«Talento Tech»

Data Analytics con Python

Clase 06











Clase N° 6| Conceptos básicos

Temario:

- Análisis de datos con Pandas
- Filtrado y selección de datos.
- Agrupación y agregación.
- Generación de estadísticas descriptivas.







Análisis de datos con Pandas

El análisis de datos con Pandas ofrece un conjunto poderoso y versátil de herramientas para manejar y manipular datos. Esta biblioteca facilita tareas como la limpieza, transformación y exploración de datos, permitiendo realizar análisis complejos de manera eficiente.

Con Pandas, puedes trabajar con datos tabulares de forma intuitiva, desde la carga y visualización hasta la preparación para modelos más avanzados. Es una herramienta esencial para quienes buscan comprender y extraer valor de la información, abriendo un mundo de posibilidades en el análisis de datos.

Información general

Continuaremos trabajando con nuestro dataset de películas y series de Amazon Prime Video.

Exploraremos varios métodos para obtener información general del dataset.

Leer el dataset y transformarlo en dataframe

Para empezar, montaremos nuestro Google Drive para acceder al archivo del dataset. A continuación, importaremos la biblioteca Pandas y utilizaremos el método read_csv para cargar el archivo en una variable llamada df, convirtiéndolo así en un DataFrame. Código:

```
from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

df = pd.read_csv('Ruta del archivo')
```





Una vez ejecutado este proceso, estaremos listos para comenzar a analizar los datos contenidos en el DataFrame.

Info, Describe, sample y duplicated

Info()

El método info() en Pandas proporciona un resumen general de un DataFrame, incluyendo el número de filas y columnas, los nombres y tipos de datos de las columnas, y el conteo de valores no nulos. Es útil para obtener una visión rápida de la estructura y el contenido del DataFrame.

Ejemplo:

Código:

```
from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

df = pd.read_csv('Ruta del archivo')

df.info()
```





```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9668 entries, 0 to 9667
Data columns (total 12 columns):
    Column
                 Non-Null Count Dtype
0
    show_id
                 9668 non-null
                                 object
1
    type
                 9668 non-null
                                 object
2
                 9668 non-null
    title
                                 object
 3
    director
                 7585 non-null
                                 object
4
    cast
                 8435 non-null
                                 object
5 country
                672 non-null
                                 object
    date_added 155 non-null
6
                                 object
    release_year 9668 non-null
                                 int64
8
    rating
                  9331 non-null
                                 object
9
    duration
                 9668 non-null
                                 object
10 listed in
                  9668 non-null
                                 object
    description 9668 non-null
                                 object
dtypes: int64(1), object(11)
memory usage: 906.5+ KB
```

Describe()

El método describe() proporciona un resumen estadístico descriptivo de las columnas numéricas del DataFrame. Incluye estadísticas como la media, desviación estándar, mínimo, y percentiles.

Ejemplo:

Código:





```
df = pd.DataFrame(data)
print(df.describe())
```

Consola:

0		columna1	Columna2
	count	5.000000	5.000000
	mean	3.000000	30.000000
	std	1.581139	15.811388
	min	1.000000	10.000000
	25%	2.000000	20.000000
	50%	3.000000	30.000000
	75%	4.000000	40.000000
	max	5.000000	50.000000

Sample()

El método sample() se utiliza para obtener una muestra aleatoria de filas de un DataFrame. Es útil para explorar una parte representativa de los datos.

Ejemplo: Código:

df.sample(5)







Como pueden ver, dentro del paréntesis se especifica la cantidad de muestras que deseamos visualizar en la consola. Si no se indica ningún número, se mostrará un solo elemento por defecto.

Duplicated()

El método duplicated() en Pandas se utiliza para identificar filas duplicadas en un DataFrame. Devuelve una Serie booleana que indica si cada fila es un duplicado o no.

Código:

df.duplicated()







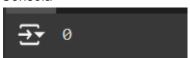
Este método devolverá True si el elemento en ese índice está duplicado y False si no lo está.

Además, podemos sumar todos los valores resultantes para verificar duplicados: si la suma es 0, significa que no hay elementos duplicados.

Ejemplo

Codigo:

df.duplicated().sum()







En conclusión, los métodos info(), describe(), sample(), y duplicated() de Pandas son herramientas clave para el análisis de datos.

info() proporciona un resumen estructural del DataFrame, ayudando a entender rápidamente su formato, tipos de datos y presencia de valores nulos.

describe() ofrece estadísticas descriptivas esenciales para conocer la distribución y características básicas de los datos numéricos.

sample() permite extraer muestras aleatorias del DataFrame, facilitando la exploración y verificación de datos sin necesidad de trabajar con el conjunto completo.

duplicated() identifica filas duplicadas, lo cual es crucial para la limpieza de datos y la detección de redundancias.

Estos métodos permiten una gestión eficiente y una comprensión más profunda de los datos, optimizando el proceso de análisis y preparación para modelos o informes.

Columnas

Para obtener los valores de una columna específica, utilizaremos la variable que contiene el dataset y, entre corchetes, especificaremos el nombre de la columna como una cadena de texto. Esto nos devolverá una Serie que contiene todos los datos de esa columna.

em		

Código:

df['type']





```
Movie
1
          Movie
2
          Movie
3
          Movie
4
          Movie
9663
          Movie
9664
        TV Show
9665
          Movie
        TV Show
9666
          Movie
9667
Name: type, Length: 9668, dtype: object
```

Unique()

El método unique() en Pandas se utiliza para encontrar los valores únicos en una columna de un DataFrame o en una Serie. Es útil para identificar los distintos valores presentes y entender la diversidad de datos en una columna específica.

Código:

df['release_year'].unique()





Consola:

```
array([2014, 2018, 2017, 1989, 2016, 1994, 2020, 2019, 2008, 2001, 1941, 1991, 2005, 2015, 2011, 2013, 1949, 2007, 2002, 1955, 1959, 1983, 2009, 2012, 2010, 1986, 1988, 1920, 1936, 1992, 2021, 1993, 2006, 1948, 1946, 1944, 1935, 1985, 1937, 1970, 1945, 1939, 1996, 1997, 1974, 1938, 1978, 2004, 1943, 1975, 1960, 1934, 1940, 1961, 2003, 2000, 1967, 1995, 1951, 1932, 1999, 1963, 1969, 1952, 1947, 1929, 1990, 1925, 1968, 1987, 1942, 1979, 1980, 1981, 1976, 1966, 1973, 1956, 1972, 1950, 1953, 1982, 1977, 1933, 1958, 1984, 1998, 1924, 1922, 1926, 1954, 1930, 1971, 1965, 1931, 1923, 1962, 1964, 1957, 1927])
```

Como podemos ver en el ejemplo, es posible obtener los valores únicos de una columna específica. El resultado será una lista que contiene todos esos valores distintos.

Shape()

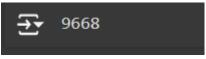
El atributo shape en Pandas proporciona la dimensión de un DataFrame o Serie. Devuelve una tupla que contiene el número de filas y el número de columnas del DataFrame, o simplemente el número de elementos en una Serie.

Ejemplo:

Código:

```
total_datos = df.shape[0]
print(total_datos)
```

Consola



Colocamos 0 dentro del atributo shape para obtener el número de filas del DataFrame y luego imprimimos este valor en la consola para visualizarlo.





Value_count()

Antes de continuar vamos a cambiar el nombre de nuestras columnas como vimos la clase anterior.

Código:

El método value_counts() en Pandas se utiliza para contar la frecuencia de aparición de cada valor único en una columna o Serie. Es útil para obtener un resumen de la distribución de datos categóricos o discretos.

Ejemplo:

Código:

```
df['tipo'].value_counts()
```

Consola:

```
→ tipo
Movie 7814
TV Show 1854
Name: count, dtype: int64
```

Como se muestra en el ejemplo, solicitamos que se nos muestre la frecuencia de cada valor en la columna 'tipo', obteniendo así el conteo de películas y series en el DataFrame.





dropna()

El método dropna() en Pandas se utiliza para eliminar filas o columnas que contienen valores NaN (Not a Number) o None de un DataFrame o Serie. Es útil para limpiar los datos antes de realizar análisis o modelado, asegurando que los conjuntos de datos sean completos y estén libres de valores faltantes.

Ejemplo:

```
Código:
```

```
paises = df['pais'].dropna()
paises
```

Consola:

		pais
	0	Canada
	1	India
	2	United States
	3	United States
	4	United Kingdom
	5	United Kingdom
	6	United States

Como se muestra en la imagen, la consola muestra una serie con todos los países de la columna "país", excluyendo los elementos vacíos.





groupby(), size() y sort_values()

El método groupby() en Pandas se utiliza para agrupar datos similares y realizar cálculos sobre esos grupos. Por ejemplo, en nuestro DataFrame, podemos agrupar los datos por países y, utilizando el método size(), podemos contar cuántas veces aparece cada país. Luego, con sort_values(), podemos ordenar estos datos en orden alfabético.

Código:

```
# agrupo por pais
grupo_paises = df.groupby('pais')
# cuento la cantidad por pais ordenando de mayor a menor
grupo_paises.size().sort_values(ascending=False)
```

₹		pais	
	United States	253	
	India	229	
	United Kingdom	28	
	Canada	16	
	United Kingdom, United States	12	
	Italy	8	
	Spain	8	
	Canada, United States	7	
	United States, United Kingdom	6	





Como se muestra en la imagen, la consola nos indica la cantidad de veces que se repite cada elemento, tal como especificamos en el código.

Filtros lógicos

Para filtrar datos en Pandas, debes acceder al DataFrame y aplicar una condición lógica dentro de los corchetes para seleccionar las filas que cumplen con esa condición.

Por ejemplo, si queremos ver las películas y series estrenadas en el año 2020, debemos escribir un filtro que compare el año de estreno con 2020 y aplicarlo al DataFrame.

Código:

```
estrenos2020 = df[df["anio_lanzamiento"] == 2020]
estrenos2020
```

50110014													
													
	9528	s9529	Movie	THE CHRISTMAS EDITION	HYBRID LLC	NaN	NaN	NaN	2020	ALL	88 min	Drama	Finding herself at a crossroads, up-and-coming
	9555	s9556	Movie	DEAR CHRISTMAS	Hartbreak Films, Inc.	NaN	NaN	NaN	2020	ALL	88 min	Drama	Natalie's the host of a popular podcast, Holid
	9577	s9578	Movie	Surviving America	Omegia Keeys	Jesse Bingham, Gerald Griggs, Natasha Pearson,	NaN	NaN	2020		96 min	Documentary, Special Interest	In the wake of the new Civil Rights Movement i
	9581	s9582	TV Show	Spoopy Movie Time	NaN	Scott Nielsen, Natalie Perez	NaN	NaN	2020	18+	1 Season	Comedy, Horror	Animated horror hosts watch public domain film
	9619	s9620	Movie	The Shadow Side	Max Coronel	Clara Kovacic, Germán Baudino, Gon Spina	NaN	NaN	2020		54 min	Horror, Suspense	Laura meets punctually every night with her fr
962 rows × 12 columns													





Como se muestra en la imagen, el resultado es un nuevo DataFrame con menos filas que el original, que solo incluye las películas y series estrenadas en 2020.





Desafío Nº 6:

Teniendo en cuenta el dataframe que usamos la clase pasada, crear dos dataframe con las películas de dos años específicos, verificar cuántas películas y cuantos TV show hay en cada año ¿en qué año se lanzaron más Películas? ¿y más shows de tv? Mostrar en consola los resultados.

Colab desafío 6



