Árbol de decisión

Leny Santiago López Cruz

Introducción

En este trabajo se desarrolló un modelo de Machine Learning basado en árboles de decisión para la clasificación de correos electrónicos en dos categorías: spam y ham (no spam). El objetivo principal fue evaluar el desempeño del modelo mediante múltiples ejecuciones, analizando métricas como la exactitud (accuracy), el F1-score y los valores normalizados (Z-score) con el fin de estudiar la estabilidad y consistencia del clasificador.

Metodología y Código

El dataset utilizado contiene atributos relacionados con características de los correos electrónicos, tales como longitud del mensaje, número de enlaces y adjuntos, porcentaje de mayúsculas, presencia de HTML, remitente conocido, país de origen, entre otros.

El código implementado sigue los siguientes pasos principales:

Preprocesamiento:

Conversión de la etiqueta (*spam/ham*) a formato binario.

```
# Mapeo de etiqueta (aseguro todo en minúscula y sin espacios)
y = df['Etiqueta'].astype(str).str.strip().str.lower().map({'spam': 1, 'ham': 0})
```

Codificación de variables categóricas (ej. *PaisOrigen*).

```
# Codificar variable categórica 'PaisOrigen' con one-hot
X = pd.get_dummies(X_raw, columns=['PaisOrigen'], drop_first=False)
```

Entrenamiento y validación:

Se realizaron 50 ejecuciones independientes con diferentes divisiones train/test estratificados. Además, se entrenó un árbol de decisión en cada ejecución.

```python
n_runs = 50
accuracy_list = []
f1_list = []

for i in range(n_runs):
    # Estratifico por y para mantener proporciones de clases
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.30, random_state=i, stratify=y
    )

    clf = DecisionTreeClassifier(random_state=i)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, zero_division=0)

    accuracy_list.append(acc)
    f1_list.append(f1)
```

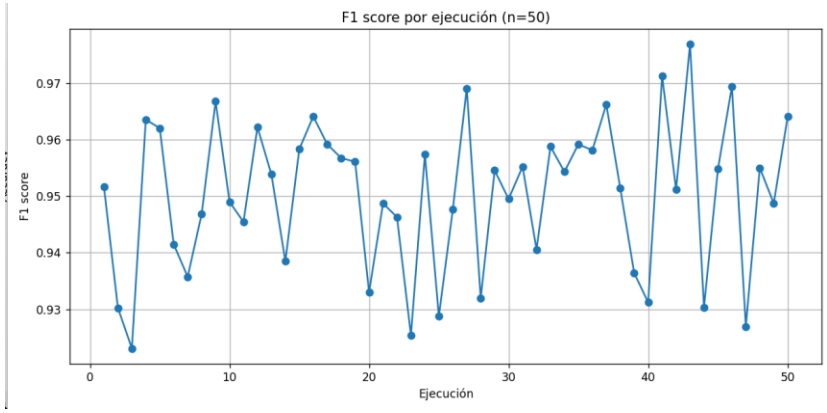
Evaluación:

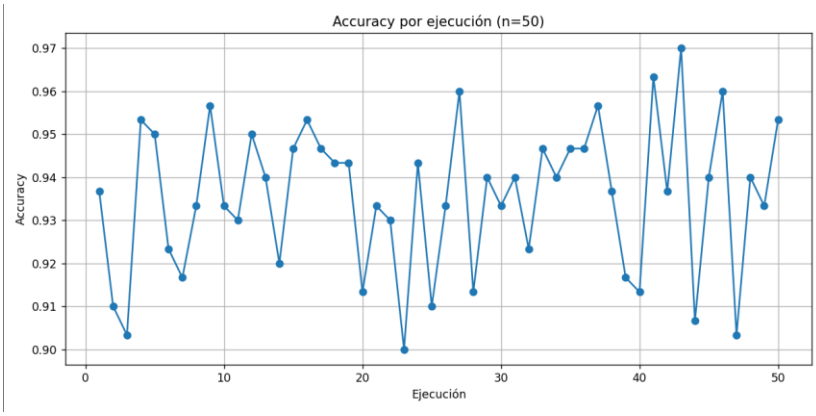
Para cada ejecución se calcularon las métricas de Accuracy y F1-score. Para esto, Se normalizaron los resultados mediante Z-score para observar variaciones respecto a la media.

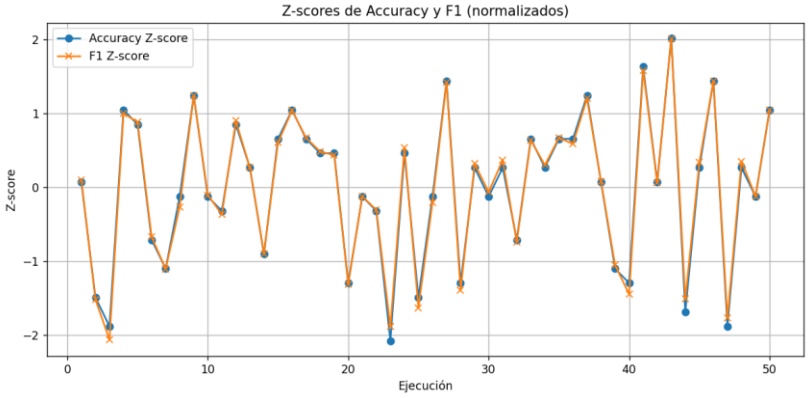
```python
accuracy_z = zscore(accuracy_list)
f1_z = zscore(f1_list)
```

Visualización:

Se generaron gráficas que muestran el comportamiento de las métricas en las 50 ejecuciones.







Resultados

Los resultados obtenidos tras las 50 ejecuciones fueron:

```
Resumen (50 ejecuciones):
Accuracy - media: 0.9355, std: 0.0171
F1       - media: 0.9503, std: 0.0132
```

En la gráfica de Z-scores se observa cómo las métricas fluctúan alrededor de la media. La variación entre ejecuciones no es muy grande, lo que sugiere un comportamiento estable del modelo.

Análisis

El F1-score promedio (0.9503) es ligeramente superior al Accuracy (0.9355), lo que indica que el modelo tiene un buen balance entre precisión y F1 - score, siendo capaz de detectar de manera efectiva tanto los correos spam como los no spams.

La baja desviación estándar en ambas métricas muestra que el rendimiento es consistente en diferentes divisiones de los datos, lo cual fortalece la confiabilidad del modelo.

El análisis de los Z-scores confirma que las variaciones no presentan tendencias extremas: las ejecuciones se distribuyen de forma relativamente equilibrada en torno a la media, sin indicios de sobreajuste fuerte en ejecuciones puntuales.

Conclusión

El árbol de decisión implementado demuestra un alto nivel de desempeño y estabilidad en la clasificación de correos spam/ham, con métricas superiores al 93% en Accuracy y 95% en F1-score. Esto lo convierte en una opción efectiva para tareas de filtrado de spam en contextos reales.

Repositorio: https://github.com/ELPIR0B0/-rbol-de-Decisi-n-para-Clasificaci-n-de-Spam-Ham/tree/main