

NOMBRE DE LA PRÁCTICA	PIPELINE_VISUALIZACIÓN_DEL_DATA SET		NO.	1
ASIGNATURA:	Simulacion	CARRERA:	INGENIERÍA EN SISTEMAS COMPUTACIONALES	DURACIÓN DE LA PRÁCTICA (HRS) 2 HORAS

NOMBRE DEL ALUMNO: ADRIANA TREJO PATRICIO

GRUPO: 3502

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

- Salon de clase

III. Material empleado:

equipo de computo

PRACTICA

Para iniciar abrimos un nuevo notebook el cual nombraremos visualización del data-set el cual por primer paso iniciamos una una descripción breve de lo que concierne el conjunto de datos NSL-KDD



NOTA : NSL-KDD : significa "Network-based Security Laboratory - KDD" (Laboratorio de Seguridad Basada en Red - KDD). Este conjunto de datos fue desarrollado como una versión mejorada del conjunto de datos KDD'99, con el objetivo de abordar algunos de los problemas inherentes y mejorar su utilidad como un conjunto de datos de referencia para la evaluación de métodos de detección de intrusiones en redes.

Jupyter

3502_Visualizacion-del-Dataset

Last Checkpoint: 17/10/2023 (autosaved)

FileEditViewInsertCellKernelWidgetsHelp

Not TrustedPython 3 (ipykernel)

Run

Markdown

Ficheros de Datos

- KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format.
- KDDTrain+.TXT: The full NSL-KDD train set including attack-type labels and difficulty level in CSV format.
- KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.arff file. * KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.txt file.
- KDDTest+.ARFF: The full NSL-KDD test set with binary labels in format.
- KDDTest+.TXT: The full NSL-KDD test set including attack- labels and difficulty level in CSV format.
- KDDTest-21.ARFF: A subset of the KDDTest+.arff file does not include records with difficulty level of 21 out of 21.
- KDDTest-21.TXT: A subset of the KDDTest+.txt which does not include records with difficulty level of 21 out of 21.

Descarga de los ficheros de datos.

<https://www.unb.ca/cic/datasets/index.html>

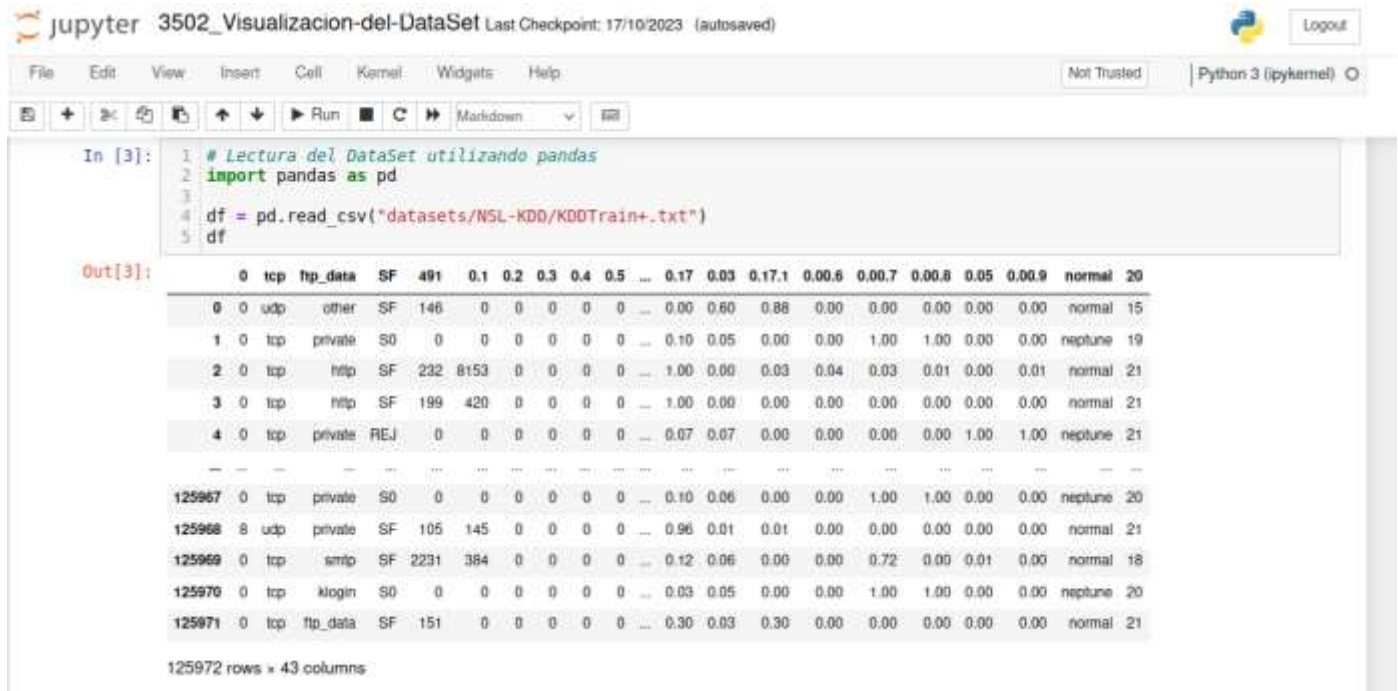
Referencias adicionales sobre el conjunto de datos

M. Tavallae, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

En la siguiente parte mostramos los ficheros que utilizaremos así como un link de descarga de los mismos , también mostramos los referencias adicionales sobre el conjunto de datos .

[illegible]

Como primer paso le daremos lectura al data set mediante funciones de python así la instrucción `with open` de Python para abrir y leer el contenido de un archivo. Así mismo el archivo que se está leyendo es "datasets/NSL-KDD/KDDTrain+.txt". Después de abrir el archivo con esta instrucción, el contenido del archivo se lee línea por línea utilizando el método `readlines()` y se almacena en la variable `df`.



Pasamos a la lectura del dataset utilizando la biblioteca pandas para leer un conjunto de datos desde un archivo de texto. En este caso, `pd.read_csv("datasets/NSL-KDD/KDDTrain+.txt")` se utiliza para leer el archivo "KDDTrain+.txt" y cargar los datos en un DataFrame llamado `df`.





utiliza el módulo `os` de Python para mostrar los archivos en el directorio "datasets/NSL-KDD/". La función `os.listdir()` devuelve una lista con los nombres de los archivos y directorios en el directorio especificado.

Asi mismo instalamos un nuevo paquete en el kernel de Jupyter Notebook utilizando la línea de comandos en el propio notebook.

NOTA :

- `sys.executable` se refiere al ejecutable de Python asociado con el kernel actual de Jupyter Notebook.
- `-m pip install` es una forma de invocar el instalador de paquetes de Python (`pip`) desde la línea de comandos.
- `liac-arff` es el nombre del paquete que estás instalando.

The screenshot shows a Jupyter Notebook window titled "jupyter Visualizacion-del-DataSet". The top bar includes a menu (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a status indicator "Not Trusted", and the Python version "Python 3 (ipykernel)".

The notebook contains three input/output pairs:

- In [6]:** A code cell with the following Python code:

```
# Lectura del conjunto de datos que se encuentra en formato .arff
import arff
with open('datasets/NSL-KDD/KDDTrain+.arff','r') as train_set:
    df = arff.load(train_set)
df.keys()
```
- Out[6]:** The output of the previous cell is: `dict_keys(['description', 'relation', 'attributes', 'data'])`
- In [7]:** A code cell with the following Python code:

```
df["data"]
```
- Out[7]:** The output of the previous cell is a list representing the first row of the dataset:

```
[0.0,  
 'tcp',  
 'ftp_data',  
 'SF',  
 491.0,  
 0.0,  
 '0',  
 0.0,  
 0.0,  
 0.0,  
 0.0,  
 '0',  
 0.0,  
 0.0,  
 0.0,  
 0.0,  
 0.0,  
 0.0]
```

Utilizamos la biblioteca `arff` para cargar un conjunto de datos en formato ARFF (Attribute-Relation File Format).


```
jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [8]: 1 df["attributes"]

Out[8]: [('duration', 'REAL'),
         ('protocol_type', ['tcp', 'udp', 'icmp']),
         ('service',
          ['aol',
           'auth',
           'bgp',
           'courier',
           'csnet_ns',
           'ctf',
           'daytime',
           'discard',
           'domain',
           'domain_u',
           'echo',
           'eco_i',
           'ecr_i',
           'efs',
           'exec',
           'finger',
           ...])
```

Con la función df accedemos a la parte del conjunto de datos ARFF que contiene información sobre los atributos la estructura de los atributos del conjunto de datos ARFF. Esto suele ser una lista de tuplas, donde cada tupla contiene información.

```
In [9]: 1 # parsear los atributos y obtener unicamente los nombres
        2 atributos = [attr[0] for attr in df["attributes"]]
        3 atributos

Out[9]: ['duration',
         'protocol_type',
         'service',
         'flag',
         'src_bytes',
         'dst_bytes',
         'hot',
         'num_failed_logins',
         'logged_in',
         'num_compromised',
         'root_shell',
         'su_attempted',
         'num_root',
         'num_file_creations',
         'num_shells',
         'num_access_files',
         'num_outbound_cmds',
         'is_host_login',
         'is_guest_login',
         'count',
         'srv_count',
         'error_rate',
         ...]
```

Parseamos los atributos y obtener únicamente los nombres. La lista df["attributes"] contiene tuplas, donde cada tupla tiene información sobre atributos.



jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [10]: 1 # leer el DataSet con pandas y facilitar su manipulacion
2 df = pd.DataFrame(df["data"], columns=atributos)
3 df
```

```
Out[10]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_f
0	0.0	tcp	ftp_data	SF	491.0	0.0	0	0.0	0.0	0.0	...	25.0	0.17	
1	0.0	udp	other	SF	146.0	0.0	0	0.0	0.0	0.0	...	1.0	0.00	
2	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	26.0	0.10	
3	0.0	tcp	http	SF	232.0	8153.0	0	0.0	0.0	0.0	...	255.0	1.00	
4	0.0	tcp	http	SF	199.0	420.0	0	0.0	0.0	0.0	...	255.0	1.00	
...
125968	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	25.0	0.10	
125969	8.0	udp	private	SF	105.0	145.0	0	0.0	0.0	0.0	...	244.0	0.96	
125970	0.0	tcp	smtp	SF	2231.0	384.0	0	0.0	0.0	0.0	...	30.0	0.12	
125971	0.0	tcp	login	S0	0.0	0.0	0	0.0	0.0	0.0	...	8.0	0.03	
125972	0.0	tcp	ftp_data	SF	151.0	0.0	0	0.0	0.0	0.0	...	77.0	0.30	

125973 rows x 42 columns

utilizamos la biblioteca pandas para crear un DataFrame a partir de los datos cargados desde el conjunto de datos ARFF y facilitar su manipulación.

jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Una vez que se ha llegado a este punto, lo ideal es construir una funcion que permita leer el conjunto de datos de manera mas limpia. Este tipo de practicas son de gran utilidad para el codigo en jupyter Notebook sea mas modular y pueda reutilizarse de manera mas sencilla para futuros ejercicios

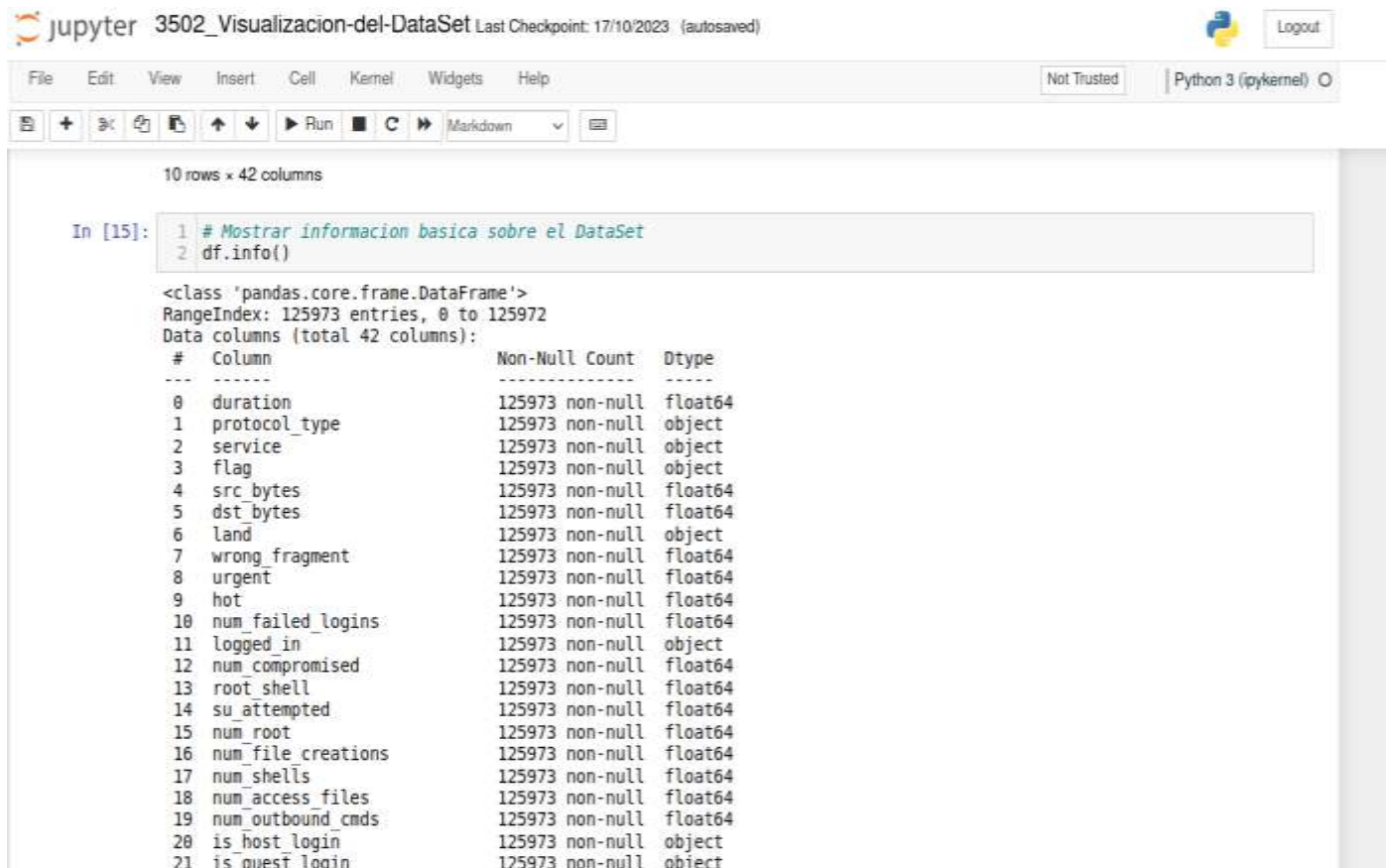
```
In [11]: 1 def load_kdd_dataset(data path):
2         """Lectura del DataSet NSL-KDD."""
3         with open(data path, 'r') as train set:
4             dataset = arff.load(train set)
5             attributes = [attr[0] for attr in dataset["attributes"]]
6             return pd.DataFrame(dataset["data"], columns=attributes)
```

```
In [12]: 1 load_kdd_dataset('datasets/NSL-KDD/KDDTrain+.arff')
```

```
Out[12]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_f
0	0.0	tcp	ftp_data	SF	491.0	0.0	0	0.0	0.0	0.0	...	25.0	0.17	
1	0.0	udp	other	SF	146.0	0.0	0	0.0	0.0	0.0	...	1.0	0.00	
2	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	26.0	0.10	
3	0.0	tcp	http	SF	232.0	8153.0	0	0.0	0.0	0.0	...	255.0	1.00	
4	0.0	tcp	http	SF	199.0	420.0	0	0.0	0.0	0.0	...	255.0	1.00	
...
125968	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	25.0	0.10	
125969	8.0	udp	private	SF	105.0	145.0	0	0.0	0.0	0.0	...	244.0	0.96	
125970	0.0	tcp	smtp	SF	2231.0	384.0	0	0.0	0.0	0.0	...	30.0	0.12	
125971	0.0	tcp	login	S0	0.0	0.0	0	0.0	0.0	0.0	...	8.0	0.03	
125972	0.0	tcp	ftp_data	SF	151.0	0.0	0	0.0	0.0	0.0	...	77.0	0.30	

La función `load_kdd_dataset` carga y procesa un conjunto de datos NSL-KDD en formato ARFF utilizando las bibliotecas `arff` y `pandas`. Esta función toma la ruta del archivo de datos como argumento (`data_path`), abre el archivo en modo de lectura, carga los datos ARFF utilizando `arff.load()`, extrae los nombres de los atributos y crea un `DataFrame` de `pandas` utilizando estos atributos y los datos.



The screenshot shows a Jupyter Notebook interface with the following elements:

- Header: "jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 {autosaved}"
- Menu bar: File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Toolbar: Run, Stop, Refresh, etc.
- Code cell content:

```
In [15]: 1 # Mostrar informacion basica sobre el DataSet
        2 df.info()
```
- Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125973 entries, 0 to 125972
Data columns (total 42 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   duration                             125973 non-null float64
1   protocol_type                         125973 non-null object
2   service                              125973 non-null object
3   flag                                  125973 non-null object
4   src_bytes                            125973 non-null float64
5   dst_bytes                            125973 non-null float64
6   land                                 125973 non-null object
7   wrong_fragment                       125973 non-null float64
8   urgent                               125973 non-null float64
9   hot                                   125973 non-null float64
10  num_failed_logins                     125973 non-null float64
11  logged_in                             125973 non-null object
12  num_compromised                       125973 non-null float64
13  root_shell                           125973 non-null float64
14  su_attempted                         125973 non-null float64
15  num_root                             125973 non-null float64
16  num_file_creations                   125973 non-null float64
17  num_shells                           125973 non-null float64
18  num_access_files                     125973 non-null float64
19  num_outbound_cmds                    125973 non-null float64
20  is_host_login                        125973 non-null object
21  is_quest_login                       125973 non-null object
```

Esto devolverá un `DataFrame` con los datos del conjunto de datos NSL-KDD, facilitando su manipulación y análisis.



```
jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [16]: 1 # Mostrar información estadística sobre el DataSet
        2 df.describe()

Out[16]:
```

	duration	src_bytes	dst_bytes	wrong_fragment	urgent	hot	num_failed_logins	num_compromised	root_shell	su_at
count	125973.00000	1.259730e+05	1.259730e+05	125973.000000	125973.000000	125973.000000	125973.000000	125973.000000	125973.000000	125973.000000
mean	287.14485	4.556674e+04	1.977911e+04	0.022687	0.000111	0.204409	0.001222	0.279250	0.001342	0.000000
std	2604.51531	5.870331e+06	4.021269e+06	0.253530	0.014366	2.149968	0.045239	23.942042	0.036603	0.000000
min	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.00000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.00000	2.750000e+02	5.160000e+02	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	42908.00000	1.379964e+09	1.309937e+09	3.000000	3.000000	77.000000	5.000000	7479.000000	1.000000	0.000000

```
8 rows x 11 columns

In [17]: 1 # Mostrar los valores únicos que tienen un atributo determinado.
        2 df["protocol_type"].value_counts()

Out[17]: tcp      102689
         udp       14993
         icmp       8291
         Name: protocol_type, dtype: int64
```

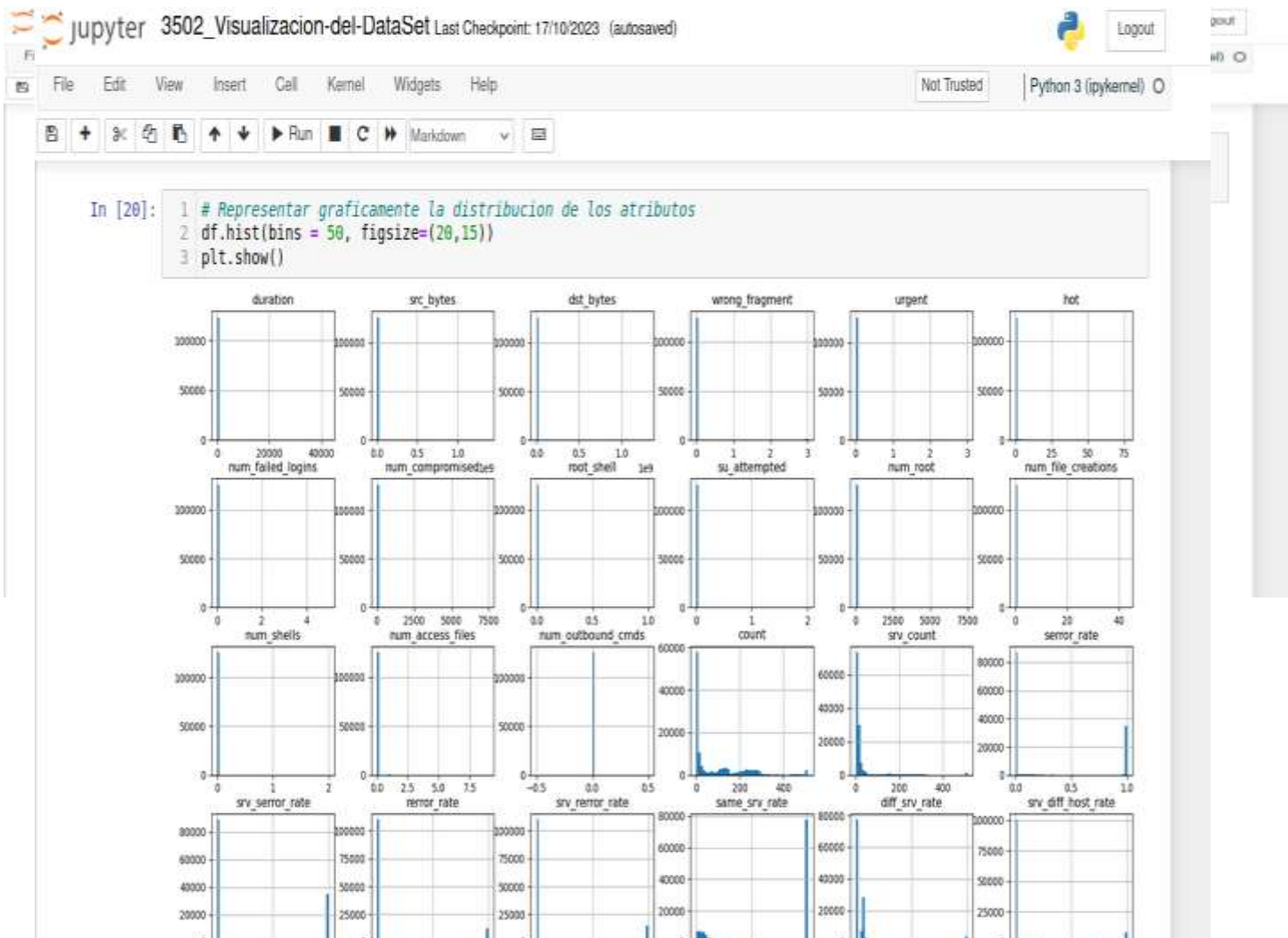
`df.describe()`, solicita las estadísticas descriptivas para todas las columnas numéricas en tu DataFrame. El método `describe()` de pandas proporciona un resumen estadístico que incluye recuento, media, desviación estándar, valores mínimo y máximo, y cuartiles (25%, 50%, y 75%).

```
jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [18]: 1 df["class"].value_counts()

Out[18]: normal      67343
         anomaly    58630
         Name: class, dtype: int64
```

`df["class"].value_counts()`, es un recuento de los valores únicos en la columna "class" de tu DataFrame y cuántas veces aparece cada valor. Este método de pandas, `value_counts()`, es útil para entender la distribución de las clases en un conjunto de datos, especialmente en problemas de clasificación.



utilizamos `%matplotlib inline` para asegurar que las visualizaciones se muestren en línea en el Jupyter Notebook y luego importa `matplotlib.pyplot` como `plt`. Luego, estás creando un histograma de la columna 'protocol_type' del DataFrame utilizando el método `hist()`.

El código utiliza `df.hist()` para crear histogramas para todas las columnas numéricas en el DataFrame, especificando 50 bins y un tamaño de figura de (20,15). Luego, `plt.show()` se utiliza para mostrar la distribución de los atributos en una sola figura.

jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

0 0.5 1.0 0 0.5 1.0 0 0.5 1.0 0 0.5 1.0

3.-Funciones Avanzadas de visualizacion de datos

Buscando Correlaciones

- Se puede calcular el coeficiente de correlacion estandar para visualizar la correlacion entre cada par de atributos.
- El coeficiente de correlacion, solo mide **correlaciones lineales**. Esto quiere decir que si x tiende a subir , mediria si y aumenta o disminuye.
- Hay que intentar buscar correlaciones sobre todo con el atributo objetivo (el que se quiere predecir), en este caso class

```
In [21]: 1 # El atributo class del DataSet tiene valores categoricos.
        2 df["class"]
```

```
Out[21]: 0      normal
        1      normal
        2    anomaly
        3      normal
        4      normal
        ...
       125968    anomaly
       125969      normal
       125970      normal
       125971    anomaly
       125972      normal
        Name: class, Length: 125973, dtype: object
```

Pasamos a las funciones avanzadas de visualizacion de datos asi mismo el código devolverá una Serie de pandas que contiene los valores de la columna "class". Puedes ver una lista de los valores únicos y la cantidad de veces que aparece cada uno.



jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

Name: class, Length: 125973, dtype: object

```
In [22]: 1 # transformar los valores del atributo class de categoricos a numericos
2 from sklearn.preprocessing import LabelEncoder
3 labelencoder = LabelEncoder()
4 df["class"] = labelencoder.fit_transform(df["class"])
5 df
```

Out[22]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_h...
0	0.0	tcp	ftp_data	SF	491.0	0.0	0	0.0	0.0	0.0	...	25.0	0.17	
1	0.0	udp	other	SF	146.0	0.0	0	0.0	0.0	0.0	...	1.0	0.00	
2	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	26.0	0.10	
3	0.0	tcp	http	SF	232.0	8153.0	0	0.0	0.0	0.0	...	255.0	1.00	
4	0.0	tcp	http	SF	199.0	420.0	0	0.0	0.0	0.0	...	255.0	1.00	
...
125968	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	...	25.0	0.10	
125969	8.0	udp	private	SF	105.0	145.0	0	0.0	0.0	0.0	...	244.0	0.96	
125970	0.0	tcp	smtp	SF	2231.0	384.0	0	0.0	0.0	0.0	...	30.0	0.12	
125971	0.0	tcp	login	S0	0.0	0.0	0	0.0	0.0	0.0	...	8.0	0.03	
125972	0.0	tcp	ftp_data	SF	151.0	0.0	0	0.0	0.0	0.0	...	77.0	0.30	

125973 rows x 42 columns

```
In [23]: 1 # Truco para no sufrir mucho.
2 # Mostrar la correlación entre los atributos del DataSet.
3 corr_matrix = df.corr(numeric_only = True)
4 corr_matrix["class"].sort_values(ascending=False)
```

utilizamos LabelEncoder de scikit-learn para transformar los valores categóricos en la columna "class" a valores numéricos

jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

125972 0.0 tcp ftp_data SF 151.0 0.0 0 0.0 0.0 0.0 ... 77.0 0.30

125973 rows x 42 columns

```
In [23]: 1 # Truco para no sufrir mucho.
2 # Mostrar la correlación entre los atributos del DataSet.
3 corr_matrix = df.corr(numeric_only = True)
4 corr_matrix["class"].sort_values(ascending=False)
```

Out[23]:

	class
same_srv_rate	1.000000
dst_host_srv_count	0.751913
dst_host_same_srv_rate	0.722535
dst_host_diff_host_rate	0.693803
num_access_files	0.119377
su_attempted	0.036701
num_file_creations	0.022448
root_shell	0.021271
hot	0.020285
num_root	0.013083
num_compromised	0.011452
num_shells	0.010198
num_failed_logins	0.009472
urgent	0.003755
srv_count	0.002787
dst_bytes	-0.000771
src_bytes	-0.004118
duration	-0.005921
dst_host_srv_diff_host_rate	-0.048785
dst_host_same_src_port_rate	-0.062332
wrong_fragment	-0.092444
diff_srv_rate	-0.095905
...	-0.203660

el código muestra la correlación entre los atributos numéricos y la columna "class" en el DataFrame. La matriz de correlación (`corr_matrix`) se calcula utilizando el método `corr()` de pandas y se limita a atributos numéricos utilizando `numeric_only=True`. Luego, se extraen las correlaciones de la columna "class" con cada atributo y se imprimen en orden descendente.

jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [25]: 1 # Mostrar la correlacion lineal entre todos los atributos del conjunto de datos
        2 df.corr(numeric_only = True)
```

Out[25]:

	duration	src_bytes	dst_bytes	wrong_fragment	urgent	hot	num_failed_logins	num_compromised	root_shell	su_attempted
duration	1.000000	0.070737	0.034878	-0.009866	0.003830	0.000705	0.009528	0.042679	0.052791	0.00
src_bytes	0.070737	1.000000	0.000204	-0.000693	-0.000059	0.000295	-0.000208	-0.000086	-0.000272	-0.00
dst_bytes	0.034878	0.000204	1.000000	-0.000440	0.000248	-0.000344	0.000504	0.001233	0.001069	0.00
wrong_fragment	-0.009866	-0.000693	-0.000440	1.000000	-0.000692	-0.008508	-0.002418	-0.001044	-0.003280	-0.00
urgent	0.003830	-0.000059	0.000248	-0.000692	1.000000	0.000293	0.097507	0.033329	0.075199	0.00
hot	0.000705	0.000295	-0.000344	-0.008508	0.000293	1.000000	0.003715	0.002014	0.015379	0.00
num_failed_logins	0.009528	-0.000208	0.000504	-0.002418	0.097507	0.003715	1.000000	0.019085	0.032567	0.07
num_compromised	0.042679	-0.000086	0.001233	-0.001044	0.033329	0.002014	0.019085	1.000000	0.224872	0.30
root_shell	0.052791	-0.000272	0.001069	-0.003280	0.075199	0.015379	0.032567	0.224872	1.000000	0.60
su_attempted	0.087183	-0.000186	0.001133	-0.002187	0.097710	0.000130	0.073175	0.362702	0.609083	1.00
num_root	0.045519	-0.000093	0.001229	-0.001108	0.032470	0.001510	0.018112	0.998833	0.243349	0.30
num_file_creations	0.099116	-0.000179	0.000089	-0.002343	0.024918	0.028716	0.021774	0.015976	0.064919	0.04
num_shells	-0.001593	-0.000134	-0.000083	-0.001665	-0.000144	0.004723	-0.000503	0.001338	0.116648	0.00
num_access_files	0.070420	-0.000309	0.000339	-0.003689	0.010803	-0.001987	0.000652	0.299631	0.365152	0.50
num_outbound_cmds	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
count	-0.079042	-0.005152	-0.003543	-0.020819	-0.005615	-0.068697	-0.019544	-0.008434	-0.025630	-0.01
srv_count	-0.039470	-0.002792	-0.001754	0.024457	-0.002848	-0.034575	-0.009880	-0.004279	-0.011892	-0.00
error_rate	-0.069873	-0.003228	-0.003059	-0.043316	-0.004929	-0.059083	-0.015254	-0.005297	-0.018220	-0.00

Se utiliza el método `corr()` de pandas para calcular la correlación lineal entre todos los atributos numéricos del conjunto de datos. Este código generará y mostrará una matriz donde cada entrada representa la correlación lineal entre dos atributos.

Jupyter 3502_Visualizacion-del-DataSet Last Checkpoint: 17/10/2023 (autosaved)



Logout

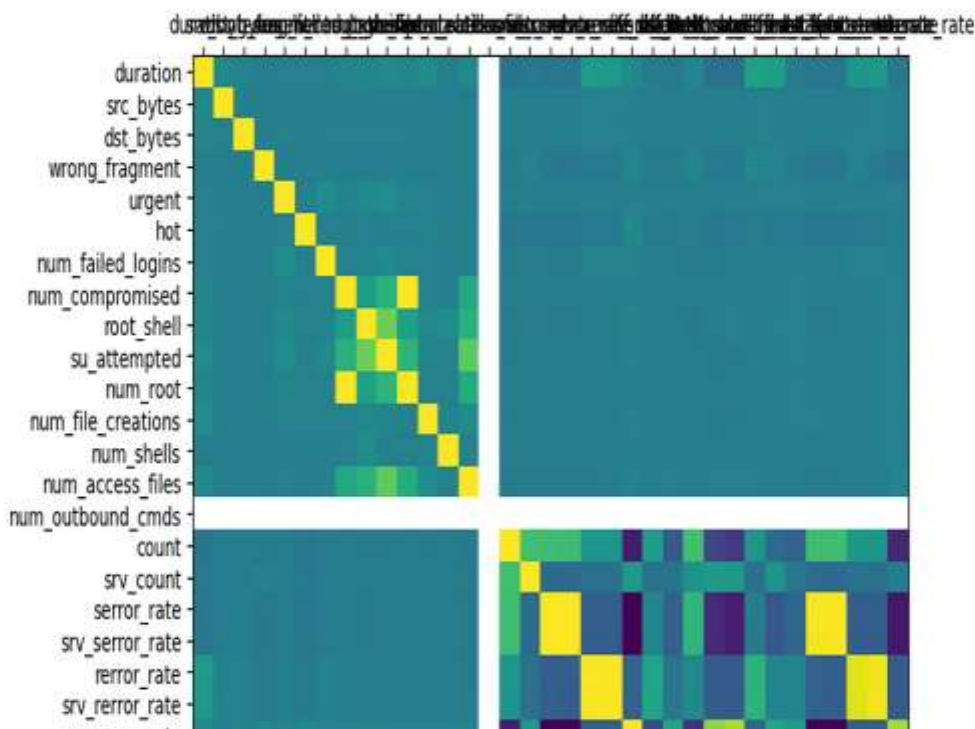
File Edit View Insert Cell Kernel Widgets Help

Not Trusted

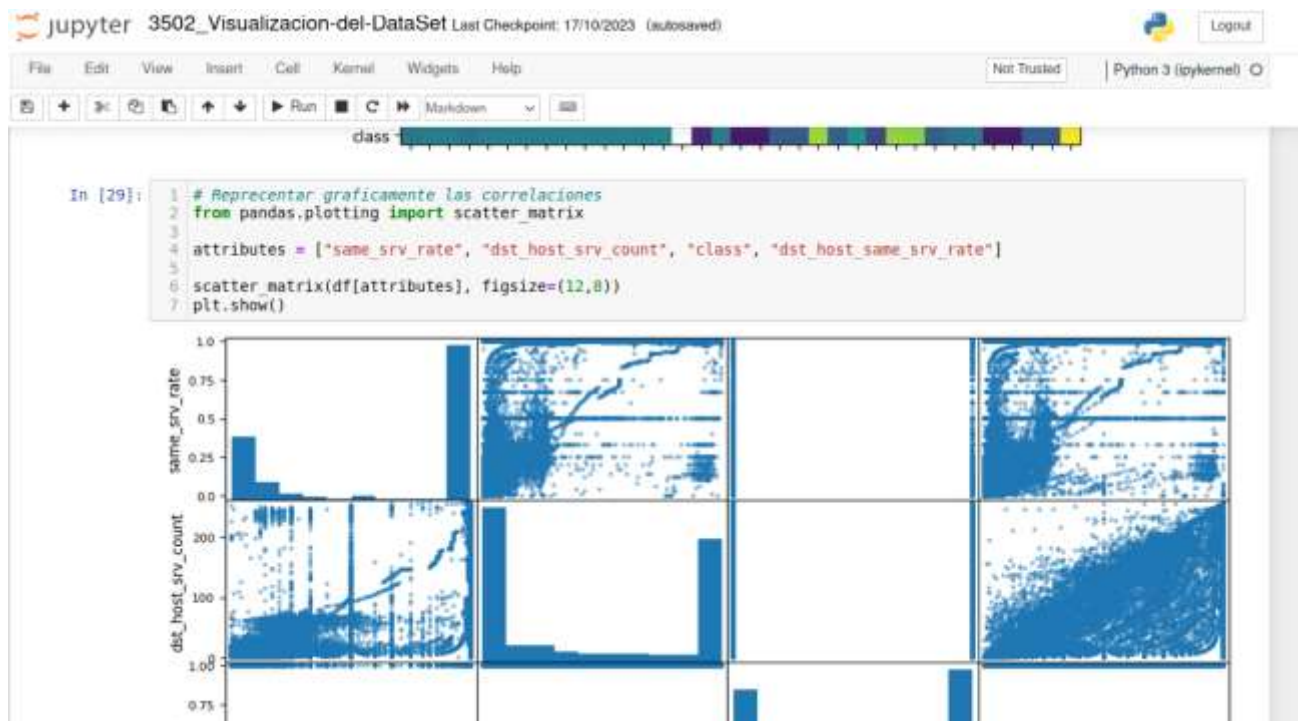
Python 3 (ipykernel)

Run

```
In [28]: 1 # Representar graficamente la matriz de correlacion
2 corr = df.corr(numeric only = True)
3 fig, ax = plt.subplots(figsize=(8,8))
4 ax.matshow(corr)
5 plt.xticks(range(len(corr.columns)), corr.columns);
6 plt.yticks(range(len(corr.columns)), corr.columns);
```



código utiliza `matshow` para mostrar la matriz de correlación como una imagen en el gráfico. Las etiquetas en los ejes x e y se configuran con los nombres de las columnas para facilitar la interpretación. El tamaño de la figura se establece en (8,8), pero puedes ajustarlo según tus preferencias.



Este

código genera una matriz de dispersión que muestra gráficamente las relaciones entre los pares de atributos especificados. En cada celda de la matriz, habrá un gráfico de dispersión que representa la relación entre dos atributos. La diagonal principal contendrá histogramas de cada atributo.

Esto es útil para visualizar las relaciones y patrones entre los atributos seleccionados. Puedes ajustar la lista de atributos en la variable `attributes` según tus necesidades específicas. La matriz de dispersión es una herramienta poderosa para explorar visualmente las correlaciones en un conjunto de datos.

V. Conclusiones

La creación de transformadores propios permite mantener el código mucho más limpio y estructurado a la hora de preparar los datos para los algoritmos de mercado libre además facilita la reutilización de código para otros conjuntos. También es en el ámbito de la ciencia de datos y el aprendizaje automático. Al desarrollar transformadores personalizados, se logra una mayor flexibilidad para adaptar los datos de entrada a las necesidades específicas del modelo, permitiendo así una mejor representación y extracción de características relevantes. Los pipelines personalizados, por otro

lado, facilitan la gestión eficiente de múltiples etapas de procesamiento de datos, desde la preparación hasta el modelado.