bernoulli, aunque más lenta (0.34 segundos), alcanzó una precisión moderada (87.17%). Por otro lado, k-NN ofreció un desempeño sólido (89.25%) con tiempos razonables (19.34 segundos), mientras que Rocchio, aunque eficiente en tiempo (0.30 segundos), fue el modelo menos preciso (86.75%).

=== Resultados SIN reducción dimensional ===				
Modelo	Método	Accuracy Promedio (Validación)	Accuracy (Prueba)	Tiempo (s)
k-NN	Sin reducción	0.8962	0.8925	19.34
Naive Bayes Multinomial	Sin reducción	0.9051	0.9042	0.18
Naive Bayes Bernoulli	Sin reducción	0.8685	0.8717	0.34
Rocchio	Sin reducción	0.8643	0.8675	0.3
SVM	Sin reducción	0.9383	0.9269	566.85

Con Reducción Dimensional

Con la reducción **Chi²**, SVM mantuvo una alta precisión (90.64%) con una reducción significativa en tiempo (123.2 segundos). Naive Bayes Multinomial continuó mostrando un sólido rendimiento (89.03%) con tiempos ultrarrápidos (0.07 segundos), y Bernoulli fue similar en precisión (87.53%). k-NN perdió algo de precisión (82.67%) pero mantuvo tiempos razonables (12.2 segundos), mientras que Rocchio mostró menor precisión (83.42%) pero excelente eficiencia (0.09 segundos).

=== Resultados CON reducción dimensional: chi2 ===				
Modelo	Método	Accuracy Promedio (Validación)	Accuracy (Prueba)	Tiempo (s)
k-NN	chi2	0.8394	0.8267	12.2
Naive Bayes Multinomial	chi2	0.889	0.8903	0.07
Naive Bayes Bernoulli	chi2	0.8735	0.8753	0.07
Rocchio	chi2	0.835	0.8342	0.09
SVM	chi2	0.9062	0.9064	123.2

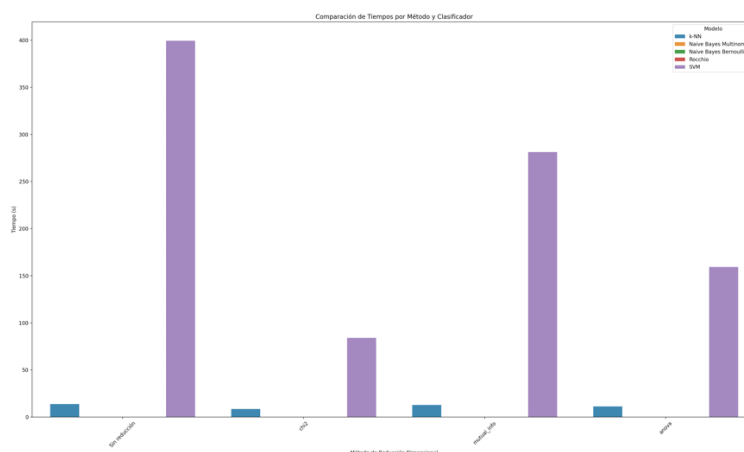
Con la reducción **Mutual Information**, aunque SVM permaneció confiable (88.72%), su tiempo de ejecución aumentó considerablemente (404.33 segundos). Naive Bayes Multinomial y Bernoulli experimentaron caídas en precisión (82.67% y 79.44%, respectivamente), aunque se mantuvieron rápidos (0.11 y 0.17 segundos). k-NN fue el menos efectivo (62.72%), y Rocchio mostró una caída significativa en precisión (80.94%) aunque mantuvo tiempos eficientes (0.18 segundos).

=== Resultados CON reducción dimensional: mutual_info ===					
Modelo	Método	Accuracy Promedio (Validación)	Accuracy (Prueba)	Tiempo (s)	
k-NN	mutual_info	0.639	0.6272	18.49	
Naive Bayes Multinomial	mutual_info	0.8284	0.8267	0.11	
Naive Bayes Bernoulli	mutual_info	0.7931	0.7944	0.17	
Rocchio	mutual_info	0.8177	0.8094	0.18	
SVM	mutual_info	0.8872	0.8872	404.33	

Con la reducción **ANOVA**, SVM y Naive Bayes Multinomial destacaron nuevamente con precisiones altas (90.22% y 88.22%, respectivamente) y tiempos moderados (213.26 y 0.09 segundos). Bernoulli se mantuvo consistente (86.61%) con tiempos rápidos (0.10 segundos). k-NN mejoró respecto a Mutual Information (78.06%) con tiempos razonables (15.87 segundos), mientras que Rocchio tuvo un desempeño moderado (83.75%) con alta eficiencia (0.13 segundos).

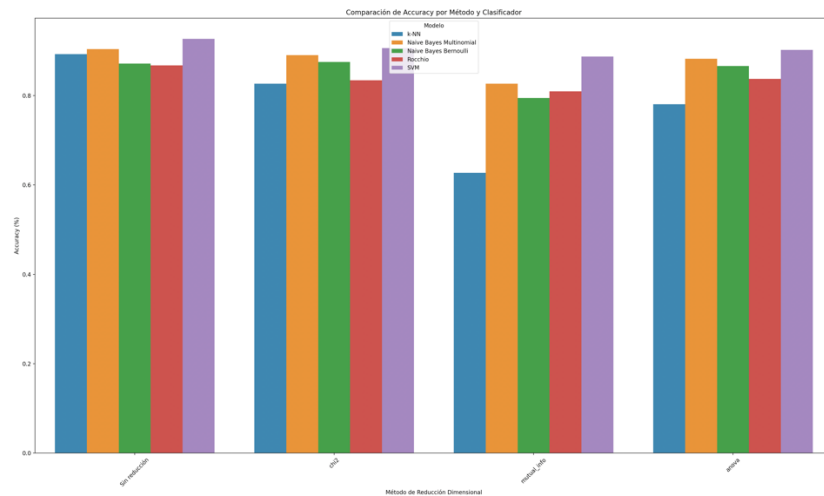
=== Resultados CON reducción dimensional: anova ===					
Modelo	Método	Accuracy Promedio (Validación)	Accuracy (Prueba)	Tiempo (s)	
k-NN	anova	0.7991	0.7806	15.87	
Naive Bayes Multinomial	anova	0.886	0.8822	0.09	
Naive Bayes Bernoulli	anova	0.8609	0.8661	0.1	
Rocchio	anova	0.8337	0.8375	0.13	
SVM	anova	0.9042	0.9022	213.26	

Los resultados indican que SVM es el modelo más preciso en todas las configuraciones, especialmente sin reducción dimensional, aunque con un alto costo en tiempo. La reducción Chi² resultó ser la más eficiente, reduciendo significativamente los tiempos sin comprometer la precisión, particularmente en SVM y Naive Bayes Multinomial. ANOVA también ofreció un buen balance, mientras que Mutual Information demostró ser la técnica menos eficiente en términos de tiempo, impactando negativamente a modelos como k-NN y SVM.



Naive Bayes Multinomial se posiciona como el clasificador más equilibrado, con altos niveles de precisión y tiempos extremadamente bajos en todas las configuraciones. Rocchio, aunque menos preciso, se mantuvo consistentemente

eficiente en términos computacionales. Por otro lado, k-NN mostró una sensibilidad mayor a la reducción dimensional, afectando tanto su precisión como sus tiempos de ejecución, especialmente con Mutual Information.



3.3. Resultados Generados y Mejor Valor de k

Además de los análisis previos, se generó un archivo CSV con los resultados finales de todas las combinaciones de modelos y métodos de reducción dimensional. Este archivo incluye las siguientes columnas:

El archivo completo se encuentra disponible como resultados_finales.csv.

Modelo	Método	Accuracy Promedio (Validación)	Tiempo Total (s)
k-NN	Sin reducción	0.8925	19.33508801460266
Naive Bayes Multinomial	Sin reducción	0.9041666666666667	0.1779019832611084
Naive Bayes Bernoulli	Sin reducción	0.8716666666666667	0.34264397621154785
Rocchio	Sin reducción	0.8675	0.30002689361572266
SVM	Sin reducción	0.9269444444444445	566.8481328487396
k-NN	chi2	0.8266666666666667	12.200433015823364
Naive Bayes Multinomial	chi2	0.8902777777777778	0.0669708251953125
Naive Bayes Bernoulli	chi2	0.8752777777777778	0.07192587652478627
Rocchio	chi2	0.8341666666666667	0.09155797958374023
SVM	chi2	0.9043888888888889	123.20099996162415
k-NN	mutual_info	0.8272222222222222	18.4934084154508
Naive Bayes Multinomial	mutual_info	0.8266666666666667	0.1112029524597168
Naive Bayes Bernoulli	mutual_info	0.7944444444444444	0.16566205024719238
Rocchio	mutual_info	0.8094444444444444	0.18173527717590332
SVM	mutual_info	0.8872222222222222	404.3256461620331
k-NN	anova	0.7805555555555556	15.873458862304688
Naive Bayes Multinomial	anova	0.8822222222222222	0.091648569790039
Naive Bayes Bernoulli	anova	0.8661111111111112	0.0980370044708252
Rocchio	anova	0.8375	0.13088393211364746
SVM	anova	0.9022222222222223	213.2575040403296

Durante el proceso de validación cruzada para el clasificador k-NN, se determinó que el mejor valor de k es k=1, logrando maximizar la precisión en las configuraciones evaluadas.

```

=== Buscando el mejor k global para k-NN ===
k=1, mean_accuracy=0.8975
k=2, mean_accuracy=0.8722
k=3, mean_accuracy=0.8854
k=4, mean_accuracy=0.8826
k=5, mean_accuracy=0.8834
k=6, mean_accuracy=0.8799
k=7, mean_accuracy=0.8804
k=8, mean_accuracy=0.8797
k=9, mean_accuracy=0.8781
k=10, mean_accuracy=0.8771
k=11, mean_accuracy=0.8765
k=12, mean_accuracy=0.8742
k=13, mean_accuracy=0.8753
k=14, mean_accuracy=0.8746
k=15, mean_accuracy=0.8751
k=16, mean_accuracy=0.8753
k=17, mean_accuracy=0.8753
k=18, mean_accuracy=0.8753
k=19, mean_accuracy=0.8753
k=20, mean_accuracy=0.8753
Mejor k encontrado: 1

```

4. CONCLUSIONES

En este informe se evaluaron técnicas de clasificación automática aplicadas a documentos, analizando el impacto de la reducción dimensional en la precisión y el tiempo de ejecución. Los hallazgos clave son:

1. Rendimiento de los Modelos:

SVM se consolidó como el modelo más preciso, destacando especialmente sin reducción (92.69%) y con χ^2 y ANOVA. Naive Bayes Multinomial mostró un excelente balance entre precisión y eficiencia computacional, mientras que Bernoulli, aunque ligeramente menos preciso, mantuvo tiempos igualmente rápidos. Por el contrario, k-NN y Rocchio mostraron mayor sensibilidad a la reducción dimensional, perdiendo precisión en categorías pequeñas.

2. Impacto de la Reducción Dimensional:

χ^2 fue el método más eficiente, reduciendo significativamente los tiempos computacionales sin comprometer la precisión, especialmente para SVM y Naive Bayes Multinomial. ANOVA también ofreció buenos resultados, mientras que Mutual Information fue el menos efectivo, aumentando los tiempos de ejecución sin mejoras significativas en precisión.

3. Eficiencia Computacional:

Naive Bayes Multinomial y Bernoulli fueron los más rápidos, con tiempos consistentemente inferiores a 0.2 segundos. SVM, aunque preciso, presentó tiempos altos sin reducción (566.85 s) y con Mutual Information (404.33 s), mejorando significativamente con χ^2 (123.2 s) y ANOVA (213.26 s). Rocchio fue consistente en eficiencia, pero menos preciso, mientras que k-NN tuvo un desempeño razonable en tiempo, excepto con Mutual Information.

4. Mejor Valor de k:

Para k-NN, el valor óptimo fue **k=1**, lo que maximizó la precisión en todas las configuraciones evaluadas.

La combinación de SVM o Naive Bayes Multinomial con χ^2 demostró ser la más efectiva, ofreciendo el mejor balance entre precisión y tiempo computacional. Por el contrario, Mutual Information no mostró ventajas competitivas, y Rocchio fue el modelo menos efectivo en términos de precisión. Estos resultados subrayan la importancia de elegir cuidadosamente los clasificadores y métodos de reducción dimensional según las necesidades específicas del problema.

ANEXO: Código

```
from tqdm import tqdm
import nltk
from nltk.corpus import stopwords
from nltk.stem.snowball import SpanishStemmer
import ssl
import os
import pandas as pd
import xml.etree.ElementTree as ET
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import NearestCentroid
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_classif, f_classif
import time
import matplotlib.pyplot as plt
import warnings
from sklearn.preprocessing import LabelEncoder

# Suprimir todos los warnings
warnings.filterwarnings("ignore")

#evitar errores al descargar recursos
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

# Descargar stopwords de NLTK
nltk.download('stopwords')
ruta_xml = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'categorias18000.xml')

# Parsear el archivo XML
tree = ET.parse(ruta_xml)
root = tree.getroot()

# Extraer los documentos
documentos = []
for doc in root.findall('DOC'):
    titulo = doc.find('TITLE').text if doc.find('TITLE') is not None else ""
    texto = doc.find('TEXT').text if doc.find('TEXT') is not None else ""
    categoria = doc.find('CATEGORY').text if doc.find('CATEGORY') is not None else ""
    if categoria.strip(): # Evitar categorías vacías
        documentos.append({'texto': titulo + ' ' + texto, 'categoria': categoria.strip().upper()})
    else:
        print("Documento sin categoría encontrado")
```

```

df = pd.DataFrame(documentos)
print(f"Total de documentos extraídos: {len(documentos)}")

# Mostrar la distribución inicial de categorías
print("Distribución inicial de categorías:")
print(df['categoria'].value_counts())

# Configurar stopwords y lematizador
stop_words = set(stopwords.words('spanish'))
stemmer = SpanishStemmer()

# Función de preprocesamiento del texto
def preprocesar_texto(texto):
    texto = texto.lower()
    texto = re.sub(r'\W+', ' ', texto)
    palabras = texto.split()
    #stemming y filtrar palabras vacías
    palabras = [stemmer.stem(palabra) for palabra in palabras if palabra not in stop_words]
    return ' '.join(palabras)

df['texto_procesado'] = df['texto'].apply(preprocesar_texto)
print("Datos preprocesados:")
print(df.head())

# Vectorización con TF-IDF
vectorizer = TfidfVectorizer(max_features=5000) # Limitar a las 1000 características más relevantes
X = vectorizer.fit_transform(df['texto_procesado'])
y = df['categoria']

# Verificar tamaño de la matriz TF-IDF
print(f"Tamaño de la matriz TF-IDF: {X.shape}")

# Convertir las categorías a valores discretos (numericos)
le = LabelEncoder()
y = le.fit_transform(df['categoria'])

# División de datos en entrenamiento y test (80-20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
print(f"Tamaño del conjunto de entrenamiento: {X_train.shape}")
print(f"Tamaño del conjunto de prueba: {X_test.shape}")

from sklearn.model_selection import cross_val_score

def obtener_mejor_k_global(X_train, y_train):
    best_k = None
    best_score = 0
    for k in range(1, 21):
        knn_model = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(knn_model, X_train, y_train, cv=5, scoring='accuracy')
        mean_score = scores.mean()
        if mean_score > best_score:
            best_k = k
            best_score = mean_score
        print(f"k={k}, mean_accuracy={mean_score:.4f}")
    return best_k

```



```

def aplicar_reduccion_dimensional(X_train, X_test, y_train, metodo, k=1000):
    if metodo == 'chi2':
        selector = SelectKBest(chi2, k=k)
    elif metodo == 'mutual_info':
        selector = SelectKBest(mutual_info_classif, k=k)
    elif metodo == 'anova':
        selector = SelectKBest(f_classif, k=k)
    else:
        raise ValueError("Método no soportado")

    X_train_reducido = selector.fit_transform(X_train, y_train)
    X_test_reducido = selector.transform(X_test)
    return X_train_reducido, X_test_reducido

resultados_tiempos = []
resultados = []

metodos_reduccion = ['chi2', 'mutual_info', 'anova']
# Encontrar el mejor k global
print("\n=== Buscando el mejor k global para k-NN ===")
mejor_k_global = obtener_mejor_k_global(X_train, y_train)
print(f"Mejor k encontrado: {mejor_k_global}")

modelos = {
    "k-NN": KNeighborsClassifier(n_neighbors=mejor_k_global),
    "Naive Bayes Multinomial": MultinomialNB(alpha=0.01),
    "Naive Bayes Bernoulli": BernoulliNB(alpha=0.01),
    "Rocchio": NearestCentroid(),
    "SVM": SVC(kernel='linear', random_state=42)
}

# k-Fold Cross Validation (usando 5 folds)
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

from tabulate import tabulate # Para generar tablas en consola

# Listas para almacenar resultados separados por método
resultados_sin_reduccion = []
resultados_con_reduccion = []

def evaluar_modelos(X_train, X_test, y_train, y_test, metodo, lista_resultados):
    """
    Evalúa los modelos utilizando validación cruzada (StratifiedKFold) y calcula las métricas.
    """
    start_global = time.time()
    for nombre, modelo in modelos.items():
        start_modelo = time.time()
        accuracies = []

        # Validación cruzada con 5 pliegues
        for train_index, val_index in kf.split(X_train, y_train):
            X_train_fold, X_val_fold = X_train[train_index], X_train[val_index]
            y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]

            # Entrenamiento y predicción en cada pliegue
            modelo.fit(X_train_fold, y_train_fold)
            y_val_pred = modelo.predict(X_val_fold)

```

```

    accuracies.append(accuracy_score(y_val_fold, y_val_pred))

# Cálculo de métricas finales
mean_acc = sum(accuracies) / len(accuracies)
y_pred = modelo.predict(X_test) # Asegurarse de usar el conjunto reducido si aplica
test_acc = accuracy_score(y_test, y_pred)
end_modelo = time.time()

# Matriz de confusión para el conjunto de prueba
cm_test = confusion_matrix(y_test, y_pred)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues')
plt.title(f"Matriz de Confusión - {nombre} ({metodo})")
plt.xlabel("Predicción")
plt.ylabel("Verdadero")
plt.savefig(f"matriz_confusion_{nombre}_{metodo}.png")
plt.close()

# Agregar resultados a la lista de resultados
lista_resultados.append([nombre, metodo, f"{mean_acc:.4f}", f"{test_acc:.4f}",
f"{end_modelo - start_modelo:.2f}"])
# Agregar a resultados_tiempos para gráficos
resultados_tiempos.append({
    "Modelo": nombre,
    "Método": metodo,
    "Accuracy": test_acc,
    "Tiempo (s)": end_modelo - start_modelo
})
end_global = time.time()
print(f"\nTiempo total para {metodo}: {end_global - start_global:.2f} segundos")

# Generar tablas al final
headers = ["Modelo", "Método", "Accuracy Promedio (Validación)", "Accuracy (Prueba)", "Tiempo (s)"]

# Evaluación sin reducción dimensional
print(f"\n=== Evaluación SIN reducción dimensional ===")
evaluar_modelos(X_train, X_test, y_train, y_test, "Sin reducción", resultados_sin_reduccion)
# Tabla para "Sin reducción"
print("\n=== Resultados SIN reducción dimensional ===")
print(tabulate(resultados_sin_reduccion, headers=headers, tablefmt="grid"))

# Evaluación con reducción dimensional
for metodo in metodos_reduccion:
    print(f"\n=== Evaluación con reducción dimensional: {metodo} ===")
    print(f"Dimensiones originales: {X_train.shape}, {X_test.shape}")

    # Aplicar la reducción dimensional
    X_train_reducido, X_test_reducido = aplicar_reduccion_dimensional(X_train, X_test, y_train,
    metodo)
    print(f"Dimensiones reducidas ({metodo}): {X_train_reducido.shape},
    {X_test_reducido.shape}")

    # Evaluar los modelos con los datos reducidos
    evaluar_modelos(X_train_reducido, X_test_reducido, y_train, y_test, metodo,
    resultados_con_reduccion)
    # Filtrar los resultados para este método
    resultados_metodo = [fila for fila in resultados_con_reduccion if fila[1] == metodo]

```

```

# Generar la tabla para este método
print(f"\n=== Resultados CON reducción dimensional: {metodo} ===")
print(tabulate(resultados_metodo, headers=headers, tablefmt="grid"))

# Convertir resultados a DataFrame
df_tiempos = pd.DataFrame(resultados_tiempos)
# Reiniciar índice
df_tiempos.reset_index(drop=True, inplace=True)

df_resultados = pd.DataFrame(resultados)

# Crear un DataFrame con los resultados incluyendo el modelo, método y tiempo total
df_resultados = pd.DataFrame([
    {
        "Modelo": r["Modelo"],
        "Método": r["Método"],
        "Accuracy Promedio (Validación)": r["Accuracy"],
        "Tiempo Total (s)": r["Tiempo (s)"]
    }
    for r in resultados_tiempos if 'Modelo' in r
])

# Guardar los resultados en un archivo CSV
df_resultados.to_csv("resultados_finales.csv", index=False)

print("Resultados exportados a 'resultados_finales.csv'")

# Comparar resultados
print("\nResultados comparativos:")
print(df_tiempos)

# Gráfico de comparación de tiempos
sns.barplot(data=df_tiempos, x="Método", y="Tiempo (s)", hue="Modelo")
plt.title("Comparación de Tiempos por Método y Clasificador")
plt.ylabel("Tiempo (s)")
plt.xlabel("Método de Reducción Dimensional")
plt.xticks(rotation=45)
plt.savefig("tiempos_comparacion.png")
plt.close()

# Gráfico de comparación de precisión
sns.barplot(data=df_tiempos, x="Método", y="Accuracy", hue="Modelo")
plt.title("Comparación de Accuracy por Método y Clasificador")
plt.ylabel("Accuracy (%)")
plt.xlabel("Método de Reducción Dimensional")
plt.xticks(rotation=45)
plt.savefig("accuracy_comparacion.png")
plt.close()

```