PUCE Virtual

CLASE

5

Procesamiento de Lenguaje Natural

Evaluación del clasificador desde ciberseguridad

QS WORLD UNIVERSITY RANKINGS

Educación de calidad

INTRODUCCIÓN DE LA CLASE

GRADO / POSGRADO / TECNOLOGÍAS

Estudia a tu tiempo
son interrupciones

EDUCACIÓN EN LÍNEA DE CALIDAD

Durante las semanas anteriores del curso se desarrolló el primer sistema de clasificación de amenazas textuales, utilizando técnicas de procesamiento de lenguaje natural para transformar mensajes no estructurados —como correos electrónicos, tickets de soporte o reportes de incidentes— en representaciones vectoriales que pueden ser analizadas por algoritmos de aprendizaje automático. Sin embargo, entrenar un modelo de clasificación no es suficiente para garantizar que el sistema sea útil en un entorno real de ciberseguridad. En la práctica, es necesario evaluar cuidadosamente el comportamiento del modelo para determinar si sus decisiones contribuyen realmente a mejorar la detección temprana de amenazas. La Semana 5 se enfoca precisamente en este proceso de evaluación, analizando cómo interpretar las métricas de desempeño del clasificador desde la perspectiva operativa de un sistema de seguridad.

En esta semana se estudiarán métricas fundamentales como precision, recall y F1-score, que permiten comprender cómo se comporta un modelo frente a diferentes tipos de errores. A diferencia de otros dominios del aprendizaje automático, en ciberseguridad los errores tienen consecuencias muy distintas: un falso positivo puede generar carga adicional para los analistas de un Centro de Operaciones de Seguridad (SOC), mientras que un falso negativo puede permitir que un ataque real pase desapercibido. Por esta razón, también se analizarán conceptos como el impacto operativo de los errores, los umbrales de alerta y el costo del error, elementos clave para ajustar el comportamiento del sistema y definir el punto de operación más adecuado para un entorno de detección automática de amenazas.

RDA.2: Diseñar sistemas de procesamiento de lenguaje natural para detectar y clasificar amenazas presentes en comunicaciones digitales, a través de la construcción de modelos de clasificación de texto, análisis semántico e identificación de señales de ingeniería social.

Reto # 2

Evaluación del clasificador desde ciberseguridad.

Interpretación de precisión, recall y F1.

Una vez que el modelo de clasificación ha sido entrenado, es necesario evaluar su desempeño para determinar si realmente puede detectar amenazas en textos digitales. En muchos problemas de aprendizaje automático se utiliza la métrica accuracy, que mide el porcentaje total de predicciones correctas. Sin embargo, en ciberseguridad esta métrica puede ser engañosa, especialmente cuando el conjunto de datos está desbalanceado (muchos mensajes legítimos y pocos ataques).

Por esta razón se utilizan métricas más informativas: precision, recall y F1-score (Eisenstein,2019).

- **Precision:** es una métrica que responde a la siguiente pregunta: De todas las alertas generadas por el sistema, ¿cuántas eran realmente ataques?

  $$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

  Donde: TP (True Positive): ataques detectados correctamente, y FP (False Positive): alertas falsas. En un sistema SOC, una precisión baja implica muchas alertas innecesarias.

  Ejemplo: Suponiendo 100 mensajes analizados, 20 alertas generadas, y 15 ataques reales.

  $$\text{Precision} = \frac{15}{15+5} = 0.75$$

  Es decir, el 75% de alertas eran correctas.

- **Recall**: es una métrica que responde a otra pregunta crítica: ¿Cuántos ataques reales detectó el sistema?

  $$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

  Donde: FN (False Negative): ataques no detectados.

  Ejemplo: Del anterior, supongamos se detectaron 8.

  $$\text{Recall} = \frac{15}{15+7} = 0.68$$

  Esto significa que el 32% de los ataques pasaron desapercibidos. En ciberseguridad, el recall es especialmente importante porque los falsos negativos representan ataques exitosos.

- **F1-score**: El F1-score es una métrica que combina Precision y Recall:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Así, en el ejemplo anterior:

$F1 = 2 * \frac{0.75 * 0.68}{0.75 + 0.68} = 0.7142$

F1 es útil cuando es necesario equilibrar la detección de amenazas y la reducción de falsas alarmas.

A continuación la implementación básica en Python en la Ilustración 1, donde se puede observar que ya están definidas las métricas de evaluación comunes para un modelo de clasificación: precisión, recall y F1-score.

- **from sklearn.metrics import precision_score, recall_score, f1_score:** Esta línea importa las funciones necesarias de la librería sklearn.metrics para calcular las métricas.
- **precision = precision_score(y_test, y_pred):** La precisión mide la proporción de predicciones positivas correctas. Es decir, de todos los casos que el modelo predijo como positivos, ¿cuántos fueron realmente positivos? Un valor alto indica un bajo número de falsos positivos.
- **recall = recall_score(y_test, y_pred):** El recall (o sensibilidad) mide la proporción de casos positivos reales que fueron identificados correctamente por el modelo. Es decir, de todos los casos que eran realmente positivos, ¿cuántos detectó el modelo? Un valor alto indica un bajo número de falsos negativos.
- **f1 = f1_score(y_test, y_pred):** El F1-score es la media armónica de la precisión y el recall. Es una métrica útil cuando buscas un equilibrio entre precisión y recall, especialmente en conjuntos de datos desbalanceados. Combina ambos valores en una sola métrica.

En todas estas funciones, y_test son las etiquetas verdaderas del conjunto de prueba y y_pred son las predicciones del modelo para ese mismo conjunto de prueba. Los "print" imprimen los resultados de las mediciones.

```python
from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

Ilustración 1. Implementación Python de Precision, Recall y F1 (creación de autor Rafael Melgarejo)

Impacto de errores en SOC

Un SOC (Security Operations Center) es el equipo encargado de monitorear incidentes de seguridad en una organización. Los clasificadores de texto que detectan phishing, ingeniería social o malware generan alertas automáticas que deben ser analizadas por los especialistas (Jurafsky et al, 2023).

Los errores del modelo tienen consecuencias distintas:

Tipo de error	Significado	Impacto
Falso positivo	mensaje legítimo marcado como ataque	alerta innecesaria
Falso negativo	ataque clasificado como legítimo	ataque no detectado

Falsos positivos

Un sistema con demasiados falsos positivos gener fatiga de alertas. Problemas que genera:

- saturación del equipo SOC
- pérdida de confianza en el sistema
- aumento del tiempo de análisis

Falsos negativos

Los falsos negativos son más peligrosos. Consecuencias posibles:

- robo de credenciales
- infiltración de malware
- exfiltración de datos
- compromisos de infraestructura

Por esta razón, en ciberseguridad se prioriza detectar ataques incluso si se generan algunas falsas alarmas.

Trade-off Precision vs Recall en ciberseguridad

Con Precision alto hay pocas falsas alarmas, y Recall alto se detectan más ataques. Esto provoca en ciberseguridad una tensión entre ambas métricas:

- Si el sistema detecta demasiadas amenazas, aumenta el recall pero también los falsos positivos.
- Si el sistema es muy conservador, aumenta la precision pero algunos ataques no serán detectados.

En la ilustración 2, se aprecia que la zona izquierda tiene un Recall alto y Precisión bajo, entonces el SOC está saturado de alertas. Mientras, la zona derecha tiene Precision alto y Recall bajo, así los ataques pasan desapercibidos.

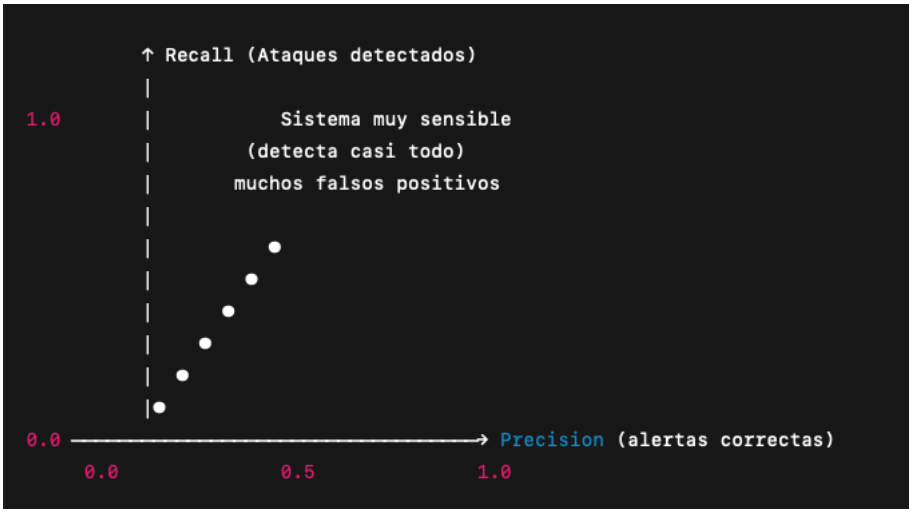

Ilustración 2. Trade-off Recall vs Precision (creación de autor Rafael Melgarejo)

El sistema útil para analistas SOC es equilibrio entre Precision y Recall. Usualmente en ciberseguridad se prioriza

$$\text{Recall} > \text{Precision}$$

Esto debido a que un falso positivo genera trabajo adicional, pero un falso negativo permite que un ataque ocurra. El objetivo es encontrar el punto operativo óptimo donde el sistema detecte suficientes ataques sin saturar al equipo SOC (ver Ilustración 3).

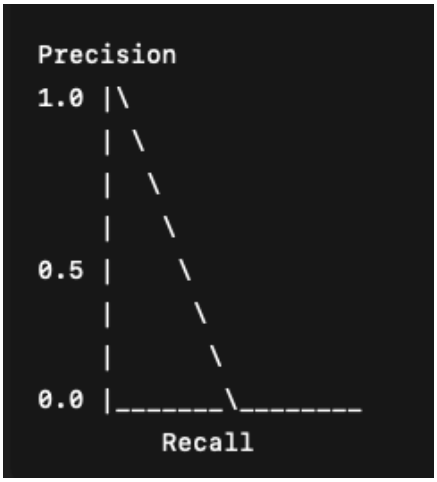

Ilustración 3. Equilibrio Precision vs Recall (creación de autor Rafael Melgarejo)

Umbrales de alerta

La mayoría de clasificadores no producen directamente una clase, sino una probabilidad de pertenecer a una clase, por ejemplo:

Texto	Probabilidad phishing	Umbral para phishing	Clasificación
“confirme su cuenta ahora”	.92	>0.5	Ataque
“error de conexión VPN”	.15	>0.5	legítimo

El umbral puede ajustarse según el riesgo. Sin embargo, el ajuste tiene ventajas y desventajas:

- Umbral bajo (e.g. 0.3)
    - Ventajas: detecta más ataques.
    - Desventajas: más falsas alarmas
- Umbral alto (e.g. 0.8)
    - Ventajas: menos falsas alertas
    - Desventajas: algunos ataques pasan desapercibidos.

Curva Precision-Recall

La curva Precision–Recall (PR) es una herramienta de evaluación utilizada en clasificación binaria para analizar el comportamiento de un modelo cuando se modifican los umbrales de decisión. En el contexto de ciberseguridad, esta curva permite comprender cómo cambia el equilibrio entre la capacidad de detectar ataques reales (recall) y la confiabilidad de las alertas generadas (precision). Cada punto de la curva corresponde a un valor diferente del umbral de clasificación aplicado al modelo (Jurafsky et al, 2023).

En un sistema de detección de amenazas, el clasificador suele generar una probabilidad de que un mensaje sea malicioso. Si el sistema considera amenaza cualquier probabilidad mayor a un cierto umbral (por ejemplo, 0.5), generará una decisión automática. Sin embargo, modificar ese umbral cambia el comportamiento del sistema:

- Umbral bajo, el sistema tiende a marcar muchos mensajes como ataques, aumentando la detección de amenazas pero también generando numerosas alertas falsas.
- Umbral es muy alto, el sistema genera pocas alertas, pero corre el riesgo de dejar pasar ataques reales.

La curva Precision–Recall muestra este comportamiento. En el eje horizontal se representa el recall, que indica la proporción de ataques reales detectados por el sistema. En el eje vertical se representa la precision, que mide la proporción de alertas que corresponden efectivamente a ataques. En general, al aumentar el recall suele disminuir la precision, lo que evidencia el trade-off entre ambas métricas (ver Ilustración 4).

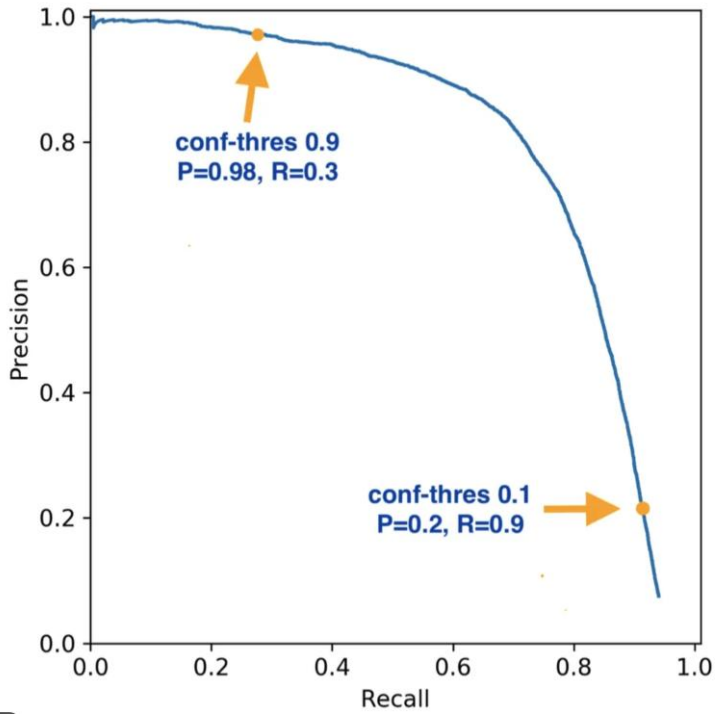

Ilustración 4. Curva Precision-Recall con diferentes umbrales (contenido de autor Rafael Melgarejo)

En la Ilustración 5 pueden identificarse cuatro zonas interpretativas que ayudan a comprender el comportamiento del sistema en un entorno SOC:

- Zona de alta precision y bajo recall: En esta región el sistema genera pocas alertas y la mayoría son correctas. Sin embargo, el modelo detecta solo una parte de los ataques existentes. Desde el punto de vista operativo, esta configuración puede resultar peligrosa porque muchos ataques reales podrían pasar desapercibidos.
- Zona de equilibrio: Representa el punto operativo deseable del sistema, donde se logra un balance entre detectar amenazas y mantener una carga manejable de alertas para los analistas. Un modelo que opere en esta región logra detectar una proporción significativa de ataques sin saturar el SOC con falsas alarmas.
- Zona de alto recall y precision moderada: Aquí el sistema detecta una gran proporción de ataques, pero aumenta el número de alertas incorrectas. Aunque esto implica mayor trabajo para los analistas, en muchos sistemas de seguridad se prefiere esta configuración porque es mejor investigar alertas adicionales que dejar pasar un ataque real.
- Zona de baja precision y alto recall extremo: En esta región el sistema marca casi todos los mensajes como sospechosos. Esto genera una cantidad excesiva

de alertas y puede provocar fatiga de alertas en el SOC, reduciendo la capacidad del equipo para responder a incidentes reales.

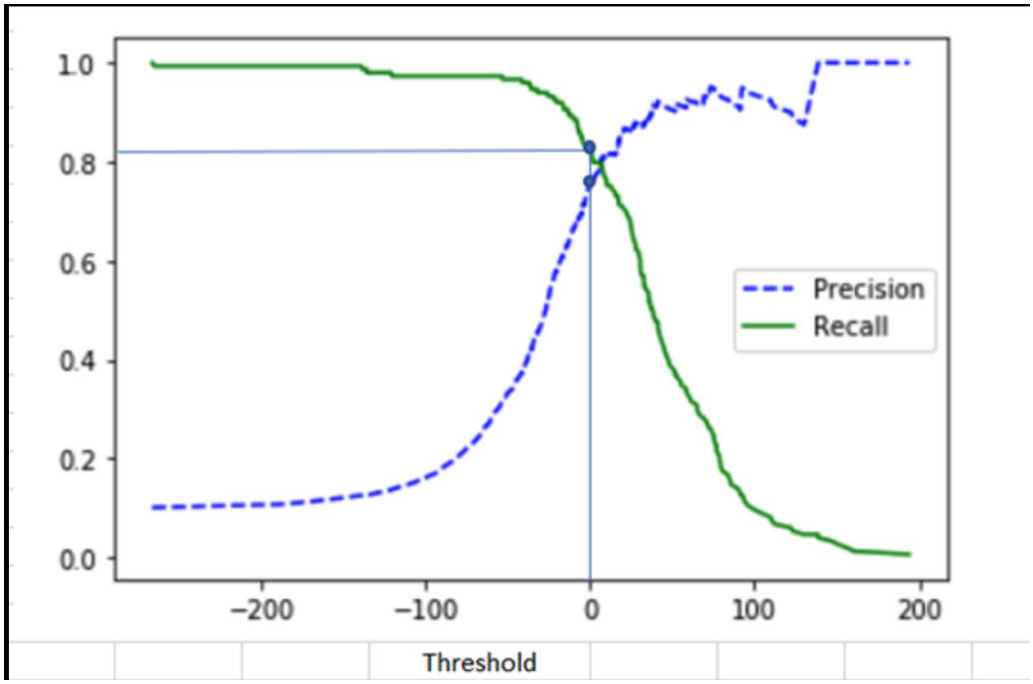

Ilustración 5. Zonas de la Curva Precision-Recall (contenido de autor Rafael Melgarejo)

En síntesis, la curva Precision–Recall permite identificar el punto de operación óptimo del modelo, es decir, el umbral que ofrece el mejor equilibrio entre detección de ataques y carga operativa para los analistas. En sistemas de ciberseguridad suele priorizarse un recall relativamente alto, ya que los falsos negativos —ataques no detectados— representan el mayor riesgo para la organización.

Se recomienda revisar el material de profundización de la semana 5, en la sección que pide cambiar los umbrales de precisión. También se recomienda revisar el video "PRECISION, RECALL Y F-SCORE: ¿qué son y cuándo usarlos?" de "Codificando Bits" en https://www.youtube.com/watch?v=H8FSfqxRWmA.

Implementación Python

El código de la Ilustración 6 toma las probabilidades de predicción de un modelo y las convierte en etiquetas binarias (0 o 1) utilizando un umbral:

- probs = model.predict_proba(X_test)[:,1]: Esta línea calcula las probabilidades de que cada instancia en el conjunto de prueba (X_test) pertenezca a cada clase. model.predict_proba() devuelve un array donde cada fila corresponde a una instancia y cada columna a la probabilidad de pertenecer a una clase específica. Al añadir [:,1], selecciona solo la columna correspondiente a la probabilidad de la clase positiva (generalmente representada como '1').

- threshold = 0.6: Aquí se define un umbral, en este caso, 0.6. Este valor se utilizará para decidir si una probabilidad predicha debe considerarse como clase positiva o negativa. Si un modelo predice que una instancia tiene una probabilidad superior a este umbral de ser de la clase positiva, se clasificará como tal.
- predictions = (probs > threshold).astype(int): Esta es la parte clave donde se aplican las reglas de decisión. Para cada probabilidad en probs, se compara si es mayor que el threshold. Esto crea un array de valores booleanos (True/False). Luego, .astype(int) convierte estos valores booleanos en enteros: True se convierte en 1 (clase positiva) y False se convierte en 0 (clase negativa). De esta manera, obtenemos las predicciones binarias finales basadas en el umbral.

```
probs = model.predict_proba(X_test)[:,1]

threshold = 0.6

predictions = (probs > threshold).astype(int)
```

Ilustración 6. Umbral de Alerta en Python (creación de autor Rafael Melgarejo)

Trade-off en SOC

La ilustración 6 resume el Trade-off visualizando las diferencias entre SOC Saturado, SOC seguro, y SOC vulnerable.

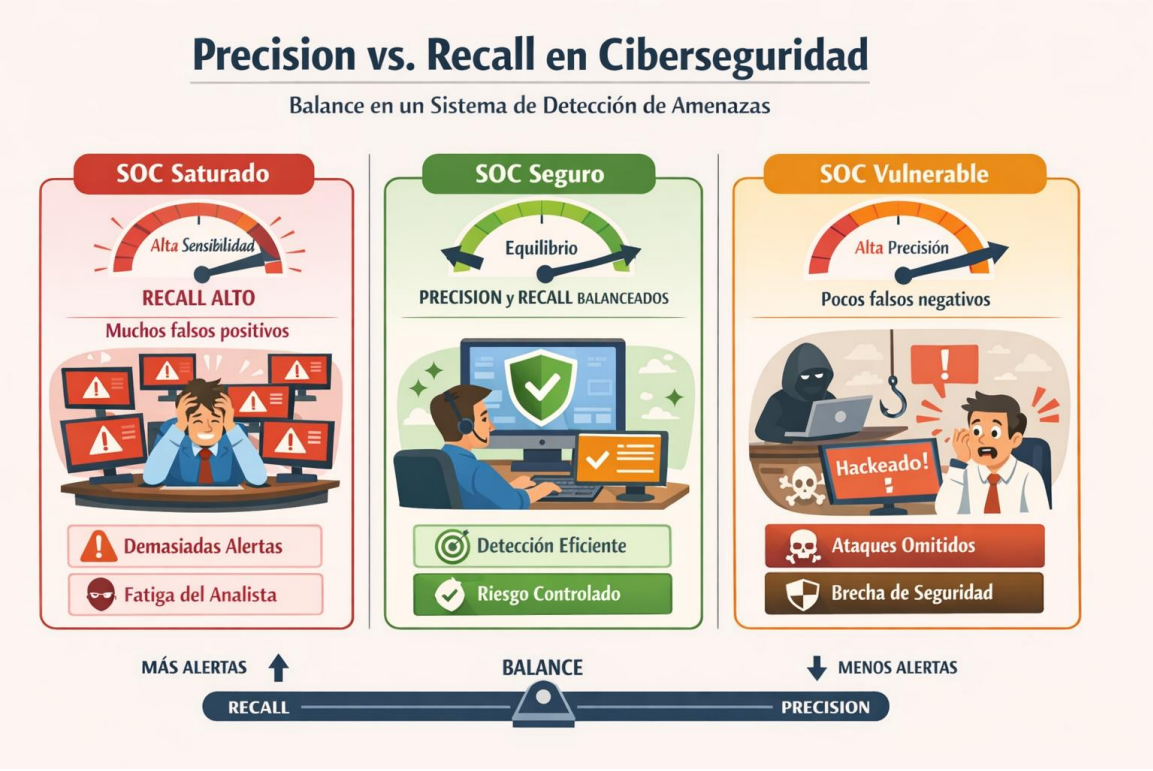

Ilustración 7. Precision vs Recall en Ciberseguridad, elaborado en ChatGPT en base a (Jurafsky,2023)

Costo del error.

En ciberseguridad no todos los errores tienen el mismo impacto. Por ejemplo, un Falso Positivo FP (analista revisa alerta necesaria) tiene el costo de 1 (una unidad), mientras un error de Falso Negativo FN (ataque exitoso), el error tiene un costo de 50 unidades. Esto refleja que un ataque no detectado es mucho más costoso que una alerta falsa.

En algunos sistemas se utilizan modelos sensibles al costo, donde el algoritmo penaliza más los falsos negativos.

Ilustración 8 muestra un código Python que define un diccionario llamado class_weight. En el contexto del aprendizaje automático, especialmente en problemas de clasificación, este diccionario se utiliza para asignar diferentes pesos a las clases:

- class_weight = { "legitimo":1, "phishing":5 }: Esta línea crea un diccionario **donde:**
    - **legitimo":** 1 significa que a la clase 'legítimo' se le asigna un peso de 1.
    - **"phishing":** 5 significa que a la clase 'phishing' se le asigna un peso de 5.

¿Para qué se usa?

Este tipo de diccionario es crucial cuando se trabaja con conjuntos de datos desbalanceados. Por ejemplo, si existen muchas más instancias de 'legítimo' que de 'phishing', el modelo podría tender a clasificar la mayoría de los casos como 'legítimo' para lograr una alta precisión general, pero fallando en detectar la clase minoritaria (phishing, que a menudo es la más importante de detectar).

Al asignar un peso mayor a la clase minoritaria (phishing con peso 5), se le está diciendo al algoritmo de entrenamiento que le dé más importancia a la correcta clasificación de las instancias de 'phishing'. Un error al clasificar una instancia de 'phishing' tendrá un "costo" 5 veces mayor que un error al clasificar una instancia 'legítima'. Esto ayuda a que el modelo preste más atención a la clase minoritaria y mejore su rendimiento en la detección de esta clase, incluso si la precisión general podría disminuir ligeramente.

Ilustración 8. Costo del error (creación de autor Rafael Melgarejo)

Para conocer más acerca del costo del error, y cómo evaluarlo, revise el video

“EVALUANDO EL ERROR EN LOS MODELOS DE CLASIFICACIÓN | #33 Curso Machine Learning con Python” de “AprendeIA”, en https://www.youtube.com/watch?v=fqfYGrkj3bo

Glosario:

- **Eisenstein, J. (2019).** Introduction to natural language processing. MIT Press. https://cdn.jsdelivr.net/gh/it-ebooks-0/it-ebooks-2018-04to07/Natural%20Language%20Processing%20%28Jacob%20Eisenstein%29.pdf

- **Jurafsky, D., & Martin, J. H. (2023).** Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (3rd ed., draft). Stanford University. https://web.stanford.edu/~jurafsky/slp3/.

PUCE Virtual

síguenos en:

www.pucevirtual.puce.edu.ec