Practica machine learning

Importo el modulo **numpy**, y lo asigno al alias **np numpy**, es una biblioteca para la computación científica en Python, importo el modulo pandas y lo asigno al alias **pd**, pandas proporciona estructura de datos, y herramientas de análisis de datos, importo el modulo **sklearn**, que es una biblioteca de aprendizaje automático que incluye varias herramientas, por ultimo importo el submodulo **pyplot** de **matplotlib** y lo asigna al alias **plt**, es una biblioteca para la creación de gráficos y visualizaciones en Python.

```
1. import numpy as np
   import pandas as pd
   import sklearn
   import matplotlib.pyplot as plt
```

Cambiamos las opciones de visualización de los DataFrames a un formato más amigable.

2. pd.options.display.max_columns = None: Muestra todas las columnas del DataFrame sin truncarlas.

   pd.options.display.float_format = "{:,.2f}".format: Muestra los números decimales con dos cifras después del punto y usa una coma como separador de miles.

Cargo los datos desde un archivo CSV llamado **cybersecurity_attacks.csv** en un DataFrame llamado **df,** uso el método head() para mostrar las primeras 5 filas del dataframe.

```
3. df = pd.read_csv("cybersecurity_attacks.csv")
   df.head()
```

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	Malware Indicators	Anomaly Scores	Alerts/Warnings	Attack Type
0	2023-05-30 06:33:58	103.216.15.12	84.9.164.252	31225	17616	ICMP	503	Data	HTTP	Qui natus odio asperiores nam. Optio nobis ius...	IoC Detected	28.67	NaN	Malware
1	2020-08-26 07:08:30	78.199.217.198	66.191.137.154	17245	48166	ICMP	1174	Data	HTTP	Aperiam quos modi officiis veritatis rem. Omni...	IoC Detected	51.50	NaN	Malware
2	2022-11-13 08:23:25	63.79.210.48	198.219.82.17	16811	53600	UDP	306	Control	HTTP	Perferendis sapiente vitae soluta. Hic delectu...	IoC Detected	87.42	Alert Triggered	DDoS
3	2023-07-02 10:38:46	163.42.196.10	101.228.192.255	20018	32534	UDP	385	Data	HTTP	Totam maxime beatae expedita explicabo porro l...	NaN	15.79	Alert Triggered	Malware
4	2023-07-16 13:11:07	71.166.185.76	189.243.174.238	6131	26646	TCP	1462	Data	DNS	Odit nesciunt dolorem nisi iste iusto.	NaN	0.52	Alert Triggered	DDoS

4. df.describe(include="all"): Genera estadísticas descriptivas para todas las columnas del DataFrame.

   fillna("-"): Rellena los valores nulos (NaN) en el resumen estadístico con un guion ("-").

10]:

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	Malware Indicators	Anomaly Scores	Alerts/Warnings	Attack Type
count	40000	40000	40000	40,000.00	40,000.00	40000	40,000.00	40000	40000	40000	20000	40,000.00	19933	40000
unique	39997	40000	40000	-	-	3	-	2	3	40000	1	-	1	3
top	2022-04-17 20:05:34	138.156.5.40	91.54.135.213	-	-	ICMP	-	Control	DNS	Fugiat tenetur natus perferendis. Mollitia bla...	IoC Detected	-	Alert Triggered	DDoS
freq	2	1	1	-	-	13429	-	20237	13376	1	20000	-	19933	13428
mean	-	-	-	32,970.36	33,150.87	-	781.45	-	-	-	-	50.11	-	-
std	-	-	-	18,560.43	18,574.67	-	416.04	-	-	-	-	28.85	-	-
min	-	-	-	1,027.00	1,024.00	-	64.00	-	-	-	-	0.00	-	-
25%	-	-	-	16,850.75	17,094.75	-	420.00	-	-	-	-	25.15	-	-
50%	-	-	-	32,856.00	33,004.50	-	782.00	-	-	-	-	50.34	-	-
75%	-	-	-	48,928.25	49,287.00	-	1,143.00	-	-	-	-	75.03	-	-
max	-	-	-	65,530.00	65,535.00	-	1,500.00	-	-	-	-	100.00	-	-

Calculo la proporción de valores nulos (NaN) en cada columna del df.

Hay unas series de columnasque tienen un 50% de valores nulos, las demás no tienen valores nulos, entonces esto sugiere que faltan datos, lo que podría afectar al análisis.

5. df.isna().mean()

```
Timestamp                0.00
Source IP Address        0.00
Destination IP Address   0.00
Source Port              0.00
Destination Port         0.00
Protocol                 0.00
Packet Length            0.00
Packet Type              0.00
Traffic Type             0.00
Payload Data             0.00
Malware Indicators       0.50
Anomaly Scores           0.00
Alerts/Warnings          0.50
Attack Type              0.00
Attack Signature         0.00
Action Taken             0.00
Severity Level           0.00
User Information         0.00
Device Information       0.00
Network Segment          0.00
Geo-location Data        0.00
Proxy Information        0.50
Firewall Logs            0.50
IDS/IPS Alerts           0.50
Log Source               0.00
dtype: float64
```

Convierto los tipos de datos de las columnas del df a los tipos adecuados, la mamyoria de columnas de texto se convirtieron a string[python] las columnas de numemros enteros a Int64, las que contienen números flotantes asido convertida a Float64.

6. df.convert_dtypes().dtypes

```
Timestamp                 string[python]
Source IP Address         string[python]
Destination IP Address    string[python]
Source Port                        Int64
Destination Port                   Int64
Protocol                  string[python]
Packet Length                      Int64
Packet Type               string[python]
Traffic Type              string[python]
Payload Data              string[python]
Malware Indicators        string[python]
Anomaly Scores                   Float64
Alerts/Warnings           string[python]
Attack Type               string[python]
Attack Signature          string[python]
Action Taken              string[python]
Severity Level            string[python]
User Information          string[python]
Device Information        string[python]
Network Segment           string[python]
Geo-location Data         string[python]
Proxy Information         string[python]
Firewall Logs             string[python]
IDS/IPS Alerts            string[python]
Log Source                string[python]
dtype: object
```

Selección de valores NULOS (NaN) y muestro todas las filas para estas columnas mdel df.

7. cols_with_nan = ["Malware Indicators", "Alerts/Warnings", "Proxy Information", "Firewall Logs", "IDS/IPS Alerts"]
   df[cols_with_nan]

	Malware Indicators	Alerts/Warnings	Proxy Information	Firewall Logs	IDS/IPS Alerts
0	IoC Detected	NaN	150.9.97.135	Log Data	NaN
1	IoC Detected	NaN	NaN	Log Data	NaN
2	IoC Detected	Alert Triggered	114.133.48.179	Log Data	Alert Data
3	NaN	Alert Triggered	NaN	NaN	Alert Data
4	NaN	Alert Triggered	149.6.110.119	NaN	Alert Data
...	...	...	...	...	...
39995	IoC Detected	NaN	NaN	Log Data	Alert Data
39996	IoC Detected	NaN	60.51.30.46	Log Data	NaN
39997	IoC Detected	NaN	NaN	Log Data	Alert Data
39998	IoC Detected	Alert Triggered	137.76.130.8	Log Data	NaN
39999	NaN	Alert Triggered	112.169.115.139	Log Data	Alert Data

40000 rows × 5 columns

las variables del DataFrame son categoricas. En lugar de eliminar o ignorar los valores nulos, se pueden considerar como una categoría adicional.

8. df[cols_with_nan] = df[cols_with_nan].fillna("NULL")

Para todas las columnas, sin importar que sean categóricas, u otro tipo, esta función reemplaza los valores nulos por (“-“)

9. df.describe(include="all").fillna("-")

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	Malware Indicators	Anomaly Scores	Alerts/Warnings	Attack Type
count	40000	40000	40000	40,000.00	40,000.00	40000	40,000.00	40000	40000	40000	40000	40,000.00	40000	40000
unique	39997	40000	40000	-	-	3	-	2	3	40000	2	-	2	3
top	2022-06-11 14:28:15	103.216.15.12	84.9.164.252	-	-	ICMP	-	Control	DNS	Qui natus odio asperiores nam. Optio nobis ius...	IoC Detected	-	NULL	DDoS
freq	2	1	1	-	-	13429	-	20237	13376	1	20000	-	20067	13428
mean	-	-	-	32,970.36	33,150.87	-	781.45	-	-	-	-	50.11	-	-
std	-	-	-	18,560.43	18,574.67	-	416.04	-	-	-	-	28.85	-	-
min	-	-	-	1,027.00	1,024.00	-	64.00	-	-	-	-	0.00	-	-
25%	-	-	-	16,850.75	17,094.75	-	420.00	-	-	-	-	25.15	-	-
50%	-	-	-	32,856.00	33,004.50	-	782.00	-	-	-	-	50.34	-	-
75%	-	-	-	48,928.25	49,287.00	-	1,143.00	-	-	-	-	75.03	-	-
max	-	-	-	65,530.00	65,535.00	-	1,500.00	-	-	-	-	100.00	-	-

Las diferentes clases dentro de esta columna tienen una distribución relativamente balanceada, el comando cuenta la frecuencia de cada valor único en la columna”action Taken”. Value_counts() devuelve una serie de contiene los conteos de cada valor en orden descendente.

```
10. df["Action Taken"].value_counts()
```

```
Action Taken
Blocked    13529
Ignored    13276
Logged     13195
Name: count, dtype: int64
```

Este comando cuenta la frecuencia de cada valor único en la columna “Severity Level” devuelve una serie que contiene los conteos de cada valor en orden descendente.

```
11. df["Severity Level"].value_counts()
```

```
Severity Level
Medium    13435
High      13382
Low       13183
Name: count, dtype: int64
```

El comando cuenta la frecuencia de cada vlaor único en la columna mencionada, devuelve una serie que contiene los conteos de cada tipo de ataque en orden descendente, el resultado muestra los tipos de ataque con sus frecuencias.

Es curioso como tienen frecuencias similares, cada tipo de ataque tiene uan presencia importante.

```
12. df["Attack Type"].value_counts()
```

```
Attack Type
DDoS         13428
Malware      13307
Intrusion    13265
Name: count, dtype: int64
```

Se selecciona las columnas mencionadas, se convierten a tipo categórico y se codifican en valores numéricos para calcular la correlación, se calcula la matriz de correlación para columnas codificadas.

Se ve correlación perfecta consigo misma y entre diferentes variables muy bajas, así que las variables son independientes entre si.

```
13. cols_to_predict = ["Action Taken", "Severity Level", "Attack Type"]
    df[cols_to_predict].astype("category").apply(lambda x: x.cat.codes).corr()
```

	Action Taken	Severity Level	Attack Type
Action Taken	1.00	0.00	0.00
Severity Level	0.00	1.00	0.01
Attack Type	0.00	0.01	1.00

Esta escala permite entender cuán fuerte o débil es la asociación entre dos variables categóricas. Cuanto más cerca esté el valor de 1.00, más fuerte es la asociación; y cuanto más cerca esté de 0.00, más insignificante es la relación.

Interpreting Cramer's V:

- 0.00 to 0.10: Negligible association
- 0.10 to 0.30: Weak association
- 0.30 to 0.50: Moderate association
- 0.50 to 0.70: Strong association
- 0.70 to 1.00: Very strong association

el análisis de correlación entre variables categóricas revelará relaciones significativas entre las categorías, "Proxy Information" parece ser una variable central con asociaciones significativas con casi todas las otras variables categóricas.

```
14. cramers_v_df, pvalue_df = categorical_correlation_analysis(df,
    cols_with_nan)
    cramers_v_df
```

	Malware Indicators	Alerts/Warnings	Proxy Information	Firewall Logs	IDS/IPS Alerts
Malware Indicators	1.00	0.00	0.71	0.00	0.01
Alerts/Warnings	0.00	1.00	0.71	0.00	0.00
Proxy Information	0.71	0.71	1.00	0.71	0.71
Firewall Logs	0.00	0.00	0.71	1.00	0.01
IDS/IPS Alerts	0.01	0.00	0.71	0.01	1.00

Para las correlaciones entre columnas categóricas especificadas, las variables incluidas se pueden ver arriba de la tabla, por ejemplo se pueden ver asociaciones fuertes en Proxy Information, Firewall Logs y Proxy Information, IDS/IPS Alerts y Proxy Information, con 0,71.

15. cramers_v_df, pvalue_df = categorical_correlation_analysis(df, cols_to_predict + cols_with_nan)
    cramers_v_df

	Action Taken	Severity Level	Attack Type	Malware Indicators	Alerts/Warnings	Proxy Information	Firewall Logs	IDS/IPS Alerts
Action Taken	1.00	0.01	0.01	0.01	0.00	0.71	0.00	0.00
Severity Level	0.01	1.00	0.00	0.00	0.00	0.71	0.00	0.01
Attack Type	0.01	0.00	1.00	0.01	0.01	0.71	0.00	0.00
Malware Indicators	0.01	0.00	0.01	1.00	0.00	0.71	0.00	0.01
Alerts/Warnings	0.00	0.00	0.01	0.00	1.00	0.71	0.00	0.00
Proxy Information	0.71	0.71	0.71	0.71	0.71	1.00	0.71	0.71
Firewall Logs	0.00	0.00	0.00	0.00	0.00	0.71	1.00	0.01
IDS/IPS Alerts	0.00	0.01	0.00	0.01	0.00	0.71	0.01	1.00

El resultado:

16. df[cols_to_predict + cols_with_nan].describe(include="all") resultados de **df.describe(include="all")** para las columnas **cols_to_predict** y **cols_with_nan**.

    Count, muestra el numero de valores no nulos en cada columna, unique, indica el numero de valores únicos en cada columna, top, indica el valor mas frecuente, freq, indica la frecuencia del valor mas frecuente en cada columna.

	Action Taken	Severity Level	Attack Type	Malware Indicators	Alerts/Warnings	Proxy Information	Firewall Logs	IDS/IPS Alerts
count	40000	40000	40000	40000	40000	40000	40000	40000
unique	3	3	3	2	2	20149	2	2
top	Blocked	Medium	DDoS	IoC Detected	NULL	NULL	Log Data	NULL
freq	13529	13435	13428	20000	20067	19851	20039	20050

Evaluó la influencia de los valores nulos en la corelacion entre categorías, calculo la matriz Cramer´s V y los p-values, por ejemplo en la salida podemos observar asociaciones fuertes como, "Firewall Logs" y "Proxy Information" o entre "IDS/IPS Alerts" y "Proxy Information" también se puede decir que La variable "Proxy Information" muestra una asociación fuerte (0.71) con casi todas las demás variables categóricas.

17. cramers_v_df, pvalue_df = categorical_correlation_analysis(df, cols_to_predict + cols_with_nan)
    cramers_v_df

	Action Taken	Severity Level	Attack Type	Malware Indicators	Alerts/Warnings	Proxy Information	Firewall Logs	IDS/IPS Alerts
Action Taken	1.00	0.01	0.01	0.01	0.00	0.01	0.00	0.00
Severity Level	0.01	1.00	0.00	0.00	0.00	0.01	0.00	0.01
Attack Type	0.01	0.00	1.00	0.01	0.01	0.01	0.00	0.00
Malware Indicators	0.01	0.00	0.01	1.00	0.00	0.00	0.00	0.01
Alerts/Warnings	0.00	0.00	0.01	0.00	1.00	0.00	0.00	0.00
Proxy Information	0.01	0.01	0.01	0.00	0.00	1.00	0.00	0.00
Firewall Logs	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.01
IDS/IPS Alerts	0.00	0.01	0.00	0.01	0.00	0.00	0.01	1.00

Selecciono las columnas numéricas y genero estadísticas descriptivas, en Packet Length los datos varían considerablemente, aunque relativamente simétrica, hay valores bastante dispersos. Y en Anomaly Score, también muestran una amplia variabilidad, con valores que van de 0 a 100, aunque la mediana y la media están cerca, siendo así una distribución razonablemente simétrica.

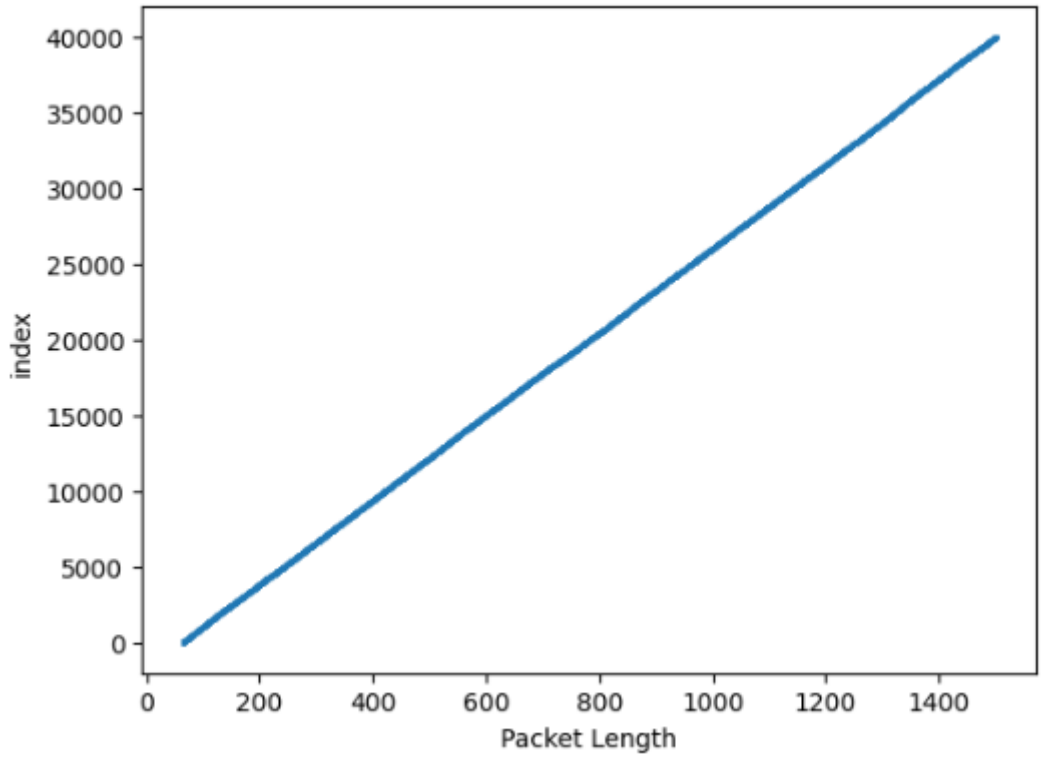
```
18. df[numerical_cols].describe()
```

	Packet Length	Anomaly Scores
count	40000.000000	40000.000000
mean	781.452725	50.113473
std	416.044192	28.853598
min	64.000000	0.000000
25%	420.000000	25.150000
50%	782.000000	50.345000
75%	1143.000000	75.030000
max	1500.000000	100.000000

Verifico la presencia de outliers (valores atípicos) en las variables numéricas, ordeno los valores de la columna "Packet Length" en orden ascendente. Reiniciio los índices del DataFrame, eliminando los índices anteriores, agrego un nuevo índice secuencial y creo un grafico de dispersión con "Packet Length" en el eje x y el nuevo índice en el eje y. El tamaño de los puntos se establece en 0.1.
El grafico que obtengo se ve una línea de subida lineal, esto indica que los valores están distribuidos de manera uniforme y creciente.

```
19. df["Packet
    Length"].sort_values().reset_index(drop=True).reset_index().plot.scatter(x="
    Packet Length", y="index", s=0.1)
```

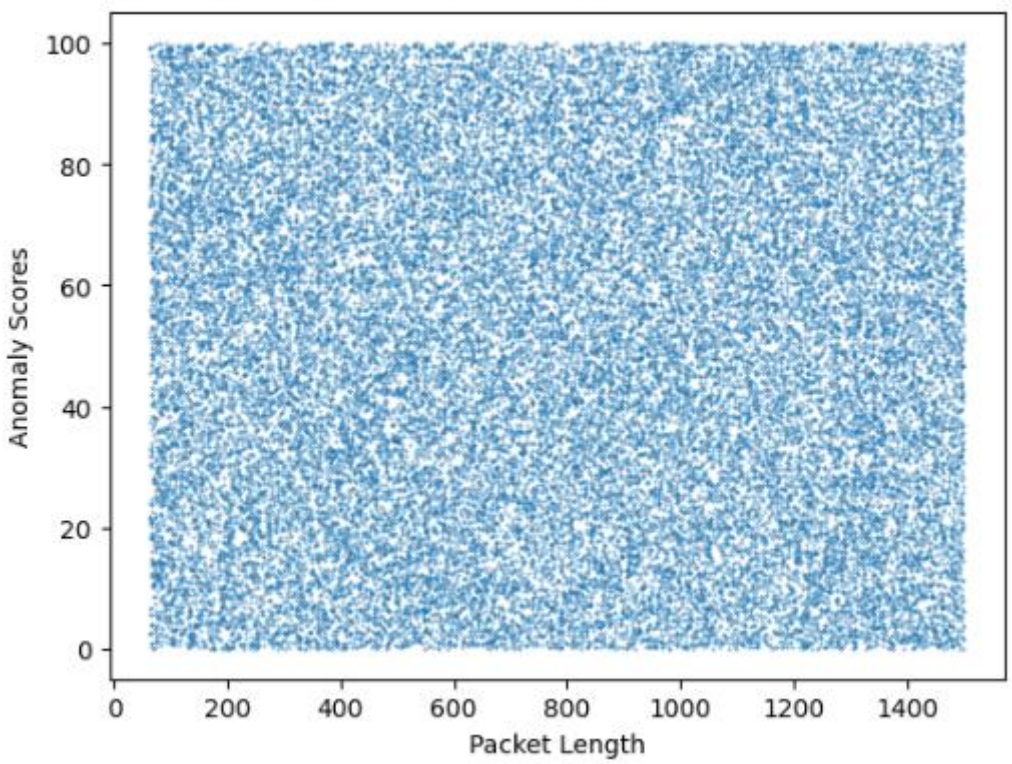
```
<Axes: xlabel='Packet Length', ylabel='index'>
```



La distribución de los datos entre Packet Length y Anomaly Scores es aleatoria. los puntos están distribuidos de manera tan aleatoria que no parece haber una estructura o patrón visible.

```
20. df[numerical_cols].plot.scatter(x="Packet Length", y="Anomaly Scores",
    s=0.1)
```

```
<Axes: xlabel='Packet Length', ylabel='Anomaly Scores'>
```



Genero estadísticas descriptivas para todas las columnas, sin importar su tipo, el método rellena todos los valores nulos con el símbolo “-” esto es útil para facilitar la lectura de las estadísticas descriptivas.

```
21. df.describe(include="all").fillna("-")
```

	Timestamp	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Payload Data	...	Action Taken	Severity Level	User Information
count	40000	40000	40000	40000.0	40000.0	40000	40000.0	40000	40000	40000	...	40000	40000	40000
unique	39997	40000	40000	29761.0	29895.0	3	-	2	3	40000	...	3	3	32389
top	2022-04-17 20:05:34	99.77.135.152	99.80.101.67	41341.0	7508.0	ICMP	-	Control	DNS	Voluptatum ut quidem ratione laboriosam commod...	...	Blocked	Medium	Ishaan Chaudhari
freq	2	1	1	6.0	6.0	13429	-	20237	13376	1	...	13529	13435	6
mean	-	-	-	-	-	-	781.452725	-	-	-	...	-	-	-
std	-	-	-	-	-	-	416.044192	-	-	-	...	-	-	-
min	-	-	-	-	-	-	64.0	-	-	-	...	-	-	-
25%	-	-	-	-	-	-	420.0	-	-	-	...	-	-	-
50%	-	-	-	-	-	-	782.0	-	-	-	...	-	-	-
75%	-	-	-	-	-	-	1143.0	-	-	-	...	-	-	-
max	-	-	-	-	-	-	1500.0	-	-	-	...	-	-	-

11 rows × 25 columns

Device Information	Network Segment	Geo-location Data	Proxy Information	Firewall Logs	IDS/IPS Alerts	Log Source
40000	40000	40000	40000	40000	40000	40000
32104	3	8723	20149	2	2	2
Mozilla/5.0 (compatible; MSIE 6.0; Windows NT ...	Segment C	Ghaziabad, Meghalaya	NULL	Log Data	NULL	Firewall
35	13408	16	19851	20039	20050	20116
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-

La columna tomestamp podría ser útil, pero como no hemos trabajado en series temporales, la elimino. La Ip de origen y destino tiene una cardialidad alta (N) lo que significa que hay tanrtos valores únicos que no aportan información útil para el análisis en este contexto. Playload Data, podría ser útil ppero como no se ha trabajado con extracción de características de texto natural, y los datos son de tipo lorem Ipsum, decido eliminarla, entonces se deciden las columnas inútiles y se eliminan.

```
22. useless_cols = ["Timestamp", "Source IP Address", "Destination IP Address",
    "Payload Data"]
    cols_to_predict = ["Action Taken", "Severity Level", "Attack Type"]
    df = df.drop(columns=useless_cols)
    df.head()
```

	Source Port	Destination Port	Protocol	Packet Length	Packet Type	Traffic Type	Malware Indicators	Anomaly Scores	Alerts/Warnings	Attack Type	...	Action Taken	Severity Level	User Information	Device Information
0	31225	17616	ICMP	503	Data	HTTP	IoC Detected	28.67	NULL	Malware	...	Logged	Low	Reyansh Dugal	Mozilla/5.0 (compatible; MSIE 8.0; Windows NT ...
1	17245	48166	ICMP	1174	Data	HTTP	IoC Detected	51.50	NULL	Malware	...	Blocked	Low	Sumer Rana	Mozilla/5.0 (compatible; MSIE 8.0; Windows NT ...
2	16811	53600	UDP	306	Control	HTTP	IoC Detected	87.42	Alert Triggered	DDoS	...	Ignored	Low	Himmat Karpe	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...
3	20018	32534	UDP	385	Data	HTTP	NULL	15.79	Alert Triggered	Malware	...	Blocked	Medium	Fateh Kibe	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_11_5; ...
4	6131	26646	TCP	1462	Data	DNS	NULL	0.52	Alert Triggered	DDoS	...	Blocked	Low	Dhanush Chad	Mozilla/5.0 (compatible; MSIE 5.0; Windows NT ...

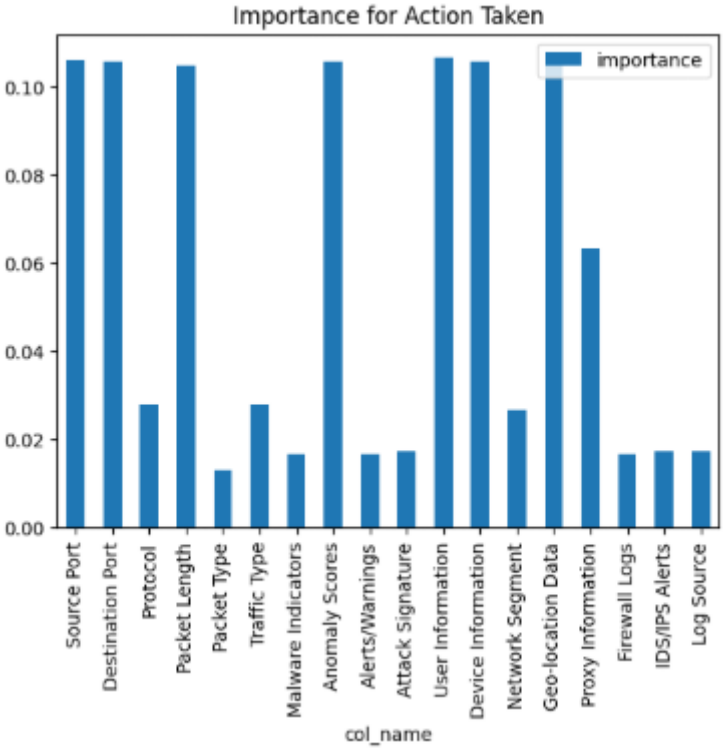
5 rows × 21 columns

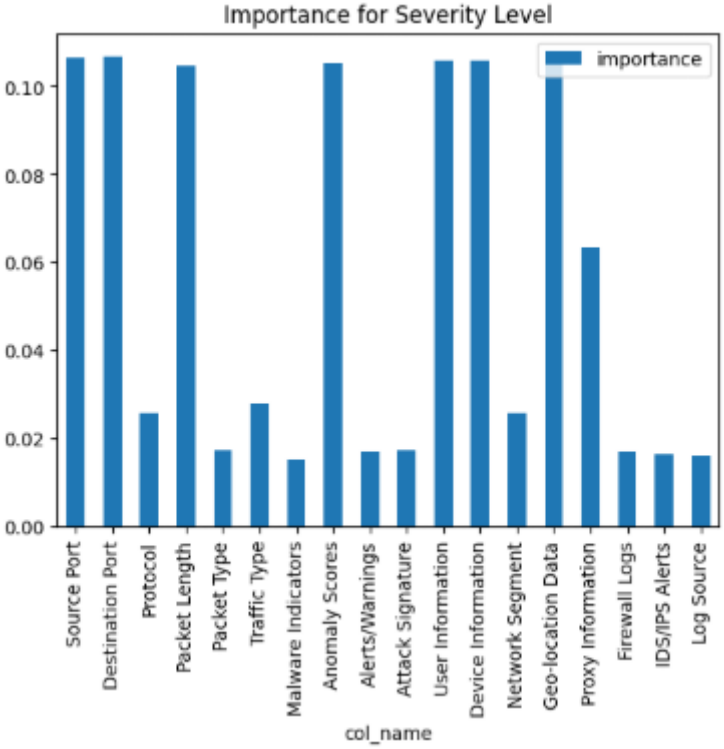
	Network Segment	Geo-location Data	Proxy Information	Firewall Logs	IDS/IPS Alerts	Log Source
)	Segment A	Jamshedpur, Sikkim	150.9.97.135	Log Data	NULL	Server
)	Segment B	Bilaspur, Nagaland	NULL	Log Data	NULL	Firewall
)	Segment C	Bokaro, Rajasthan	114.133.48.179	Log Data	Alert Data	Firewall
)	Segment B	Jaunpur, Rajasthan	NULL	NULL	Alert Data	Firewall
)	Segment C	Anantapur, Tripura	149.6.110.119	NULL	Alert Data	Firewall

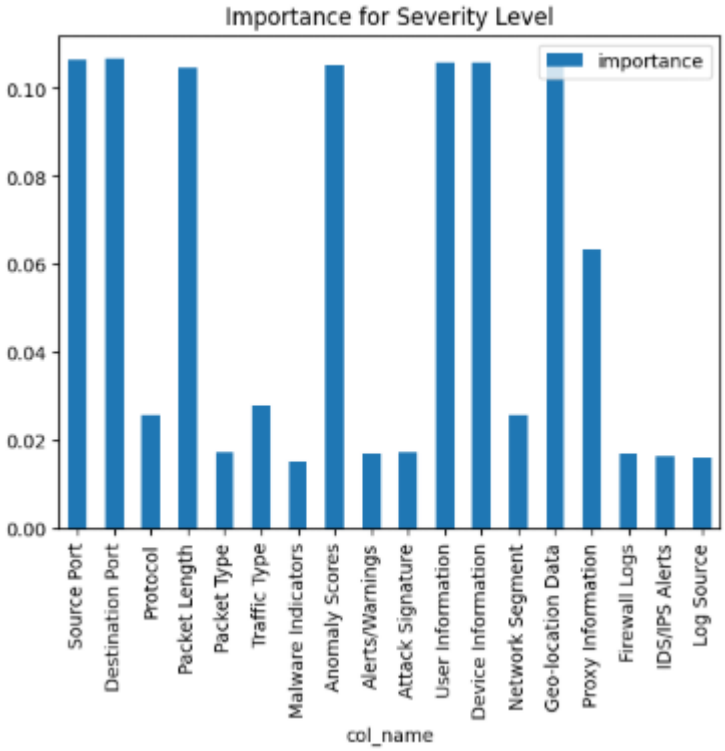
importo la clase **RandomForestClassifier** del módulo **sklearn. ensemble**, selecciono todas las columnas del DataFrame **df** que no están en **cols_to_predict** para ser utilizadas como características independientes **(X).** Convierto las columnas categóricas en códigos numéricos, lo cual es necesario para que el modelo de Random Forest pueda procesarlas.
Para cada columna en **cols_to_predict**, se entrena un modelo de Random Forest y se crea un gráfico de barras para visualizar la importancia de cada característica **(X_cols).** La importancia de las características se mide en términos de cómo influyen en la predicción de cada **y_col.**

Random Forest genera una medida de importancia para cada característica. Las características con mayores valores de importancia tienen un mayor impacto en la predicción de la variable objetivo, al visualizar estas importancias ayuda a identificar qué características son más relevantes para el problema en cuestión, es posible que el modelo esté asignando importancias similares a múltiples características porque las diferencias entre ellas no son significativas en términos de predicción.

```
23. from sklearn.ensemble import RandomForestClassifier
    X_cols = [c for c in df.columns if c not in cols_to_predict]
    X = df[X_cols]
    for c in X.select_dtypes("category"):
    X[c] = X[c].cat.codes
    for y_col in cols_to_predict:
    clf = RandomForestClassifier().fit(X, df[y_col].cat.codes)
    pd.DataFrame({"col_name": X_cols,
    "importance":clf.feature_importances_}).plot.bar(x="col_name",
    title=f"Importance for {y_col}")
    plt.plot()
```


Importance for Action Taken
importance
0.10
0.08
0.06
0.04
0.02
0.00
Source Port
Destination Port
Protocol
Packet Length
Packet Type
Traffic Type
Malware Indicators
Anomaly Scores
Alerts/Warnings
Attack Signature
User Information
Device Information
Network Segment
Geo-location Data
Proxy Information
Firewall Logs
IDS/IPS Alerts
Log Source
col_name


Importance for Severity Level
importance
0.10
0.08
0.06
0.04
0.02
0.00
Source Port
Destination Port
Protocol
Packet Length
Packet Type
Traffic Type
Malware Indicators
Anomaly Scores
Alerts/Warnings
Attack Signature
User Information
Device Information
Network Segment
Geo-location Data
Proxy Information
Firewall Logs
IDS/IPS Alerts
Log Source
col_name

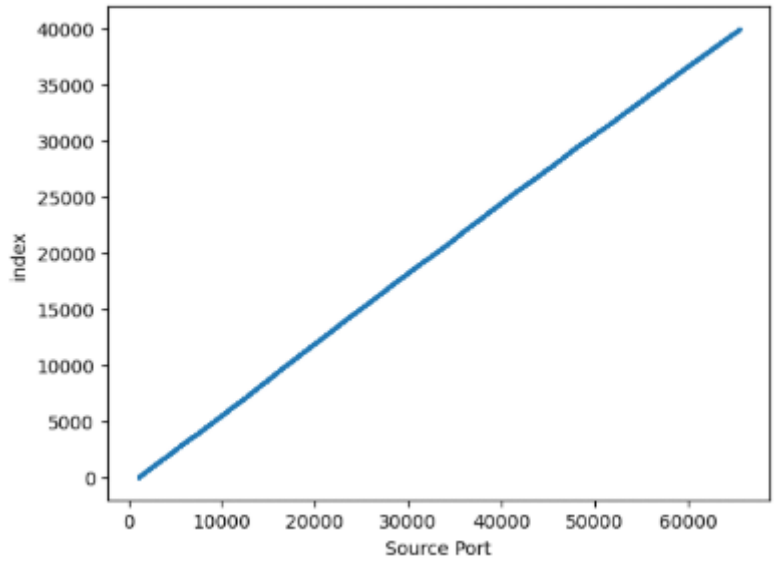


La columna **Sopource Port** no tiene valores atípicos (Outliers) convierto los valores de la columna **Source Port** a tipo entero, ordeno los valores de la columna **Source Port** en orden ascendente, reinicio los índices del DataFrame, eliminando los índices anteriores y agrego un nuevo índice secuencial al DataFrame ordenado. Creo un gráfico de dispersión con **Source Port** en el eje x y el nuevo índice en el eje y. El tamaño de los puntos se establece en 0.1.

Al verse el grafico en una línea de subida lineal, indica que los valores están distribuidos de manera uniforme y creciente.

```
24. df["Source
    Port"].astype(int).sort_values().reset_index(drop=True).reset_index().plot.sc
    atter(x="Source Port", y="index", s=0.1)
```

<Axes: xlabel='Source Port', ylabel='index'>



El propósito de esta sección del código es entrenar el modelo Random Forest y evaluar su rendimiento en predecir las columnas objetivo (cols_to_predict)

Divido los datos de entrenamiento y prueba, típico en machine learning, para entrenar el modelo, hacer predicciones y evaluar rendimiento.

Divido el dataframe X en conjuntos de entrenamiento y prueba, usando y_col como la columna objetivo. La semilla aleatoria se fija con random_state=42 para asegurar reproducibilidad.

Imprimo el informe de clasificación (precisión, recall, f1-score) y la matriz de confusión para evaluar el rendimiento del modelo en las predicciones.

Los valores de precisión, recall y f1-score están alrededor de 0.34, lo que indica que el modelo no es muy efectivo en la clasificación precisa de las categorías objetivo.

Las matrices de confusión muestran que el modelo tiene dificultad para distinguir entre las diferentes clases, ya que los valores en las diagonales no son significativamente mayores que los valores fuera de las diagonales.

Debido a que los resultados son casi aleatorios, realizar una validación cruzada (cross-validation) no aportaría valor adicional al análisis.

```
25. from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, confusion_matrix
    for y_col in cols_to_predict:
    X_train, X_test, y_train, y_test = train_test_split(X, df[y_col].cat.codes,
    random_state=42)
    clf = RandomForestClassifier().fit(X_train, y_train)
```

```
    y_pred = clf.predict(X_test)
    print(f"Classification report for {y_col}")
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
    print("-"*12)
    print()
```

```
Classification report for Action Taken
              precision    recall  f1-score   support

           0       0.34      0.38      0.36      3372
           1       0.33      0.34      0.33      3297
           2       0.34      0.30      0.32      3331

    accuracy                           0.34     10000
   macro avg       0.34      0.34      0.34     10000
weighted avg       0.34      0.34      0.34     10000

[[1276 1129  967]
 [1266 1114  917]
 [1229 1115  987]]
------------

Classification report for Severity Level
              precision    recall  f1-score   support

           0       0.32      0.37      0.35      3249
           1       0.34      0.31      0.32      3341
           2       0.35      0.32      0.33      3410

    accuracy                           0.33     10000
   macro avg       0.34      0.33      0.33     10000
weighted avg       0.34      0.33      0.33     10000

[[1218 1000 1031]
 [1265 1034 1042]
 [1293 1025 1092]]
------------

Classification report for Attack Type
              precision    recall  f1-score   support

           0       0.34      0.38      0.36      3309
           1       0.34      0.31      0.33      3392
           2       0.33      0.33      0.33      3299

    accuracy                           0.34     10000
   macro avg       0.34      0.34      0.34     10000
weighted avg       0.34      0.34      0.34     10000

[[1261  993 1055]
 [1245 1057 1090]
 [1196 1024 1079]]
------------
```

Conclusión:

La presente práctica, también conocida como una "kata" de machine learning, es un ejercicio común en procesos de selección. Se utiliza para evaluar el conocimiento técnico de los candidatos. En este caso, los datos han sido generados de manera artificial, lo que significa que no hay patrones discernibles.

Este tipo de ejercicios hay que aplicar todas las técnicas y conocimientos que se posean, paso a paso. La falta de patrones en los datos obliga a una exploración meticulosa y a la aplicación de diversas técnicas de análisis y modelado. La conclusión de este tipo de ejercicios revela si en mi caso realmente comprendo lo que estoy haciendo, más allá de seguir pasos predefinidos.

En este ejercicio específico los datos no tienen un patrón aparente, saber cómo manejar estos casos es una habilidad valiosa en el campo del análisis de datos.

El objetivo principal de esta práctica es prepararme para futuros desafíos similares en procesos de selección. Si me encuentro con una prueba semejante, puedo abordar el problema con la experiencia adquirida, aumentando así la posibilidad de éxito.

Enfrentar conjuntos de datos desafiantes sin patrones claros es una prueba de la capacidad para pensar críticamente y adaptarse a situaciones nuevas. Recordando que cada ejercicio es una oportunidad de aprendizaje, y cada fallo es un paso más hacia la maestría.

Seguire explorando, aprendiendo! El mundo del machine learning y el análisis de datos está lleno de oportunidades, sin duda es un vasto mundo.

Gracias por todo.