Informe: Clasificación de Sitios Web con Phishing

1. Descripción del problema

El **phishing** es una técnica de ingeniería social en la que los atacantes publican sitios web que imitan a servicios legítimos con el objetivo de obtener credenciales, datos bancarios u otra información sensible. Para el usuario final, la diferencia entre un sitio genuino y uno malicioso suele ser imperceptible, por lo que la **detección automática** resulta fundamental: un filtro algorítmico permite bloquear la mayoría de los intentos antes de que lleguen a los usuarios, reduciendo el riesgo de fraude a escala.

2. Distribución de clases e impacto

El dataset de UCI contiene **11 055 instancias** con la variable objetivo `Result` codificada como -1 (sitio legítimo) y 1 (sitio de phishing). La distribución observada es 44.31 % de sitios legítimos y 55.69 % de phishing, lo que implica un desbalance moderado.

Ese leve predominio de la clase positiva influye en las métricas:

- Un **accuracy** alto puede convivir con un número importante de falsos negativos si el modelo se limita a seguir la clase mayoritaria.
- El **recall** de la clase positiva es crítico: cada falso negativo implica que un sitio fraudulento logró evadir el detector.
- En un sistema real, un exceso de falsos negativos expone a los usuarios a ataques, mientras que demasiados falsos positivos degrada la experiencia bloqueando sitios legítimos. Por eso el informe enfatiza el equilibrio entre Precision y Recall.

La figura `distribucion_clases_notebook.png` (generada en el cuaderno) resume visualmente esta proporción y sirve de referencia para el análisis posterior.

3. Reproducibilidad y control de aleatoriedad

Para garantizar que los resultados puedan replicarse exactamente se utilizó una **semilla global** `random_state = 42` en todos los pasos estocásticos:

- La partición `train_test_split` mantiene fija la selección de ejemplos en entrenamiento y prueba.
- El `StratifiedKFold` empleado durante el ajuste de hiperparámetros vuelve determinísticos los folds de la validación cruzada.

- La búsqueda aleatoria de características (`random.sample`) también recibe la misma semilla para explorar siempre las mismas combinaciones.

Definir la semilla permite contrastar versiones del pipeline, validar experimentos y compartir el flujo con otros miembros del equipo sin discrepancias numéricas.

4. Elección del método de validación

Se combinan dos estrategias complementarias:

- **Hold-out estratificado 80/20:** asegura que entrenamiento y prueba respeten la distribución original de clases (≈44/56). Con más de 10 000 muestras, esta división es suficiente para evaluar el rendimiento final.
- **Validación cruzada estratificada (5 folds):** durante la búsqueda de `var_smoothing` , `GridSearchCV` utiliza `StratifiedKFold(shuffle=True, random_state=42)` para promediar el desempeño en diferentes pliegues. Esto reduce la varianza de la estimación y evita que la elección del hiperparámetro dependa de una única partición.

Si el dataset fuese significativamente menor, se priorizaría la validación cruzada para aprovechar mejor los datos; con datasets más grandes, la validación simple es menos costosa y aún representativa.

5. Selección y optimización de hiperparámetros

El algoritmo **Gaussian Naive Bayes** posee un hiperparámetro principal: `var_smoothing` , que añade una constante a la varianza de cada característica. Su ajuste es relevante porque:

- Valores muy pequeños pueden amplificar el ruido en atributos con variancia reducida.
- Valores grandes suavizan en exceso las distribuciones y disminuyen la capacidad de discriminación.

El proceso seguido fue:

1. **Búsqueda aleatoria de características:** 500 iteraciones seleccionando subconjuntos de tamaño 5 con `random_state=42` . Las cinco variables más frecuentes en los óptimos locales fueron:
   - `SSLfinal_State`
   - `having_At_Symbol`
   - `having_IP_Address`
   - `age_of_domain`
   - `URL_of_Anchor`

2. **GridSearchCV sobre `var_smoothing`:** rango logarítmico entre 1e-9 y 1e-3 (10 valores). El criterio de optimización fue el **F1-score**, priorizando un balance entre Precision y Recall.
3. **Mejor configuración encontrada:** `var_smoothing = 1e-09`. Este valor coincide con el extremo inferior del rango, confirmando que el modelo base ya operaba cerca de su óptimo cuando se utilizan las variables más informativas.

Los resultados comparativos (obtenidos de `Data/resultados_notebook/comparacion_metricas_notebook.csv`) pueden resumirse así:

- **Modelo base**
  - Accuracy **0.5975**
  - Precision **1.0000**
  - Recall **0.2770**
  - F1-score **0.4338**
  - AUC-ROC **0.9702.**
- **Modelo optimizado**
  - Accuracy **0.8969**
  - Precision **0.9002**
  - Recall **0.9163**
  - F1-score **0.9082**
  - AUC-ROC **0.9445.**
- **Variación**
  - El recall mejora **+0.6393 (≈ +231 %)**
  - El F1-score aumenta **+0.4744 (≈ +109 %)**
  - Mientras que el AUC-ROC cae levemente **−0.0257 (≈ −2.65 %)** al priorizar sensibilidad.

6. Resultados y conclusiones

- **Modelo base:** al entrenar GaussianNB con todas las variables, el clasificador prioriza la precisión (1.00) a costa de un **recall de 0.277**. La matriz de confusión muestra que clasifica casi todo como phishing (TP=341, FN=890, TN=980, FP=0). Este comportamiento evita los falsos positivos pero deja escapar la mayoría de los sitios maliciosos, algo inaceptable en producción.
- **Modelo optimizado:** con las cinco variables seleccionadas y `var_smoothing = 1e-09`, el modelo alcanza **recall 0.916**, **precision 0.900** y **F1-score 0.908**. La matriz de confusión (TN=855, FP=125, FN=103, TP=1 128) evidencia una reducción drástica de falsos negativos (de 890 a 103), asumiendo un incremento moderado de falsos positivos, lo cual es preferible en este dominio.

- **Curva ROC:** el AUC-ROC pasa de 0.970 en el modelo base a 0.944 en el modelo optimizado, una ligera disminución explicada por el nuevo equilibrio entre sensibilidad y especificidad. Aun así, ambos valores indican alta capacidad de discriminación.

En síntesis, la combinación de selección de características y ajuste de `var_smoothing` ofrece un compromiso más adecuado para la detección de phishing: se maximizan los verdaderos positivos con una tasa de falsos positivos manejable. El modelo final es apto para desplegarse en un escenario real, siempre acompañado de políticas de revisión periódica y, eventualmente, de ajustes en el umbral de decisión para adaptar la sensibilidad del sistema a diferentes contextos de riesgo.