ChatSentinel

Presentamos ChatSentinel, una solución contra el grooming online.

Integrantes:

- Javier Cadagan Parra
- Gabriel Huerta Torres
- Diego Oyarzo Navia

La Amenaza Invisible del Grooming Online

El grooming online es un riesgo creciente. Depredadores explotan la confianza de menores para fines dañinos. Los niños, vulnerables, necesitan nuestra protección constante.

Problemática

El grooming online es una amenaza creciente. Individuos malintencionados contactan a menores.

Nuestra Misión

Desarrollar un sistema de detección automática de este abuso.

Protegemos a los niños en línea.

ChatSentinel: Un Escudo Digital

ChatSentinel es un sistema inteligente. Detecta conversaciones de grooming. Usa procesamiento de lenguaje natural y machine learning.

Análisis Exploratorio Inicial

Comenzamos con una investigación profunda de patrones.

Implementación de Modelos

Desarrollamos tres modelos de clasificación.

La Base de Nuestro Modelo: Los Datos

Nuestra investigación se basa en datos clave. Usamos dos conjuntos de datos para entrenar a ChatSentinel.

Dataset Principal: ChatCoder 2.0

- Fuente: ChatCoder HomePage (Wayback Machine)
- Formato: 56 archivos XML
- Estructura: user_id, datetime, message, role

Dataset Complementario: DailyDialog

- Fuente: Hugging Face platform
- Propósito: Conversaciones cotidianas
- Clase: "no grooming" (diálogos cortos)
- Procesamiento: Concatenación de diálogos

Limpieza y Normalización de Datos

El procesamiento de datos fue crucial. Superamos desafíos técnicos y lingüísticos. Esto aseguró un dataset balanceado y preciso.

1

Manejo de XML

Expresiones regulares para caracteres inválidos.

2

Estandarización de Tiempo

Fechas y horas a formato consistente.

3

Mensajes Vacíos

Eliminación y normalización para consistencia.

4

Normalización Lingüística

Diccionario de slangs y contracciones.

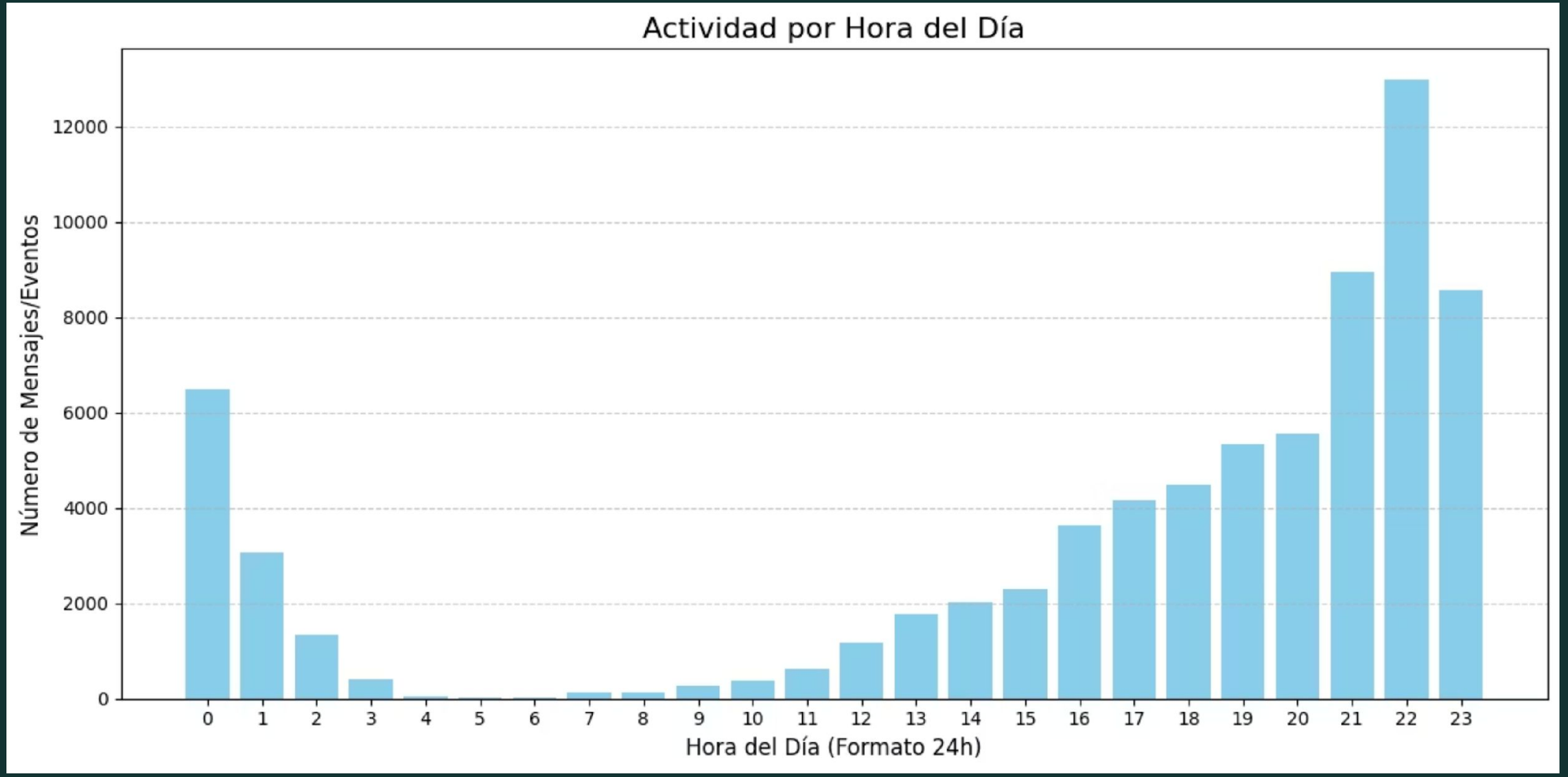
```
    r"\bthx\b": "thanks",
    r"\bomg\b": "oh my god",
    r"\bcauz\b": "because",
    r"\bcuz\b": "because",
    r"\bcomdom\b": "condom",
    r"\bpreggerz\b": "pregnant",
    r"\bmin\b": "minutes",
    r"\bscard\b": "scared",
    r"\bhafta\b": "have to",
    r"\bprof\b": "profile",
    r"\bpromis\b": "promise",
    r"\bcallin\b": "calling",
    r"\bhav\b": "have",
    r"\bno\b": "no",
    r"\bsooo\b": "so",
    r"\byeah\b": "yes",
    r"\bwait\b": "wait",
}

def normalize_text(text):
    if not isinstance(text, str):
        return ""

    #Convertir a minúsculas
    text = text.lower()

    #Expandir contracciones con la librería contractions
    text = contractions.fix(text)

    #Reemplazar slangs usando diccionario
    for pattern, replacement in replacement_dict.items():
        text = re.sub(pattern, replacement, text)

    text = re.sub(r'[^a-z0-9\s]', ' ', text) #Elimina caracteres no alfabéticos
    text = re.sub(r'\s+', ' ', text).strip()  #Elimina espacios extras

    return text
```

```
def parse_chatlog(xml_file_path):
    # lee el archivo xml
    with open(xml_file_path, 'r', encoding='utf-8', errors='ignore') as file:
        content = file.read()

    # envuelve el contenido del body con cdata para preservar caracteres
    content_cdata = re.sub(
        r'<BODY>(.*?)</BODY>',
        r'<BODY><![CDATA[\1]]></BODY>',
        content,
        flags=re.DOTALL | re.IGNORECASE
    )

    # parsea el contenido modificado
    root = ET.fromstring(content_cdata)

    # extrae usernames de predadores y victimas
    predator_usernames = {sn.findtext('USERNAME') for sn in root.findall('.//PREDATOR/SCREENNAME')}
    victim_usernames = {sn.findtext('USERNAME') for sn in root.findall('.//VICTIM/SCREENNAME')}
```

Dataset DailyDialog

- Concatenar conversaciones (40 en 40) → promedio caracteres “full chat” similar (aprox. 14 000 vs 17 000)
- Cantidad de full chats similares (47 vs 56)
- Muestra aleatoria del 20 %

Patrones Temporales de Actividad



Analizamos la actividad por hora del día. Observamos picos en horas de menor supervisión parental.

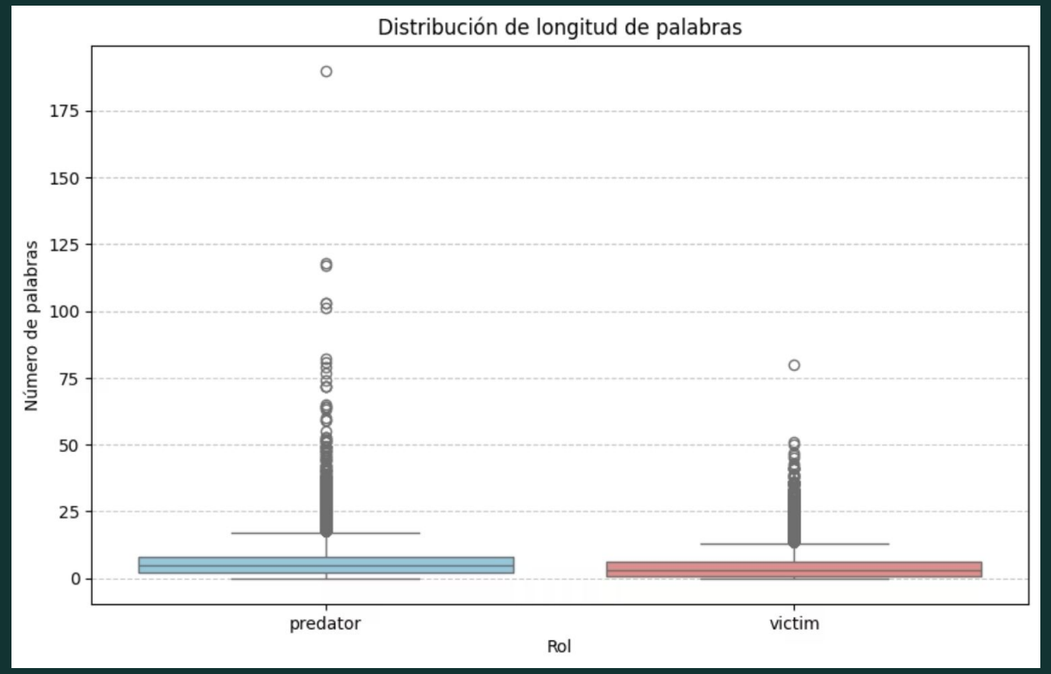
Descubriendo Patrones Lingüísticos

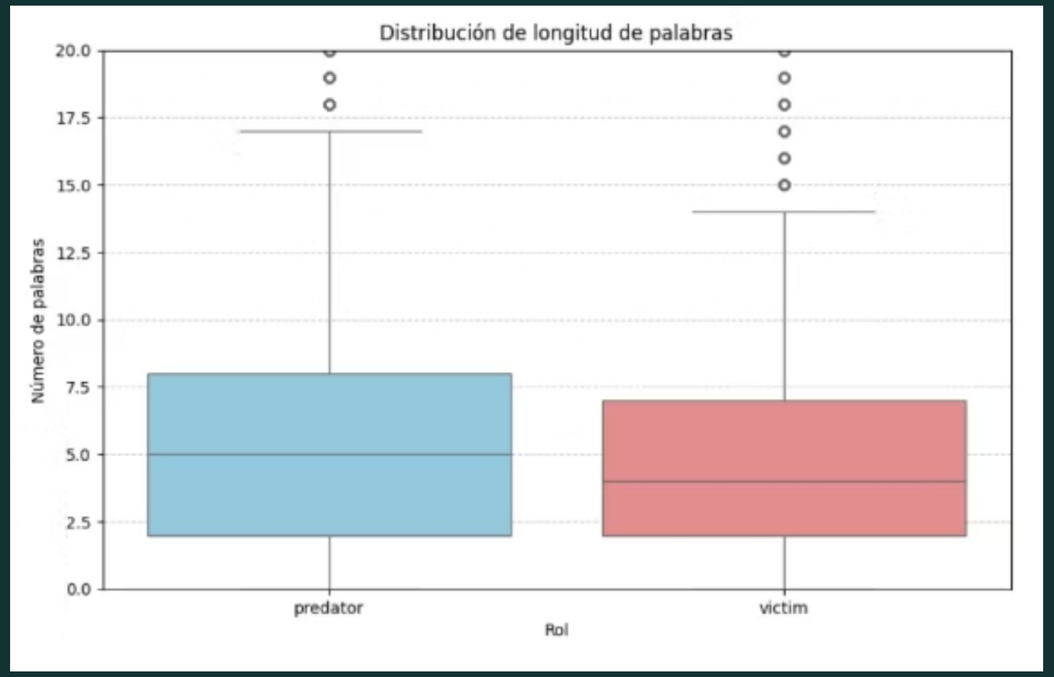
Identificamos diferencias clave en el lenguaje. Los predadores usan mensajes más largos.

Longitud de Mensajes

Víctimas: mensajes más cortos y frecuentes. Predadores: mayor variabilidad.

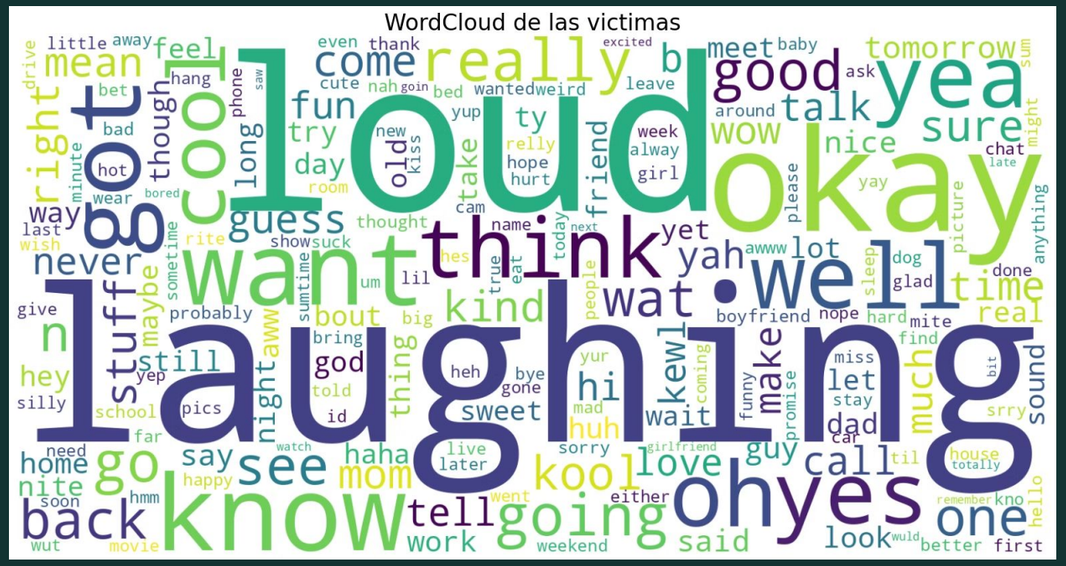
Los predadores buscan manipular y ganar confianza con mensajes extensos.





Análisis Léxico

Las nubes de palabras revelan vocabulario distintivo. Esto confirma patrones detectables.



Modelo de Clasificación primer enfoque

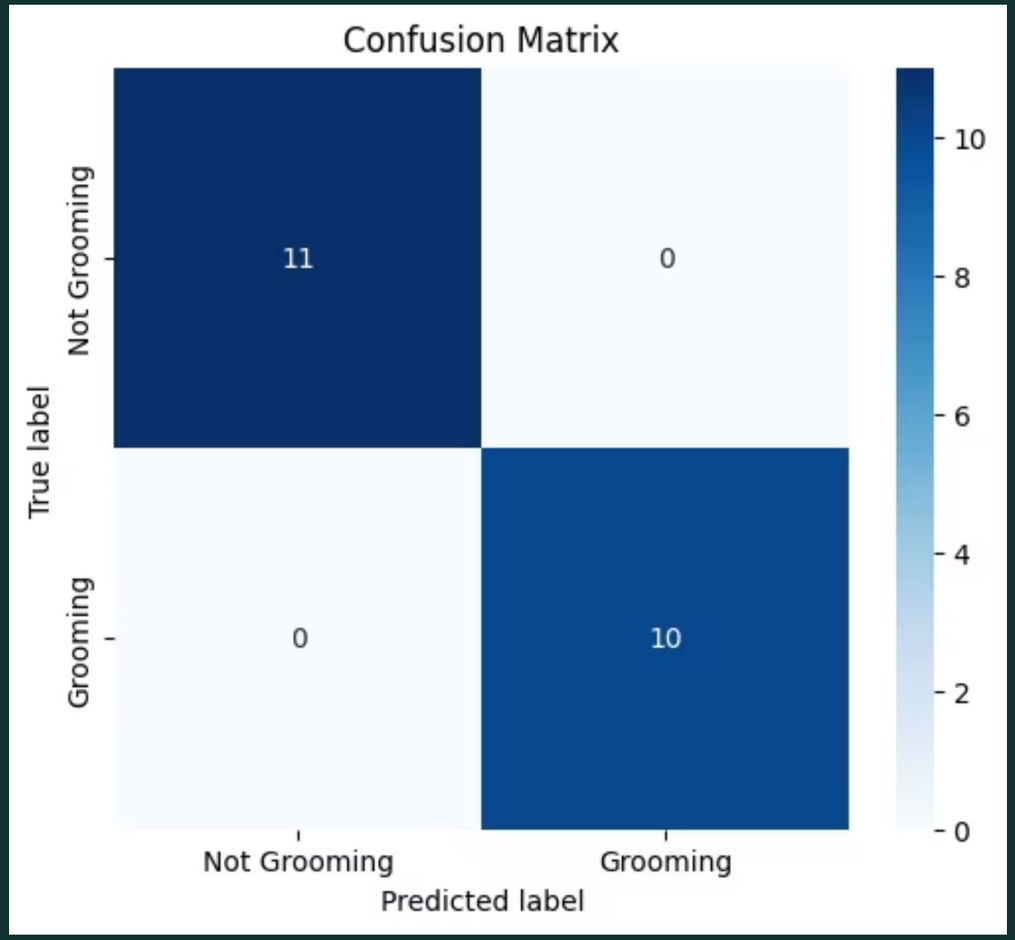
Regresión Logística

- Máximo de iteraciones: 10,000
- random_state=42

Se usó Random Forest

- n_estimators: 1000
- max_depth = 10 (para evitar overfiting)

Matriz de Confusión



Resultados del primer enfoque

El modelo actual es prometedor. Sin embargo, el sobreajuste es un desafío.

Modelo	Precision (Grooming)	**Recall (Grooming)**	**F1-Score (Predators)**
Regresión Logística	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00

Precisión general del 100% en conjunto de prueba. Esto indica un sobreajuste. Debemos mejorar la generalización del modelo.

Enfoque Final: Clasificación de Roles

Grooming or Not Grooming → Predator or Victim

Evitar fuga de datos

Para garantizar la integridad del modelo y evitar la fuga de información, se utilizó **GroupShuffleSplit**. Este método divide los datos por chat completo, y no por mensaje individual, asegurando que todas las interacciones de una misma conversación pertenezcan exclusivamente a un único conjunto, ya sea de entrenamiento o de prueba.

Conjunto de Entrenamiento (Train)

Aquí se agrupan conversaciones completas para que el modelo aprenda de patrones y características.

Chat A

Groomer - Víctima

Chat C

Groomer - Víctima

Conjunto de Prueba (Test)

Este conjunto contiene conversaciones completas nunca vistas por el modelo, utilizadas para evaluar su rendimiento.

Chat B

Groomer - Víctima

Chat D

Groomer - Víctima

Modelos de Clasificación segundo enfoque

Se usaron 3 modelos de clasificación para medir los resultados del modelo con las siguientes características

Regresión Logística

- Máximo de iteraciones: 10,000
- random_state=42

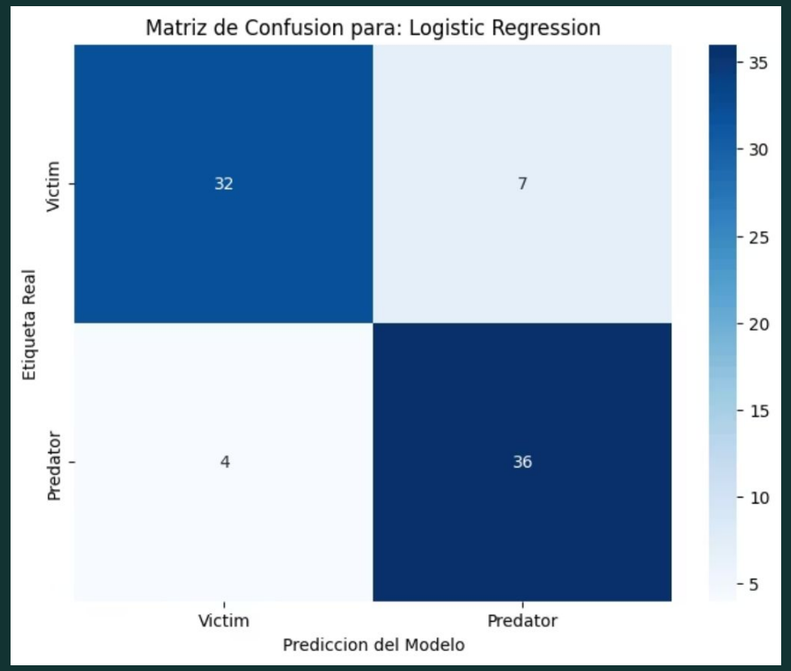
Se usó Random Forest

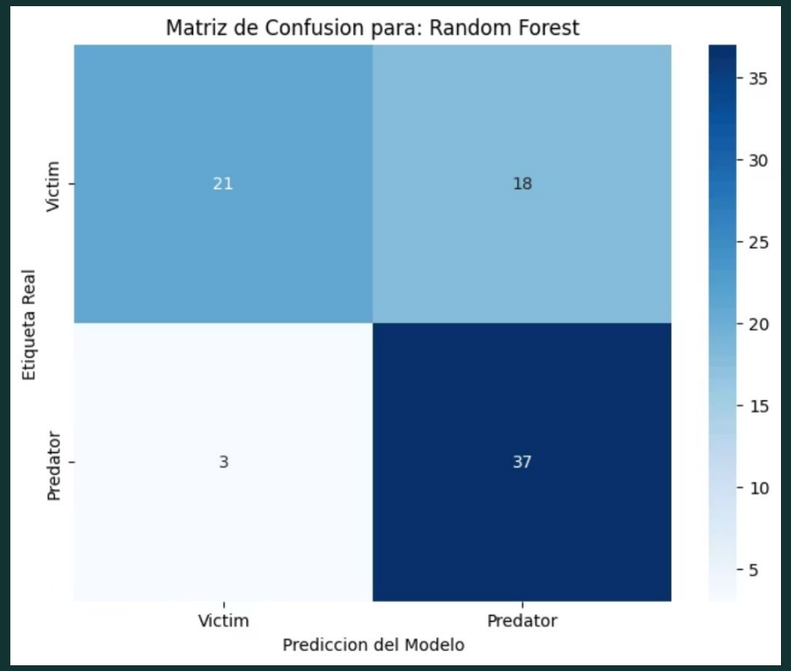
- n_estimators: 100
- random_state = 42

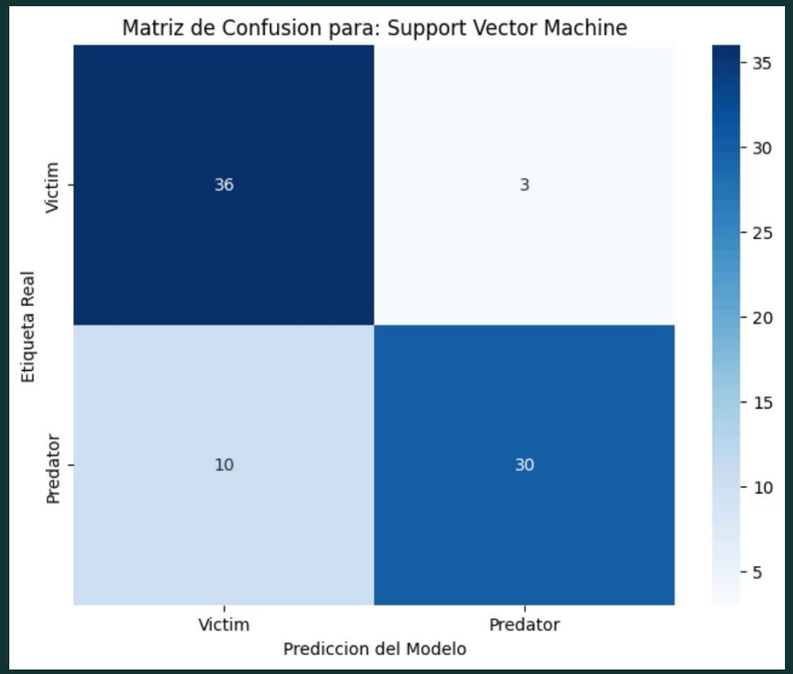
Support Vector Machine

- kernel = 'linear'
- random_state=42

Matrices de Confusión para la Clasificación de Roles







Resultado segundo enfoque

Modelo	Precision (Predators)	**Recall (Predators)**	**F1-Score (Predators)**
Regresión Logística	0.84	**0.90**	**0.87**
Support Vector Machine	**0.91**	0.75	0.82
Random Forest	0.67	0.92	0.78

La tabla muestra las métricas de rendimiento de los modelos evaluados para la clasificación de roles, donde lo que mas nos interesa es detectar a los Predators, junto con un buen rendimiento del modelo.

Selección del Modelo Ganador

Visualizando Errores Críticos

El error más crítico para nuestro caso es el Falso Negativo, que representa un "Predator" no detectado, con consecuencias potencialmente severas.

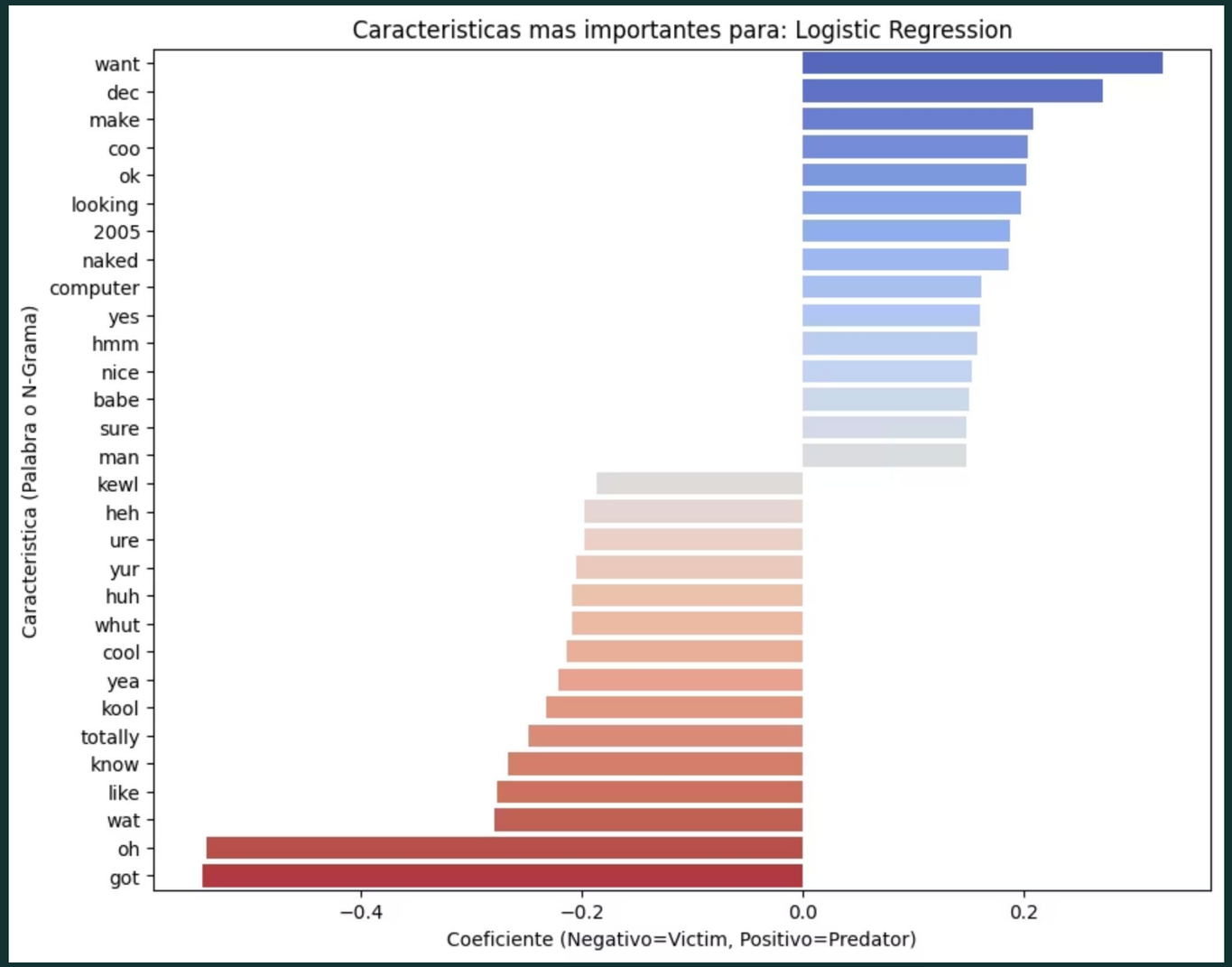
La Matriz de Confusión para Regresión Logística, con solo 4 Falsos Negativos, demuestra el mejor equilibrio entre la detección y la minimización de errores.

Regresión Logística: La Mejor Elección

Basado en los resultados, la **Regresión Logística** es seleccionada como el modelo final para ChatSentinel. Su equilibrio entre un alto recall y una buena precisión maximiza la detección de amenazas sin generar una cantidad excesiva de falsas alarmas.

La priorización del recall en la clase "Predator" es fundamental para la seguridad del sistema, asegurando que la mayoría de los casos peligrosos sean identificados.

Importancia de características Logistic Regression



Propuestas de Mejora Inmediatas del Modelo

Ajuste del Umbral de Decisión

El umbral de probabilidad por defecto (0.5) puede ajustarse. Analizando la curva Precision-Recall, se podría seleccionar un umbral que maximice el recall, utilizando métricas como el F2-Score, que prioriza el recall el doble que la precisión.

Optimización de Hiperparámetros

Los modelos se entrenaron con parámetros por defecto. Una búsqueda sistemática de hiperparámetros utilizando **GridSearchCV** o **RandomizedSearchCV** podría mejorar los resultados para el **TfidfVectorizer** y el clasificador de Regresión Logística.

Estas mejoras pueden implementarse directamente sobre el "pipeline" existente para optimizar su rendimiento de manera significativa.

Mejoras Avanzadas y Futuras Direcciones

Ingeniería de Características y Técnicas de NLP

Incorporar **word embeddings** (ej. Word2Vec, GloVe) para representar palabras como vectores. Además de métricas de emoción de mensajes.

Adopción de Modelos de Lenguaje Profundo

Implementar modelos pre-entrenados basados en Transformers (ej. **BERT**) que entienden el contexto completo del texto.

Hacia un Sistema de Detección en Tiempo Real

Desplegar el modelo en un entorno controlado para validación continua y medición de latencia.

- Estas propuestas implican cambios más profundos en la metodología para capturar patrones complejos y avanzar hacia un sistema de detección más sofisticado.

Conclusiones y Reflexiones Finales

Logros del Proyecto

- **Iteración Metodológica Exitosa:** El proyecto permitió identificar y corregir fallas iniciales, derivando en una solución superior.
- **Modelo de Clasificación Robusto:** Un recall del 90% y F1-Score de 0.87 para la detección de "predators" en Regresión Logística.
- **Validación de Patrones Lingüísticos:** Se confirmó la existencia de patrones lingüísticos distintivos para "predators" y "víctimas", validando la hipótesis central.

Lecciones Aprendidas

- **Metodología vs. Métricas Superficiales:** Una precisión del 100% inicial no indicaba éxito, sino un error metodológico.
- **Validación Robusta:** La implementación de **GroupShuffleSplit** fue crucial para evitar la fuga de datos y obtener estimaciones realistas.
- **Contexto del Problema:** El **recall** para la clase "Predator" se identificó como la métrica más crítica en este dominio de seguridad.

ChatSentinel representa un avance fiable en la detección de "grooming". Aunque no es infalible., subrayando que la protección de menores en el entorno digital es una responsabilidad colectiva que la ciencia de datos puede potenciar.