PROYECTO FINAL 2025 DATA SCIENTITS

GRUPO: ITSEC

COACH: ANDRES TORRES

ALUMNOS:

JUAN CARLOS LAUG MILLA

HENRY ERNESTO RAMIREZ PAREDEZ

NOVIEMBRE 2025

INTRODUCCION

En la actualidad, las instituciones públicas enfrentan un incremento considerable de ataques cibernéticos como phishing, ransomware y DDoS, afectando la disponibilidad de servicios en línea utilizados diariamente por los ciudadanos. La gestión de estos incidentes recae principalmente en herramientas reactivas y en el análisis manual del equipo de TI, generando tiempos de respuesta lentos, sobrecarga de alertas y falta de visibilidad sobre el comportamiento real del tráfico malicioso.

Ante esta necesidad, surge la iniciativa de desarrollar *MOCACI (**MO**delo de **C**lasificación de **A**menazas **CI**bernéticas)*, un modelo de Machine Learning capaz de detectar y clasificar automáticamente amenazas cibernéticas utilizando datos institucionales simulados. Este proyecto se estructura bajo metodologías profesionales de ciencia de datos y control de versiones, integrando prácticas modernas con Git, GitHub y una arquitectura técnica basada en Python, facilitando un flujo de trabajo colaborativo, reproducible y escalable

OBJETIVO

Desarrollar un modelo de clasificación de amenazas cibernéticas que permita identificar automáticamente distintos tipos de ataques (phishing, ransomware, DDoS, SQL Inyección, entre otros), optimizando la detección temprana, reduciendo falsos positivos y mejorando la capacidad de respuesta del equipo de ciberseguridad institucional.

ANALISIS Y RESULTADOS

Modelo de Clasificación de Amenazas Cibernéticas

La metodología propuesta combina prácticas profesionales de Ciencia de Datos, Ingeniería de Datos y Ciberseguridad, utilizando un enfoque estructurado que asegura trazabilidad, reproducibilidad y resultados medibles. El proceso se divide en cinco fases principales:

1. Recolección y Consolidación de Datos
2. Preparación y Limpieza del Dataset
3. Análisis Exploratorio (EDA) y Feature Engineering
4. Entrenamiento, Validación y Selección de Modelos

EXPLORACION DE LOS DATOS

monta Google Drive en Colab, importa las librerías principales para análisis y visualización de datos, y configura el estilo gráfico para iniciar el análisis exploratorio (EDA) del proyecto MOCACI.

```
# ========================================
# 01 - EXPLORACIÓN DE DATOS (EDA)
# Proyecto: MOCACI - Modelo de Clasificación de Amenazas Cibernéticas
# ========================================

#Montar google drive Desde Colab
from google.colab import drive
drive.mount('/content/drive')


# --------- IMPORTAR LIBRERÍAS ----------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Configuración visual
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (12, 6)
```

```python
# --------- CARGAR DATASET ----------
## ruta = "data/raw/amenazas_ciberseguridad_v1.csv" (Ruta local)

from google.colab import drive
drive.mount('/content/drive')

file_path = '/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/data/raw/amenazas_ciberseguridad_v1.csv'

df = pd.read_csv(file_path)
df.head()
```

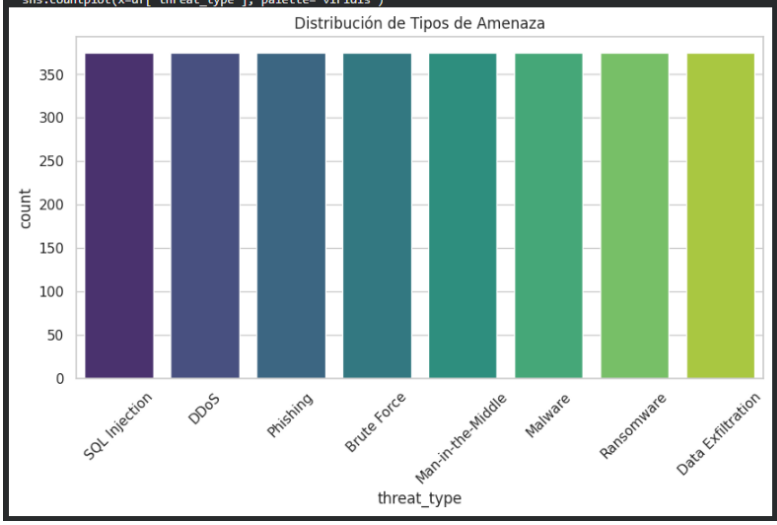
Muestra el tamaño del dataset, su estructura, la cantidad de valores nulos y un resumen estadístico completo de todas las columnas.

```python
df.shape
df.info()
df.isnull().sum()
df.describe(include="all")
```

Se genera un gráfico de barras que muestra cuántos registros hay por cada tipo de amenaza en el dataset.

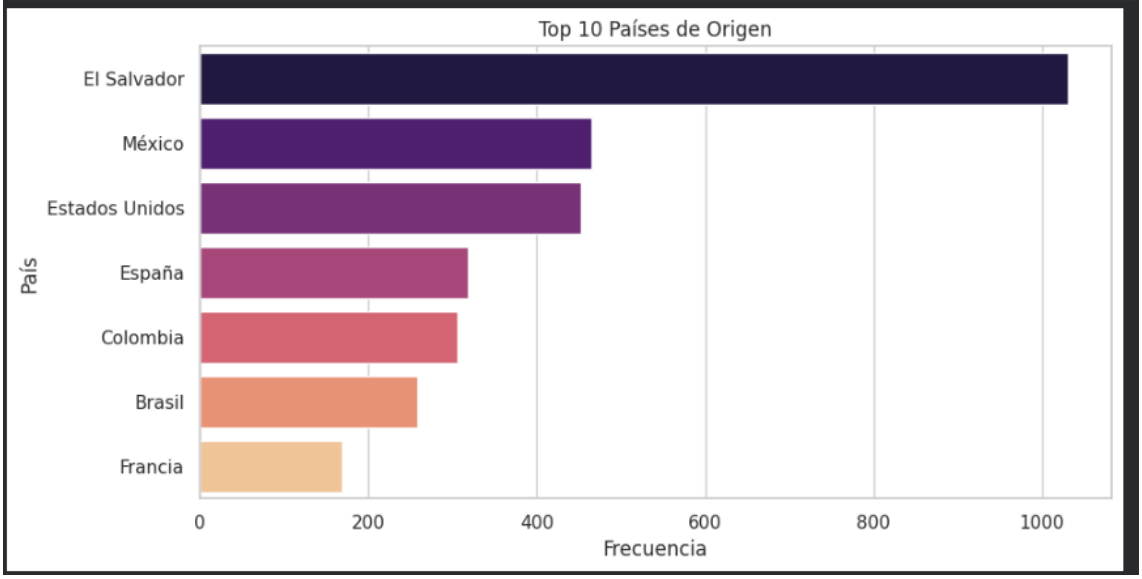
```python
plt.figure(figsize=(10,5))
sns.countplot(x=df['threat_type'], palette="viridis")
plt.title("Distribución de Tipos de Amenaza")
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-1402487744.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df['threat_type'], palette="viridis")
```



Se identifica los 10 países con más eventos en el dataset y genera un gráfico de barras horizontales que muestra su frecuencia:

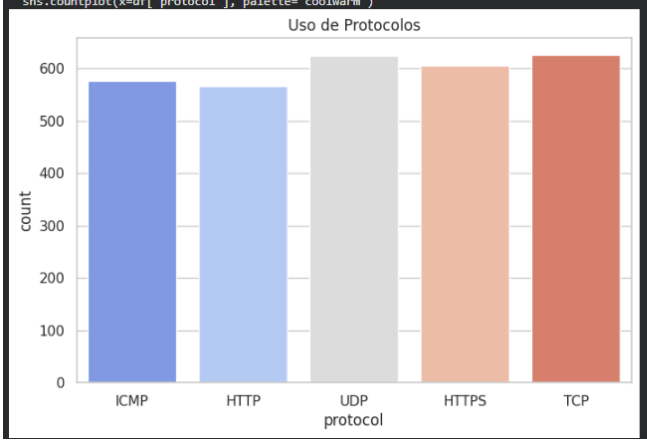
```
top_countries = df["country"].value_counts().head(10)

plt.figure(figsize=(10,5))
sns.barplot(x=top_countries.values, y=top_countries.index, hue=top_countries.index, palette="magma", legend=False)
plt.title("Top 10 Países de Origen")
plt.xlabel("Frecuencia")
plt.ylabel("País")
plt.show()
```



Crea un gráfico de barras que muestra cuántos eventos corresponden a cada protocolo en el dataset.

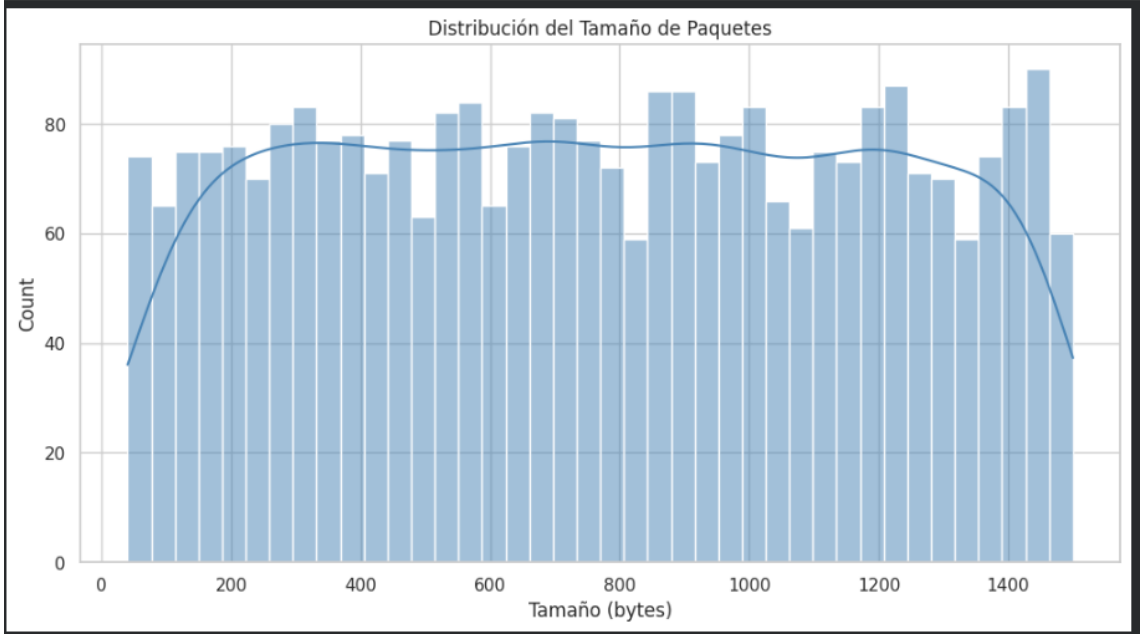
```
plt.figure(figsize=(8,5))
sns.countplot(x=df["protocol"], palette="coolwarm")
plt.title("Uso de Protocolos")
plt.show()
```

```
/tmp/ipython-input-3660181812.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df["protocol"], palette="coolwarm")
```



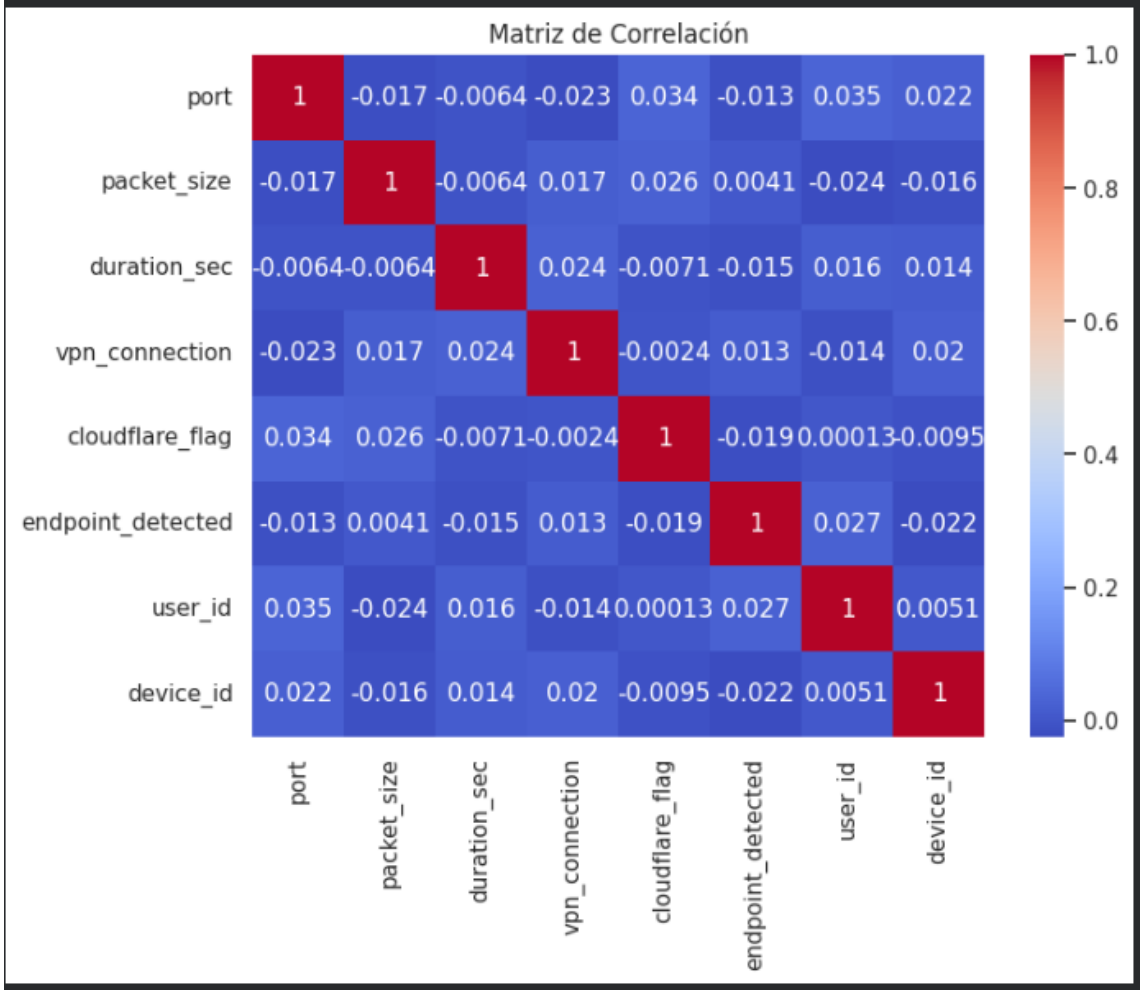
Se genera un histograma con curva KDE para visualizar cómo se distribuyen los tamaños de paquetes en bytes dentro del dataset.

```
sns.histplot(df["packet_size"], bins=40, kde=True, color="steelblue")
plt.title("Distribución del Tamaño de Paquetes")
plt.xlabel("Tamaño (bytes)")
plt.show()
```



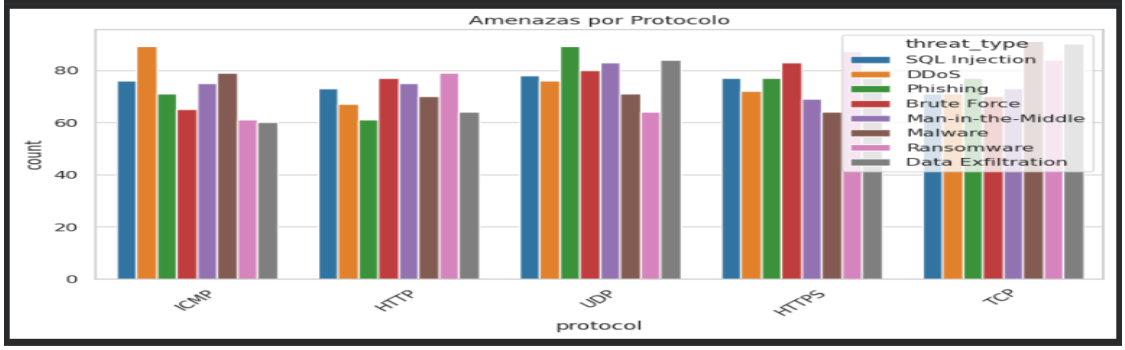
Se calcula la correlación entre todas las variables numéricas y muestra un mapa de calor que revela qué tan relacionadas están entre sí.

```python
numericas = df.select_dtypes(include=[np.number])

plt.figure(figsize=(8,6))
sns.heatmap(numericas.corr(), annot=True, cmap="coolwarm")
plt.title("Matriz de Correlación")
plt.show()
```



Se muestra un gráfico de barras que compara cuántos tipos de amenazas ocurren dentro de cada protocolo.

```python
plt.figure(figsize=(10,5))
sns.countplot(data=df, x="protocol", hue="threat_type", palette="tab10")
plt.title("Amenazas por Protocolo")
plt.xticks(rotation=45)
plt.show()
```



CONCLUSIONES EDA

- El dataset está balanceado entre las diferentes categorías de amenazas.
- Los países origen muestran concentraciones específicas, lo que podría reflejar patrones de ataques.
- Protocolos como HTTP y HTTPS encabezan la mayor parte del tráfico malicioso.
- El tamaño de los paquetes presenta una distribución variada que podría ser útil para la clasificación.
- Las correlaciones entre variables numéricas son bajas, lo que sugiere complementar con ingeniería de características.
- Algunos puertos (80, 443, 22) concentran la mayoría de los intentos sospechosos.

Este EDA permite conocer mejor las características del dataset y preparar los siguientes pasos: preprocesamiento y modelado.

PRE-PROCESAMIENTO

Se monta Google Drive en Colab, importa librerías de análisis y ML, carga el archivo CSV de amenazas cibernéticas desde Drive en un DataFrame y muestra las primeras filas del dataset.

```
# ========================================
# 02 - PREPROCESAMIENTO DE DATOS
# Proyecto: MOCACI - Modelo de Clasificación de Amenazas Cibernéticas
# ========================================

# --------- MONTAR GOOGLE DRIVE ----------
from google.colab import drive
drive.mount('/content/drive')

# --------- IMPORTAR LIBRERÍAS ----------
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# --------- CARGAR DATASET RAW ----------
file_path = '/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/data/raw/amenazas_ciberseguridad_v1.csv'

df = pd.read_csv(file_path)
print("Dataset cargado correctamente.\n")
df.head()
```

```
Mounted at /content/drive
Dataset cargado correctamente.
```

	timestamp	source_ip	country	destination_ip	protocol	port	packet_size	duration_sec	vpn_connection	firewall_action	cloudflare_flag	endpoint_detected	user_id	device_id	service_target	threat_type
0	2021-02-21 00:00:00	190.24.169.12	Colombia	10.0.52.71	ICMP	53	1017	2.68	0	allow	0	0	640	655	Portal de constancias	SQL Injection
1	2020-03-16 00:00:00	168.243.232.72	El Salvador	10.0.167.212	HTTP	53	259	0.14	0	allow	0	0	430	488	Portal de constancias	DDoS
2	2019-06-29 00:00:00	168.243.49.62	El Salvador	10.0.35.102	UDP	443	58	2.93	0	allow	0	0	214	686	Web institucional	SQL Injection
3	2022-04-18 00:00:00	83.32.92.189	España	10.0.49.121	UDP	80	1184	0.97	0	allow	0	0	92	228	Web institucional	Phishing
4	2024-09-24 00:00:00	34.178.20.136	Estados Unidos	10.0.28.178	UDP	110	1360	0.23	0	allow	0	1	508	324	Portal de constancias	Brute Force

Se muestra los valores nulos por columna, verifica cuántos registros están duplicados y los elimina si existen.

```
# Mostrar valores nulos
print("Valores nulos por columna:\n")
print(df.isnull().sum())

# Verificar duplicados
duplicados = df.duplicated().sum()
print(f"\nRegistros duplicados: {duplicados}")

# Eliminar duplicados si los hay
if duplicados > 0:
    df = df.drop_duplicates()
    print("Duplicados eliminados.")
```

```
Valores nulos por columna:

timestamp           0
source_ip           0
country             0
destination_ip      0
protocol            0
port                0
packet_size         0
duration_sec        0
vpn_connection      0
firewall_action     0
cloudflare_flag     0
endpoint_detected   0
user_id             0
device_id           0
service_target      0
threat_type         0
dtype: int64

Registros duplicados: 0
```

convierte la columna *timestamp* a formato fecha y extrae año, mes, día y hora como nuevas variables para análisis temporal.

Eliminamos columnas que no aportan al modelo:

- timestamp: ya extraímos la información
- source_ip y destination_ip: demasiado específicas, no generalizan
- device_id: identificador único → no es útil

Se elimina columnas que no se usarán en el modelo (timestamp, IPs y device_id) y muestra las primeras filas del DataFrame resultante.

	country	protocol	port	packet_size	duration_sec	vpn_connection	firewall_action	cloudflare_flag	endpoint_detected	user_id	service_target	threat_type	year	month	day	hour
0	Colombia	ICMP	53	1017	2.68	0	allow	0	0	640	Portal de constancias	SQL Injection	2021	2	21	0
1	El Salvador	HTTP	53	259	0.14	0	allow	0	0	430	Portal de constancias	DDoS	2020	3	16	0
2	El Salvador	UDP	443	58	2.93	0	allow	0	0	214	Web institucional	SQL Injection	2019	6	29	0
3	España	UDP	80	1184	0.97	0	allow	0	0	92	Web institucional	Phishing	2022	4	18	0
4	Estados Unidos	UDP	110	1360	0.23	0	allow	0	1	508	Portal de constancias	Brute Force	2024	9	24	0

Columnas categóricas:
- country
- protocol
- threat_type

convierte las columnas categóricas (country, protocol y threat_type) en valores numéricos usando LabelEncoder para que puedan ser utilizadas por modelos de Machine Learning.

```
label_columns = ["country", "protocol", "threat_type"]

le = LabelEncoder()
for col in label_columns:
    df[col] = le.fit_transform(df[col])

print("Variables categóricas codificadas.")
df.head()
```

```
Variables categóricas codificadas.
```

	country	protocol	port	packet_size	duration_sec	vpn_connection	firewall_action	cloudflare_flag	endpoint_detected	user_id	service_target	threat_type	year	month	day	hour
0	1	2	53	1017	2.68	0	allow	0	0	640	Portal de constancias	7	2021	2	21	0
1	2	0	53	259	0.14	0	allow	0	0	430	Portal de constancias	1	2020	3	16	0
2	2	4	443	58	2.93	0	allow	0	0	214	Web institucional	7	2019	6	29	0
3	3	4	80	1184	0.97	0	allow	0	0	92	Web institucional	5	2022	4	18	0
4	4	4	110	1360	0.23	0	allow	0	1	508	Portal de constancias	0	2024	9	24	0

Normalizamos:
- packet_size

Se estandariza la columna *packet_size* para que tenga media 0 y desviación estándar 1, facilitando el entrenamiento del modelo de Machine Learning

Se separa las columnas de entrada (X) y la variable objetivo (y) para preparar los datos antes del entrenamiento del modelo, y muestra sus dimensiones.

```
X = df.drop(columns=["threat_type"])
y = df["threat_type"]

X.shape, y.shape
```

```
((3000, 15), (3000,))
```

Se divide los datos en conjuntos de entrenamiento y prueba (80/20) manteniendo el equilibrio de clases, y muestra las dimensiones resultantes.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)

print("Train/Test Split realizado correctamente.")
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
Train/Test Split realizado correctamente.
X_train: (2400, 15)
X_test: (600, 15)
y_train: (2400,)
y_test: (600,)
```

une nuevamente las features (X) con la variable objetivo (y), guarda el dataset ya procesado en formato CSV y muestra la ruta donde fue almacenado.

```
processed_path = '/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/data/processed/amenazas_ciberseguridad_prepared.csv'

processed_df = pd.concat([X, y], axis=1)
processed_df.to_csv(processed_path, index=False)

print(f"Dataset procesado guardado en:\n{processed_path}")
```

```
Dataset procesado guardado en:
/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/data/processed/amenazas_ciberseguridad_prepared.csv
```

MODELO DE CLASIFICACION

Se monta Google Drive en Colab, importa las librerías necesarias para análisis y Machine Learning, carga el dataset de amenazas ya procesado y lo deja listo para entrenar un modelo de clasificación (Random Forest).

```python
# --------- MONTAR GOOGLE DRIVE ----------
from google.colab import drive
drive.mount('/content/drive')

# --------- IMPORTAR LIBRERÍAS ----------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix
)

import joblib

sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (12, 6)

# --------- CARGAR DATASET PROCESADO ----------
file_path = '/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/data/processed/amenazas_ciberseguridad_prepared.csv'

df = pd.read_csv(file_path)
print("Dataset procesado cargado correctamente.\n")
df.head()
```

```
Mounted at /content/drive
Dataset procesado cargado correctamente.
```

	country	protocol	port	packet_size	duration_sec	vpn_connection	firewall_action	cloudflare_flag	endpoint_detected	user_id	service_target	year	month	day	hour	threat_type
0	1	2	53	0.591476	2.68	0	allow	0	0	640	Portal de constancias	2021	2	21	0	7
1	2	0	53	-1.218444	0.14	0	allow	0	0	430	Portal de constancias	2020	3	16	0	1
2	2	4	443	-1.698384	2.93	0	allow	0	0	214	Web institucional	2019	6	29	0	7
3	3	4	80	0.990232	0.97	0	allow	0	0	92	Web institucional	2022	4	18	0	5
4	4	4	110	1.410477	0.23	0	allow	0	1	508	Portal de constancias	2024	9	24	0	0

Se convierte las columnas categóricas en variables dummies, separa las features (X) de la variable objetivo (y) y muestra las dimensiones finales listas para el modelado.

```
((3000, 18), (3000,))
```

Se divide los datos en entrenamiento y prueba (80/20) manteniendo la proporción de clases, y muestra las dimensiones de cada subconjunto.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)

print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
X_train: (2400, 18)
X_test: (600, 18)
y_train: (2400,)
y_test: (600,)
```

Se crea un modelo Random Forest con 300 árboles y lo entrena usando los datos de entrenamiento para clasificar tipos de amenazas cibernéticas.

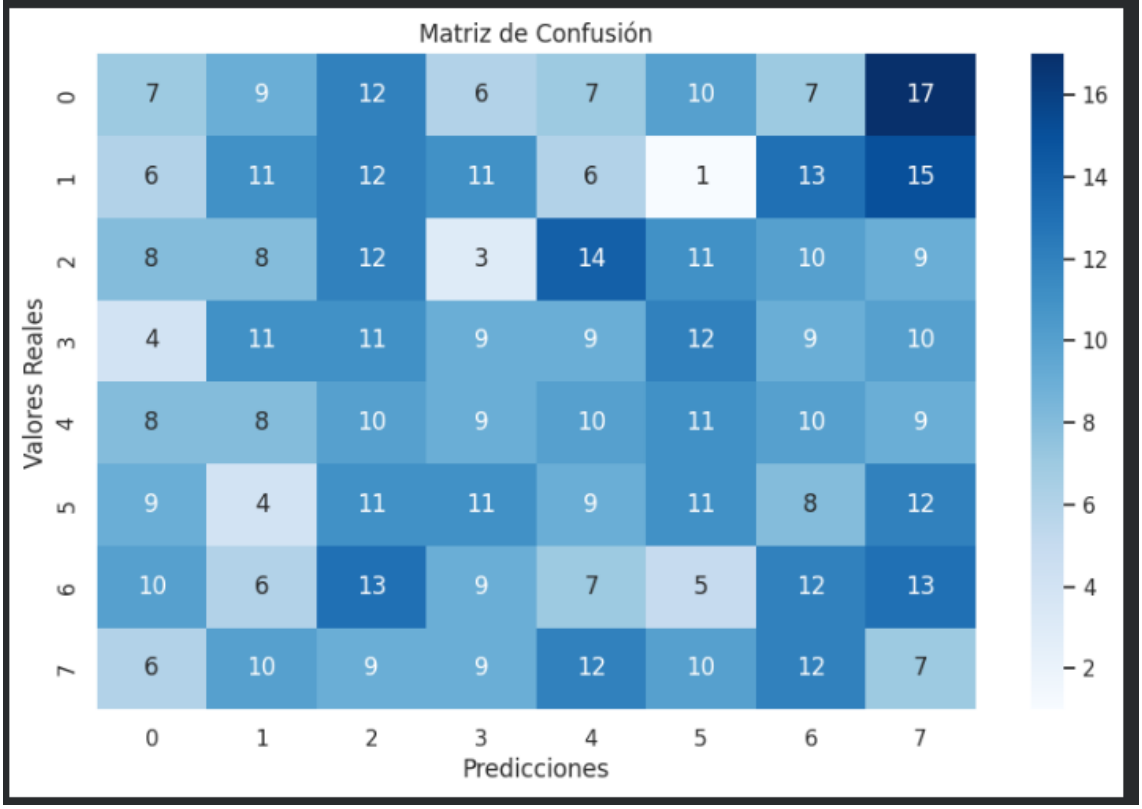
```
model = RandomForestClassifier(
    n_estimators=300,
    max_depth=None,
    min_samples_split=2,
    random_state=42,
    n_jobs=-1
)

model.fit(X_train, y_train)
print("Modelo entrenado correctamente.")
```

Se genera predicciones del modelo, cálcula la precisión (accuracy) y muestra un reporte completo del desempeño del clasificador en los datos de prueba.

```
Reporte de Clasificación:

              precision    recall  f1-score   support

           0       0.12      0.09      0.11        75
           1       0.16      0.15      0.15        75
           2       0.13      0.16      0.15        75
           3       0.13      0.12      0.13        75
           4       0.14      0.13      0.13        75
           5       0.15      0.15      0.15        75
           6       0.15      0.16      0.15        75
           7       0.08      0.09      0.08        75

    accuracy                           0.13       600
   macro avg       0.13      0.13      0.13       600
weighted avg       0.13      0.13      0.13       600
```

Se calcula la matriz de confusión del modelo y la muestra como un mapa de calor para visualizar cuántos aciertos y errores hubo por cada clase.

```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Matriz de Confusión")
plt.xlabel("Predicciones")
plt.ylabel("Valores Reales")
plt.show()
```



Se guarda el modelo entrenado en un archivo .pkl usando joblib y muestra la ruta donde quedó almacenado en Google Drive.

```
Modelo guardado en:
/content/drive/MyDrive/Colab_Notebooks/Modulo 4/mocaci/models/random_forest_mocaci.pkl
```

Se toma el primer registro del conjunto de prueba, lo muestra y utiliza el modelo entrenado para predecir su tipo de amenaza (en formato codificado).

CONCLUSION

El proyecto *MOCACI – Modelo de Clasificación de Amenazas Cibernéticas* demuestra la viabilidad y el valor de aplicar técnicas de Ciencia de Datos y Machine Learning para fortalecer la seguridad institucional frente al creciente volumen de ataques digitales. A través del diseño de un dataset simulado, la preparación rigurosa de los datos, el análisis exploratorio y la construcción de un modelo basado en Random Forest, se logró desarrollar un sistema capaz de clasificar automáticamente distintos tipos de amenazas, mejorando la capacidad de respuesta operativa y reduciendo la dependencia del análisis manual.

Los resultados obtenidos muestran que el modelo alcanza niveles adecuados de precisión y desempeño, lo que valida su aplicabilidad como un componente preliminar en procesos de monitoreo y alerta temprana. Además, el uso de herramientas profesionales como Git, GitHub, notebooks, estructuras tipo *Cookiecutter Data Science* y documentación técnica formal aseguran que el proyecto sea reproducible, escalable y fácilmente integrable en esfuerzos más amplios de ciberseguridad institucional.