UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

Parcial II – Parte Práctica (40%)

Materia: Inteligencia Artificial aplicada a la Ciberseguridad **Profesor**: Dr. Carlos A. Rovetto
Fecha: miércoles, 11 de junio de 2025 Grupo: _1S3134__

Estudiante: Ernesto Morán Cédula: 4-828-1810 Fila: ___

Estudiante: Aurelio Parra Cédula: 8-1024-554 Fila: ___

Objetivo
Aplicar los conocimientos recibidos sobre la creación de datasets utilizando Python y el análisis de estos.

Descripción: Crear un dataset desde procesos de su computadora y subirlo a un sitio público como los siguientes (GitHub, Google Drive, Kaggle, AWS S3, Dropbox, etc). Posteriormente, elabore el código Python para acceder a él a través de python y genere la siguiente información para el dataset generado.

- Estadísticas básicas: Media, mediana, moda y desviación estándar.
- Correlación entre variables: Matriz de correlación para ver las relaciones entre variables numéricas.

Ejemplos de tipos de datasets que puede crear

1. Uso de la CPU: Detectar anomalías en el comportamiento de la CPU para identificar posibles ataques de denegación de servicio (DDoS).
2. Uso de la Memoria RAM: Monitorear el uso excesivo de memoria para detectar malware o procesos maliciosos.
3. Uso del Disco Duro: Identificar cambios sospechosos en el uso del disco que podrían indicar filtración de datos o actividad maliciosa.
4. Uso de la Red: Detectar patrones de tráfico anómalos para identificar intrusiones o ataques de red.
5. Lista de Procesos Activos: Clasificar procesos maliciosos y legítimos para prevenir accesos no autorizados o malware.
6. Temperatura de Componentes del Sistema: Detectar sobrecalentamientos anómalos que puedan indicar un ataque físico o manipulación del hardware.
7. Tiempo de Actividad del Sistema: Analizar tiempos de actividad inusuales para detectar sistemas comprometidos o accesos no autorizados.
8. Eventos del Sistema (Logs): Identificar patrones de eventos que puedan señalar intentos de intrusión o fallos de seguridad.
9. Consumo de Energía: Detectar patrones inusuales de consumo energético relacionados con ataques físicos o manipulaciones de hardware.
10. Actividades de Entrada del Usuario (Teclado): Analizar patrones de comportamiento del usuario para detectar accesos no autorizados o actividades maliciosas.

Observaciones: Se le dará un enlace de un formulario para que suba el código Python y el enlace del dataset público. Debe asegurarse que a través de la ejecución del código se puede tener acceso al dataset.

DatasetRedes.py !M

DatasetRedes.py > ...

```python
# Dependencias Necesarias:
# pip install pandas seaborn matplotlib requests psutil

# Se Requiere pip, Y Ejecutar El Comando Arriba Para La Ejecucion De Este Codigo.

# Ernesto Morán (4-828-1810)
# Aurelio Parra (8-1024-554)

import subprocess
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import requests
import psutil
import re
import time
import os
import tkinter as tk
from tkinter import messagebox

# ========================
def get_netstat_output():
    result = subprocess.run(["netstat", "-ano"], capture_output=True, text=True)
    return result.stdout.splitlines()

def parse_netstat(lines):
    connections = []
    for line in lines:
        if line.startswith("  TCP") or line.startswith("  UDP"):
            parts = re.split(r"\s+", line.strip())
            if len(parts) >= 5:
                protocolo = parts[0]
                local = parts[1]
                remote = parts[2]
                estado = parts[3] if protocolo == "TCP" else "ESTABLISHED"
                pid = parts[4] if protocolo == "TCP" else parts[3]
```

```
                pid = parts[4] if protocolo == "TCP" else parts[3]

                if ":" in remote:
                    ip, port = remote.rsplit(":", 1)
                    if ip not in ["0.0.0.0", "[::]", "::", ""]:
                        try:
                            process_name = psutil.Process(int(pid)).name()
                        except:
                            process_name = "Desconocido"
                        try:
                            port = int(port)
                            pid = int(pid)
                            connections.append({
                                "Protocolo": protocolo,
                                "IP Remota": ip,
                                "Puerto Remoto": port,
                                "Estado": estado,
                                "PID": pid,
                                "Proceso": process_name
                            })
                        except:
                            continue
    return pd.DataFrame(connections)

# =======================
def geolocalizar_ips(df):
    unique_ips = df['IP Remota'].unique()
    geo_data = {}

    for ip in unique_ips:
        if ip.startswith("127.") or ip in ["::1"]:
            geo_data[ip] = {'País': 'LOCAL', 'Región': 'LOCAL', 'Ciudad': 'LOCAL', 'Org': 'LOCAL', 'Latitud': None, 'Longitud': None}
            continue

        try:
```

```
            response = requests.get(f"http://ip-api.com/json/{ip}").json()
            if response['status'] == 'success':
                geo_data[ip] = {
                    'País': response.get('country', 'N/A'),
                    'Región': response.get('regionName', 'N/A'),
                    'Ciudad': response.get('city', 'N/A'),
                    'Org': response.get('org', 'N/A'),
                    'Latitud': response.get('lat', None),
                    'Longitud': response.get('lon', None)
                }
            else:
                geo_data[ip] = {'País': 'N/A', 'Región': 'N/A', 'Ciudad': 'N/A', 'Org': 'N/A', 'Latitud': None, 'Longitud': None}
        except:
            geo_data[ip] = {'País': 'N/A', 'Región': 'N/A', 'Ciudad': 'N/A', 'Org': 'N/A', 'Latitud': None, 'Longitud': None}
        time.sleep(0.5)

    geo_df = pd.DataFrame.from_dict(geo_data, orient='index')
    geo_df.index.name = 'IP Remota'
    return df.join(geo_df, on='IP Remota')

# =======================
def mostrar_estadisticas(df):
    media = df['Puerto Remoto'].mean()
    mediana = df['Puerto Remoto'].median()
    moda = df['Puerto Remoto'].mode()[0]
    desviacion = df['Puerto Remoto'].std()

    texto = f"""
    📊 Estadísticas del Puerto Remoto

    Media: {media:.2f}
    Mediana: {mediana:.2f}
    Moda: {moda}
    Desviación estándar: {desviacion:.2f}
    """
```

(variable) geo_data: dict

```
    Desviación estándar: {desviacion:.2f}
    """

    root = tk.Tk()
    root.withdraw()
    messagebox.showinfo("🐾 Estadísticas Básicas", texto)
    root.destroy()

def mostrar_correlacion(df):
    df['ProtocoloNum'] = df['Protocolo'].map({'TCP': 1, 'UDP': 2})
    df['EstadoNum'] = df['Estado'].astype('category').cat.codes
    df['ProcesoID'] = df['Proceso'].astype('category').cat.codes

    correlacion = df[['Puerto Remoto', 'PID', 'ProtocoloNum', 'EstadoNum', 'ProcesoID']].corr()
    print("\n📌 Matriz de correlación:")
    print(correlacion)

    sns.heatmap(correlacion, annot=True, cmap="coolwarm")
    plt.title("💻 Matriz de correlación")
    plt.show()

def abrir_csv(path):
    print(f"📂 Abriendo el archivo: {path}")
    os.startfile(path)

# ========================
def main():
    todas_las_conexiones = pd.DataFrame()

    print("El monitoreo ha comenzado, presiona Ctrl + C para detenerlo...")
    try:
        while True:
            lines = get_netstat_output()
            df = parse_netstat(lines)
            if not df.empty:
                df = geolocalizar_ips(df)
```

```
# ========================
def main():
    todas_las_conexiones = pd.DataFrame()

    print("El monitoreo ha comenzado, presiona Ctrl + C para detenerlo...")
    try:
        while True:
            lines = get_netstat_output()
            df = parse_netstat(lines)
            if not df.empty:
                df = geolocalizar_ips(df)
                todas_las_conexiones = pd.concat([todas_las_conexiones, df], ignore_index=True)
            time.sleep(5)
    except KeyboardInterrupt:
        print("\n🚨 Ctrl + C detectado, terminando captura...")

    if todas_las_conexiones.empty:
        print("No se recolectó ninguna conexión activa.")
        return

    file_path = "conexiones_monitorizadas.csv"
    todas_las_conexiones.to_csv(file_path, index=False)
    print(f"✅ Dataset final guardado como '{file_path}'")

    # Mostrar primero estadísticas y correlaciones, luego abrir el archivo
    mostrar_estadisticas(todas_las_conexiones)
    mostrar_correlacion(todas_las_conexiones)
    abrir_csv(file_path)

# ========================
main()
```

Resultados


PROBLEMS
OUTPUT
DEBUG CONSOLE
PORTS
TERMINAL
PS C:\Users\aurel\Desktop\P4M1\Code\parcial-ia\parcial-ia> & C:/U
parcial-ia/DatasetRedes.py
El monitoreo ha comenzado, presiona Ctrl + C para detenerlo...
Ctrl + C detectado, terminando captura...
Dataset final guardado como 'conexiones_monitorizadas.csv'
Estadísticas Básicas
Estadísticas del Puerto Remoto
Media: 8732.16
Mediana: 443.00
Moda: 443
Desviación estándar: 17607.22
Aceptar


Figure 1
Matriz de correlación
Puerto Remoto
PID
ProtocoloNum
EstadoNum
ProcesoID
1
0.044
-0.28
0.57
0.044
1
-0.81
0.26
-0.28
-0.81
1
-0.48
0.57
0.26
-0.48
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
abrir el archivo
Python
Matriz de correlación:
Puerto Remoto PID ProtocoloNum EstadoNum ProcesoID
Puerto Remoto 1.000000 0.043806 NaN -0.282164 0.572507
PID 0.043806 1.000000 NaN -0.806364 0.258392
ProtocoloNum NaN NaN NaN NaN NaN
EstadoNum -0.282164 -0.806364 NaN 1.000000 -0.481623
ProcesoID 0.572507 0.258392 NaN -0.481623 1.000000
C:\Users\aurel\AppData\Local\Programs\Python\Python312\Lib\tkinter__init__.py:862: UserWarning: Glyph 128187 (\N{PERSONAL COMPUTER}) missing from font(s) DejaVu Sans.
func(*args)
C:\Users\aurel\AppData\Local\Programs\Python\Python312\Lib\tkinter__init__.py:862: UserWarning: Glyph 128187 (\N{PERSONAL COMPUTER}) missing from font(s) DejaVu Sans.
func(*args)
Abriendo el archivo: conexiones_monitorizadas.csv