Nombres: Andres Julian Leon Montoya, Samuel Leonardo Tarazona Arciniegas

Materia: Machine Learning - 802

Actividad: Clasificación correos SPAM/HAM con árbol de decisión

Enlace Github:

Informe: Clasificación de Correos Spam y Ham con Árbol de Decisión

Descripción del Procedimiento

1. Carga del Dataset

Se utilizó el dataset **Spambase (ID=44 en OpenML)**, el cual contiene características extraídas de correos electrónicos con el fin de clasificarlos como spam (no deseados) o ham (legítimos).

Las características incluyen la frecuencia de ciertas palabras y símbolos, así como estadísticas relacionadas con el uso de letras en mayúscula.

La variable objetivo es binaria: 0 para correos ham y 1 para correos spam.

```
# ============================
# 1. Cargar dataset Spambase
# ============================
data = fetch_openml(data_id=44, as_frame=True)
X, y = data.data, data.target

# Convertir etiquetas a enteros (0 = ham, 1 = spam)
y = y.astype(int)

scores = []      # Accuracy
f1_scores = []   # F1-score
```

2. Preparación de los datos

- Se separaron las variables predictoras (X) de la variable objetivo (y).

- Las etiquetas fueron convertidas a enteros para que el algoritmo pudiera procesarlas.

3. Entrenamiento del Modelo

- Se empleó un **Árbol de Decisión** como modelo de clasificación.
- Para garantizar consistencia, se fijó un valor en el parámetro random_state.
- El entrenamiento no se realizó una sola vez, sino en **50 repeticiones**, con el fin de observar la estabilidad del modelo bajo diferentes particiones de los datos.

```
# Crear un solo árbol de decisión
clf = DecisionTreeClassifier(random_state=42)

# ============================
# 2. Repetir evaluación 50 veces
# ============================
for i in range(1, 51):
    # Dividir los datos en entrenamiento y prueba (aleatorio en cada iteración)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, shuffle=True
    )

    # Entrenar siempre el mismo árbol
    clf.fit(X_train, y_train)

    # Evaluar
    score = clf.score(X_test, y_test)
    y_pred = clf.predict(X_test)
    f1 = f1_score(y_test, y_pred)

    # Guardar métricas
    scores.append(score)
    f1_scores.append(f1)
```

4. Evaluación del Modelo

En cada ejecución, los datos se dividieron en **80% entrenamiento y 20% prueba**.

Se calcularon dos métricas principales:

- **Accuracy (precisión):** proporción de aciertos totales del modelo.
- **F1-Score:** equilibrio entre la precisión y la cobertura, especialmente útil en problemas donde una clase puede ser más difícil de detectar que la otra.

Además, se calcularon los **Z-scores del accuracy**, para analizar qué ejecuciones se alejaban más del comportamiento promedio.

5. Visualización de Resultados

Se generaron gráficas para observar:

- La evolución del accuracy y del F1-score en las 50 ejecuciones.
- La variación del Z-score del accuracy.
- Las características más utilizadas en forma de gráfico circular (pie chart).

Figura 1: Evolución del Accuracy y F1-score.

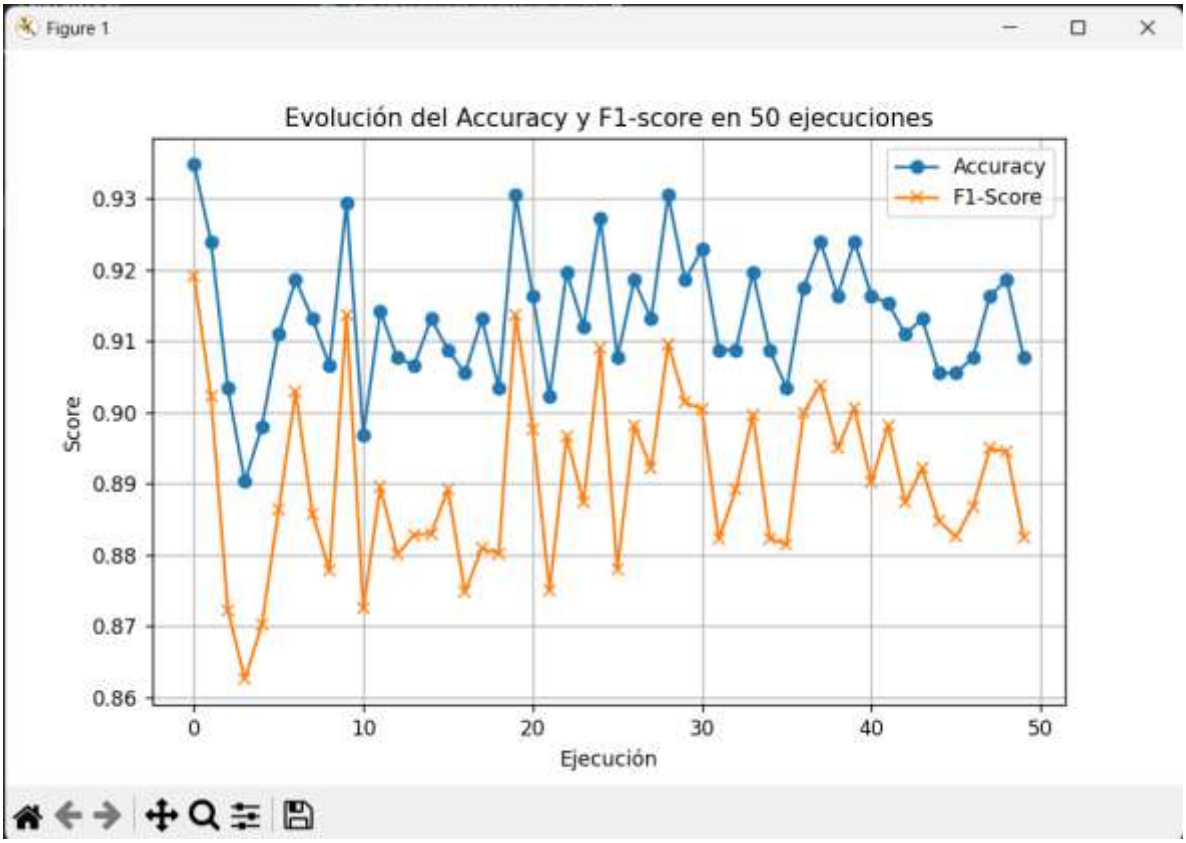


Figura 2: Z-score del Accuracy.

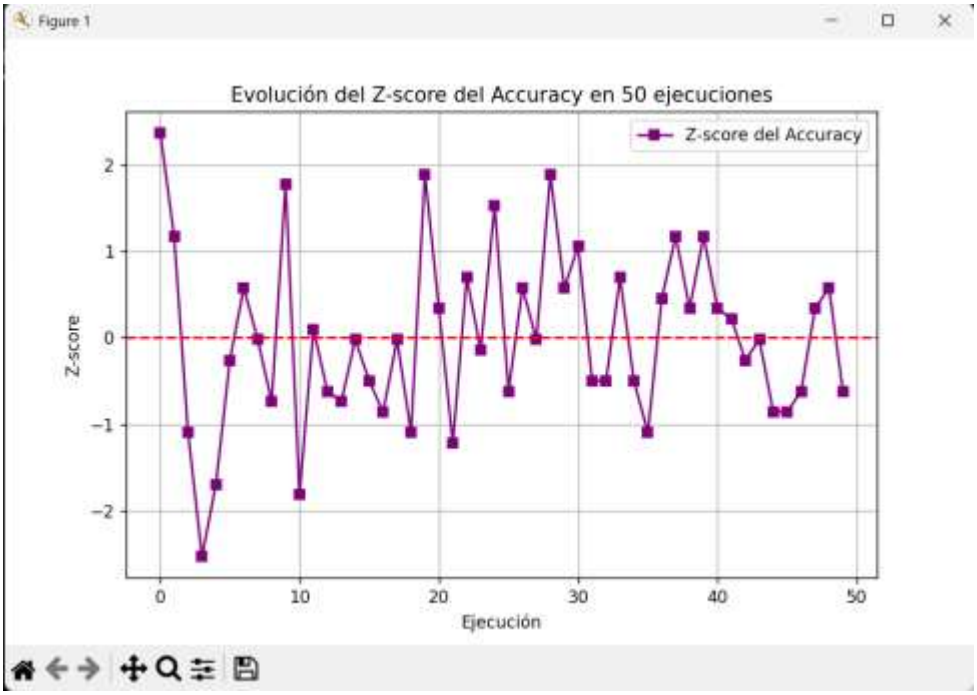
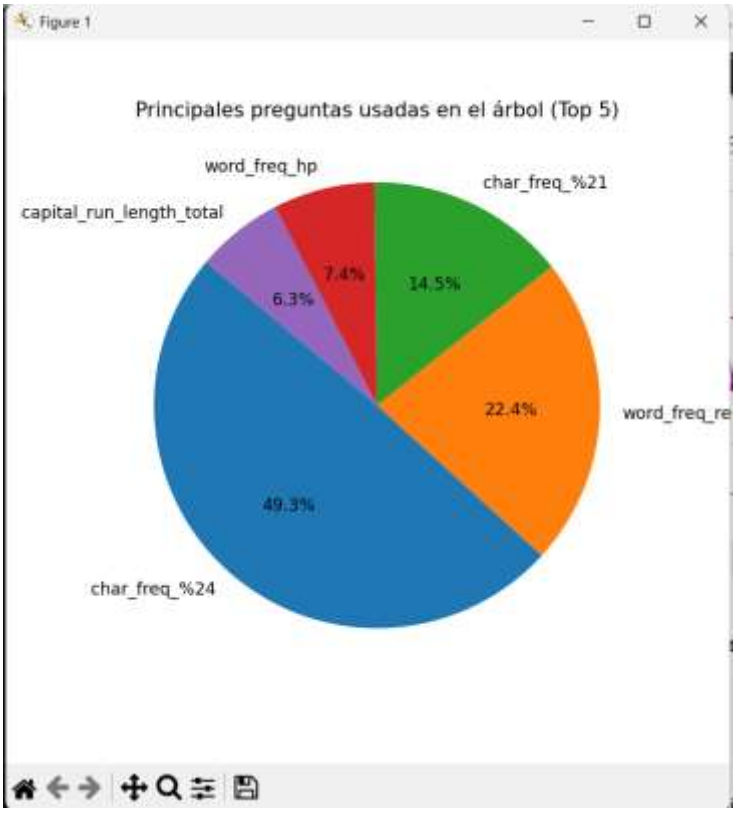


Figura 3: Gráfico circular de características importantes.



Descripción del Algoritmo

El **Árbol de Decisión** funciona dividiendo los datos en ramas a partir de preguntas basadas en características (ejemplo: "¿La frecuencia de la palabra *money* es mayor a cierto valor?").

Cada división busca separar mejor los datos en correos spam y ham.
El proceso continúa hasta que se alcanza una condición de parada:

- que no haya más mejoras, o
- que el nodo tenga muestras puras (todas de la misma clase).

El resultado final es una serie de reglas de clasificación fáciles de interpretar.

Ventajas:

- Intuitivo y fácil de interpretar.
- No requiere normalización de los datos.
- Puede manejar datos con muchas variables.

Limitaciones:

- Tiende a sobreajustarse (overfitting) si no se regulan los parámetros.
- Pequeñas variaciones en los datos pueden generar cambios grandes en la estructura del árbol.

Resultados Obtenidos

Evolución del Accuracy y F1-Score

- El accuracy se mantuvo alrededor del **91% en promedio**.
- El F1-score fue más variable, rondando el **88-89%**, lo cual refleja que detectar spam puede ser ligeramente más difícil que detectar ham.
- La dispersión indica que, aunque el modelo es estable, existen fluctuaciones según la partición de los datos.

```
📌 Principales preguntas usadas en el árbol (features más importantes):
1. char_freq_%24: 0.3357
2. word_freq_remove: 0.1527
3. char_freq_%21: 0.0987
4. word_freq_hp: 0.0502
5. capital_run_length_total: 0.0430
6. word_freq_free: 0.0384
7. word_freq_you: 0.0234
8. word_freq_edu: 0.0221
9. capital_run_length_average: 0.0210
10. word_freq_george: 0.0187
Resultados finales después de 50 ejecuciones:
Accuracy promedio: 0.9133
Desviación estándar del accuracy: 0.0091
F1-score promedio: 0.8903
Z-scores de los accuracy: [ 2.36510125  1.17181101 -1.09544044 -2.52738872 -1.69208555 -0.26013727
  0.57516589 -0.02147922 -0.73745337  1.76845613 -1.81141458  0.0978498
 -0.61812434 -0.73745337 -0.02147922 -0.49879532 -0.85678239 -0.02147922
 -1.09544044  1.88778515  0.33650785 -1.21476946  0.69449492 -0.14080825
  1.52979808 -0.61812434  0.57516589 -0.02147922  1.88778515  0.57516589
  1.05248199 -0.49879532 -0.49879532  0.69449492 -0.49879532 -1.09544044
  0.45583687  1.17181101  0.33650785  1.17181101  0.33650785  0.21717882
 -0.26013727 -0.02147922 -0.85678239 -0.85678239 -0.61812434  0.33650785
  0.57516589 -0.61812434]
```

Evolución del Z-score

- La mayoría de los valores de accuracy se mantuvieron cerca del promedio (línea base en 0).
- Hubo ejecuciones que se alejaron con valores positivos o negativos, mostrando mejoras o caídas de rendimiento en comparación con la media.

Características más importantes del Árbol

Entre las variables más influyentes se encuentran la frecuencia de ciertas palabras clave (como *money*, *hp*, *remove*) y el uso de letras en mayúscula.

Estas características resultan lógicas, ya que los correos spam suelen abusar de palabras asociadas con ventas, publicidad o promesas falsas.