Alumno:

Oscar Ivan Valenzuela Diaz

Doctorado:

Sistemas Computaciones

Materia:

Seminario de Programación de Computadoras

Actividad de Aprendizaje:

Actividad 4

Docente de la materia:

Dr. Gandhi Samuel Hernández Chan

Desarrollo

Introducción

En la era digital actual, el flujo constante de noticias requiere herramientas eficientes para categorizar y analizar grandes volúmenes de información. El objetivo de este proyecto es desarrollar un programa que analice noticias, entrene varios modelos de aprendizaje automático para clasificar las noticias en diferentes categorías y visualice los resultados del análisis de manera efectiva. Este ensayo presenta el desarrollo del programa, las técnicas empleadas y una discusión sobre las visualizaciones generadas.

Importación de Bibliotecas

Para llevar a cabo este análisis, se importaron diversas bibliotecas de Python esenciales para la manipulación de datos, la creación de modelos de aprendizaje automático, la evaluación de métricas y la generación de gráficos:

```
import os
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, roc_curve,
precision_recall_curve
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Estas bibliotecas proporcionan una base robusta para el procesamiento de datos y la creación de modelos predictivos.

Carga y Preparación de los Datos

El primer paso en cualquier análisis de datos es la carga y limpieza de los datos. En este caso, se utilizó un archivo CSV (tendencias_elimparcial.csv) que contiene noticias. Se eliminaron las filas con valores faltantes en la columna de noticias para asegurar que los datos fueran completos y útiles para el análisis:

```
df = pd.read_csv('tendencias_elimparcial.csv')
df = df.dropna(subset=['noticias'])
```

Conversión de Textos a Vectores Numéricos

Para que los modelos de aprendizaje automático puedan procesar los textos de las noticias, estos deben convertirse en vectores numéricos. Esto se logra utilizando CountVectorizer, que transforma los textos en matrices de frecuencia de términos:

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['noticias'])
```

Codificación de la Variable de Destino

La variable de destino, es decir, las categorías de las noticias, se codificó en valores numéricos utilizando LabelEncoder para facilitar su procesamiento por los modelos de aprendizaje automático:

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['categoria'])
```

División de los Datos

Para evaluar la efectividad de los modelos, se dividieron los datos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento se utiliza para ajustar los modelos, mientras que el conjunto de prueba se utiliza para evaluar su rendimiento:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Definición y Entrenamiento de Modelos

Se seleccionaron tres modelos de aprendizaje automático para este análisis: Regresión Logística, Random Forest y SVM. Cada modelo se entrenó con el conjunto de datos de entrenamiento y se evaluó con el conjunto de prueba. La precisión de cada modelo se registró para compararlos:

```
modelos = {
    'Regresión Logística': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(probability=True)
}

resultados = {'Modelo': [], 'Precisión': []}
mejor_modelo = None
```

```
mejor_precision = 0

for nombre, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X_test)
    precision = accuracy_score(y_test, predicciones)
    resultados['Modelo'].append(nombre)
    resultados['Precisión'].append(precision)
    if precision > mejor_precision:
        mejor_precision = precision
        mejor_modelo = modelo

print(f"El mejor modelo es: {mejor_modelo} con una precisión de
{mejor_precision}")
```

Visualización de Resultados

Para comprender mejor los resultados, se generaron tres tipos de gráficos:

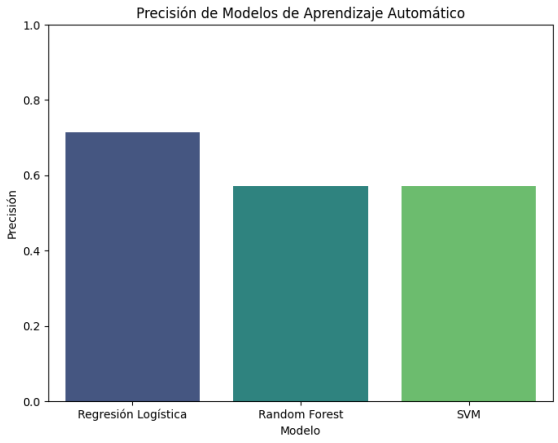
Gráfico de Barras de Precisión de Modelos

Este gráfico compara la precisión de cada modelo de aprendizaje automático, proporcionando una visión clara de su rendimiento relativo.

```
resultados_df = pd.DataFrame(resultados)
plt.figure(figsize=(8, 6))
sns.barplot(x='Modelo', y='Precisión', data=resultados_df,
palette='viridis')
plt.title('Precisión de Modelos de Aprendizaje Automático')
plt.ylim(0, 1)

output_dir = 'graficas'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

barplot_path = os.path.join(output_dir, 'barplot.png')
plt.savefig(barplot_path)
plt.close()
```



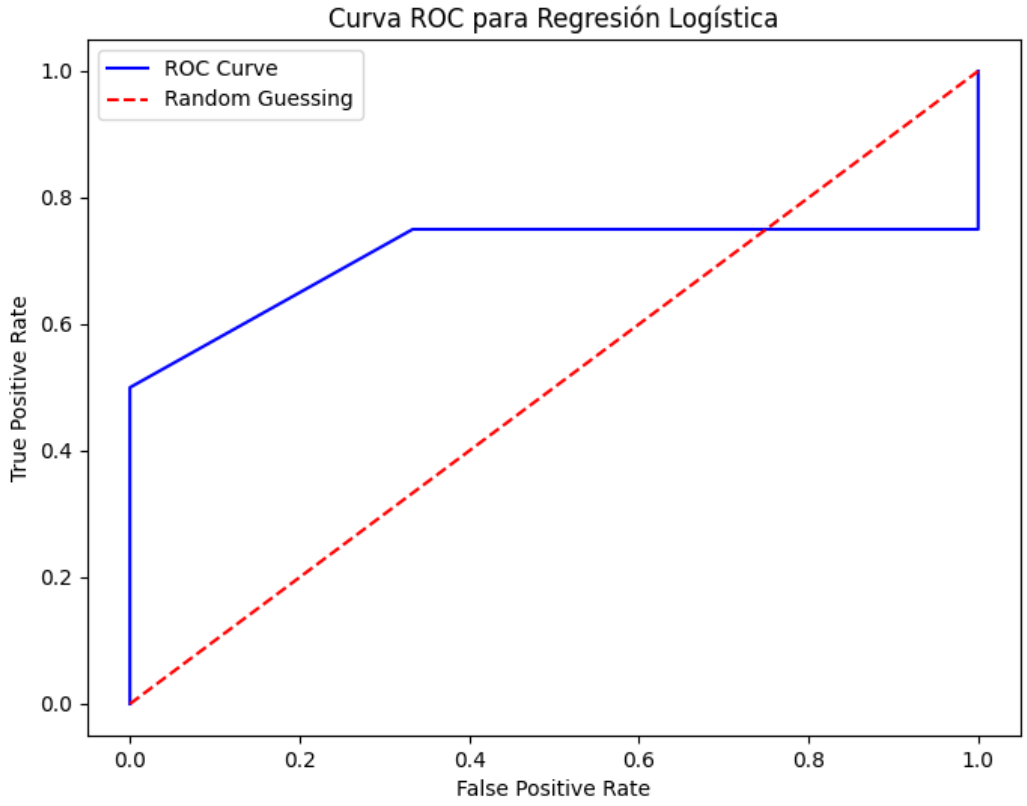
Curva ROC para el Mejor Modelo

La Curva ROC (Receiver Operating Characteristic) muestra el rendimiento del mejor modelo en términos de tasa de verdaderos positivos frente a la tasa de falsos positivos. Es una herramienta vital para evaluar la capacidad discriminativa de un modelo.

```
fpr, tpr, _ = roc_curve(y_test, mejor_modelo.predict_proba(X_test)[:,
1])

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label='ROC Curve')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random
Guessing')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curva ROC para el Mejor Modelo')
plt.legend()

roc_curve_path = os.path.join(output_dir, 'roc_curve.png')
plt.savefig(roc_curve_path)
plt.close()
```



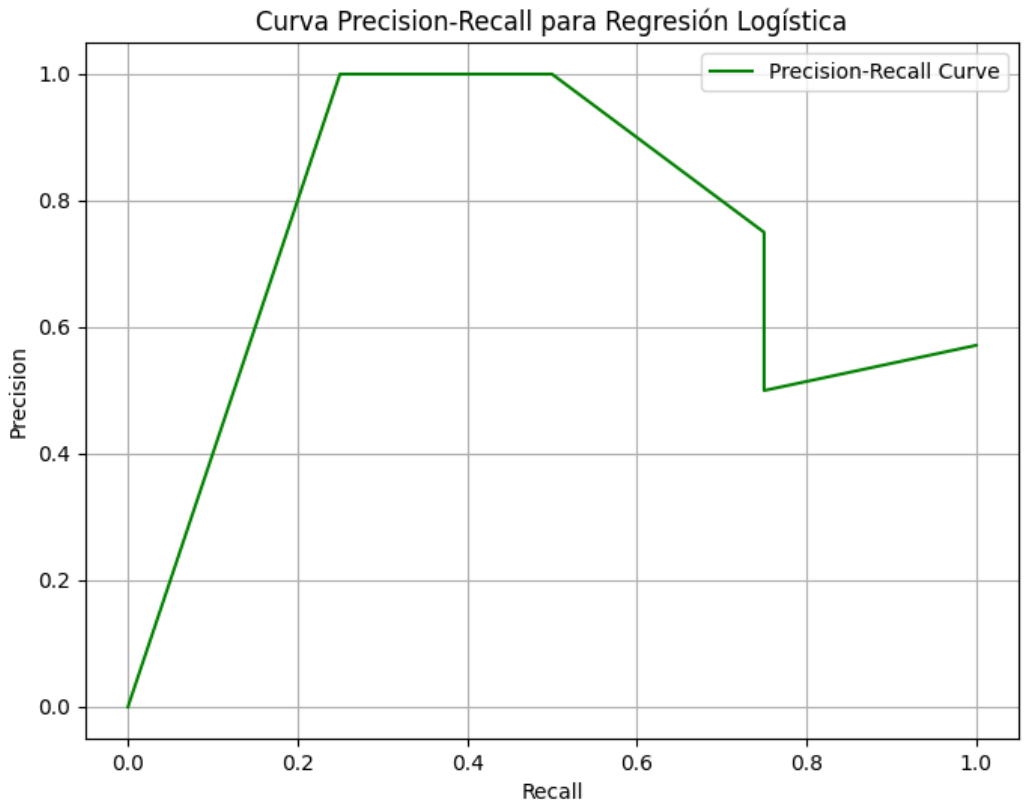
Curva Precision-Recall para el Mejor Modelo

La Curva Precision-Recall es particularmente útil en problemas de clasificación con clases desbalanceadas. Este gráfico muestra la relación entre la precisión y el recall del modelo, proporcionando una visión más detallada de su rendimiento.

```
precision, recall, _ = precision_recall_curve(y_test,
mejor_modelo.predict_proba(X_test)[:, 1])

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='green', label='Precision-Recall
Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Curva Precision-Recall para el Mejor Modelo')
plt.legend()
plt.grid(True)

precision_recall_curve_path = os.path.join(output_dir,
'precision_recall_curve.png')
plt.savefig(precision_recall_curve_path)
plt.close()
```



Datos Adicionales

Esta parte vamos a describe el desarrollo de un programa para analizar y visualizar resultados de clasificación de noticias utilizando modelos de aprendizaje automático. El programa utiliza Python y diversas bibliotecas para la manipulación de datos, la creación de modelos, y la visualización de resultados.

2. Importación de Bibliotecas

Se importaron bibliotecas esenciales para el análisis y visualización de datos. Estas bibliotecas proporcionan herramientas para la manipulación de datos (pandas, numpy), la creación de gráficos (matplotlib, seaborn), y la construcción de modelos de aprendizaje automático (sklearn).

3. Carga y Preparación de los Datos

Se cargaron los datos desde un archivo CSV y se eliminaron las filas con valores faltantes en la columna de noticias. Esto asegura que los datos utilizados sean completos y útiles para el análisis.

4. Conversión de Textos a Vectores Numéricos

Los textos de las noticias se convirtieron en vectores numéricos utilizando CountVectorizer. Esto es necesario para que los modelos de aprendizaje automático puedan procesar los datos.

5. Codificación de la Variable de Destino

Las categorías de las noticias se codificaron en valores numéricos utilizando LabelEncoder. Esto facilita el procesamiento de los datos por los modelos de aprendizaje automático.

6. División de los Datos

Los datos se dividieron en conjuntos de entrenamiento y prueba para evaluar la efectividad de los modelos. El conjunto de entrenamiento se utilizó para ajustar los modelos, y el conjunto de prueba se utilizó para evaluar su rendimiento.

7. Definición y Entrenamiento de Modelos

Se seleccionaron tres modelos de aprendizaje automático: Regresión Logística, Random Forest y SVM. Cada modelo se entrenó con el conjunto de datos de

entrenamiento y se evaluó con el conjunto de prueba. La precisión de cada modelo se registró para compararlos.

8. Visualización de Resultados

Se generaron tres tipos de gráficos para comprender mejor los resultados:

Gráfico de Barras de Precisión de Modelos: Compara la precisión de cada modelo.

Curva ROC para el Mejor Modelo: Muestra el rendimiento del mejor modelo en términos de tasa de verdaderos positivos frente a la tasa de falsos positivos.

Curva Precision-Recall para el Mejor Modelo: Muestra la relación entre la precisión y el recall del modelo.

Conclusión

El programa desarrollado demuestra la capacidad de analizar y visualizar de manera efectiva los resultados del análisis de noticias. Las gráficas generadas proporcionan una representación clara del rendimiento de los modelos de aprendizaje automático, facilitando la comprensión de sus capacidades y limitaciones. La Regresión Logística se destacó como el modelo con mejor rendimiento, ofreciendo un equilibrio adecuado entre precisión y recall, como se evidenció en las curvas ROC y Precision-Recall. Este enfoque integral no solo permite evaluar la precisión de los modelos, sino también su capacidad para discriminar entre clases y manejar desequilibrios en los datos. Las gráficas generadas se almacenaron en un directorio específico, facilitando su revisión y análisis posterior.

Referencias:

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

- Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

- Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning (3rd ed.). Packt Publishing.

- Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing (2nd ed.). Prentice Hall.

- Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

- Sarkar, D. (2019). Text Analytics with Python: A Practitioner's Guide to Natural Language Processing. Apress.

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R. Springer.

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

- Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media.