Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Inteligencia Artificial Aplicada
Practica 3
Conceptos básicos de pandas
Ana Sofía Medina Martínez

Fecha 29/08/2024

Objetivo

Que el estudiante aprenda a utilizar la biblioteca Pandas para la carga de datos, exploración de datasets, filtrada, transformación y agregación de datos.

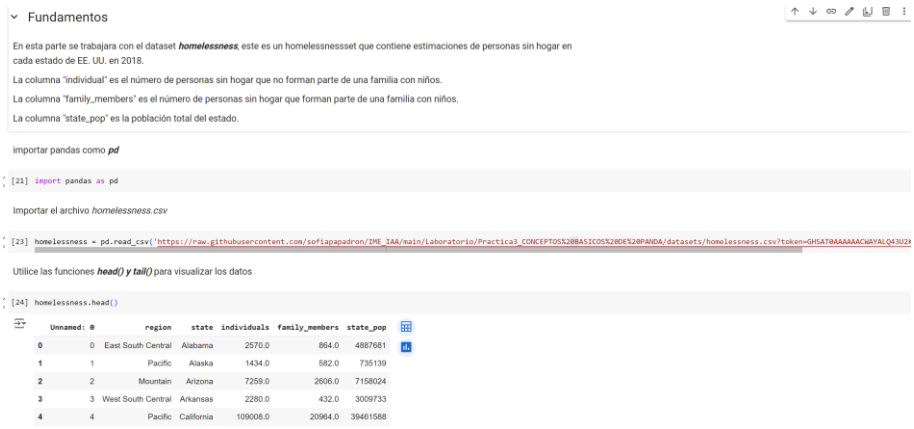
Procedimiento

3.1.- Inicie Jupyter Notebooks y abra los notebooks "fundamentos", “agregacion” e“indexado” proporcionados.

3.2.- Siga las instrucciones en los notebooks para explorar los conceptos básicos de Pandas.

3.3.- Resuelva los ejercicios proporcionados en el notebook “ejercicios”.

Resultados



Fundamentos

En esta parte se trabajara con el dataset ***homelessness***, este es un homelessnessset que contiene estimaciones de personas sin hogar en cada estado de EE. UU. en 2018.

La columna "individual" es el número de personas sin hogar que no forman parte de una familia con niños.

La columna "family_members" es el número de personas sin hogar que forman parte de una familia con niños.

La columna "state_pop" es la población total del estado.

importar pandas como ***pd***

```
[21] import pandas as pd
```

Importar el archivo *homelessness.csv*

```
[23] homelessness = pd.read_csv('https://raw.githubusercontent.com/sofiapapadron/IME_IAA/main/Laboratorio/Practica3_CONCEPTOS%20BASICOS%20DE%20PANDA/datasets/homelessness.csv?token=GHSAT0AAAAACWAYALQ43U2
```

Utilice las funciones ***head() y tail()*** para visualizar los datos

```
[24] homelessness.head()
```

	Unnamed: 0	region	state	individuals	family_members	state_pop
0	0	East South Central	Alabama	2570.0	864.0	4887681
1	1	Pacific	Alaska	1434.0	582.0	735139
2	2	Mountain	Arizona	7259.0	2606.0	7158024
3	3	West South Central	Arkansas	2280.0	432.0	3009733
4	4	Pacific	California	109008.0	20964.0	39461588

```
homelessness.tail()
```

	Unnamed: 0	region	state	individuals	family_members	state_pop
46	46	South Atlantic	Virginia	3928.0	2047.0	8501286
47	47	Pacific	Washington	16424.0	5880.0	7523869
48	48	South Atlantic	West Virginia	1021.0	222.0	1804291
49	49	East North Central	Wisconsin	2740.0	2167.0	5807406
50	50	Mountain	Wyoming	434.0	205.0	577601

Imprima las columnas y el tipo de dato de cada columna utilizando la función ***info()***

```
[26] homelessness.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      51 non-null     int64
 1   region          51 non-null     object
 2   state           51 non-null     object
 3   individuals     51 non-null     float64
 4   family_members  51 non-null     float64
 5   state_pop       51 non-null     int64
dtypes: float64(2), int64(2), object(2)
memory usage: 2.5+ KB
```

Imprima las ***columnas*** del homelessnessset

```
[27] homelessness.columns
```

```
Index(['Unnamed: 0', 'region', 'state', 'individuals', 'family_members',
       'state_pop'],
      dtype='object')
```

Ordenar y filtrar Datos

Ordene el dataset *homelessness* por *individuals*, de menor a mayor, guarde el sesultado en una variable llamada *homelessness_ind*.

Imprima los primeros elementos del resultado.

```
[28] homelessness_ind = homelessness.sort_values('individuals')
     homelessness_ind.head()
```

	Unnamed: 0	region	state	individuals	family_members	state_pop
50	50	Mountain	Wyoming	434.0	205.0	577601
34	34	West North Central	North Dakota	467.0	75.0	758080
7	7	South Atlantic	Delaware	708.0	374.0	965479
39	39	New England	Rhode Island	747.0	354.0	1058287
45	45	New England	Vermont	780.0	511.0	624358

Next steps: Generate code with homelessness_ind | View recommended plots | New interactive sheet

Ordene el homelessnessset *homelessness* primero por *region (ascendente)* y luego por *family_members (descendente)*, guarde el sesultado en una variable llamada *homelessness_reg_fam*.

Imprima los primeros elementos del resultado.

```
[29] homelessness_reg_fam = homelessness.sort_values(['region', 'family_members'], ascending=[True, False])
     homelessness_reg_fam.head()
```

	Unnamed: 0	region	state	individuals	family_members	state_pop
13	13	East North Central	Illinois	6752.0	3891.0	12723071
35	35	East North Central	Ohio	6929.0	3320.0	11676341
22	22	East North Central	Michigan	5209.0	3142.0	9984072
49	49	East North Central	Wisconsin	2740.0	2167.0	5807406

Crea un homelessnessFrame llamado *state_fam* que contenga únicamente la columna *state* y *family_members*.

```
[30] state_fam = homelessness[['state', 'family_members']]
     state_fam.head()
```

	state	family_members
0	Alabama	864.0
1	Alaska	582.0
2	Arizona	2606.0
3	Arkansas	432.0
4	California	20964.0

Next steps: Generate code with state_fam | View recommended plots | New interactive sheet

De *state_fam* Obtenga todos los renglones donde *family_members* sea mayor o igual a 2000.

```
state_fam[state_fam['family_members'] >= 2000]
```

	state	family_members
2	Arizona	2606.0
4	California	20964.0
5	Colorado	3250.0
8	District of Columbia	3134.0
9	Florida	9587.0
10	Georgia	2556.0
11	Hawaii	2399.0
13	Illinois	3891.0
20	Maryland	2230.0
21	Massachusetts	13257.0

De *state_fam* Obtenga todos los renglones donde *state* sea New York o Washington

```
[32] state_fam[(state_fam['state'] == 'New York') | (state_fam['state'] == 'Washington')]
```

	state	family_members
32	New York	52070.0
47	Washington	5880.0

Generar nuevas columnas

Agrega una nueva columna llamada ***total***, esta columna tendrá la suma de *individuals* y *family_members*.

```
[34] homelessness['total'] = homelessness['individuals'] + homelessness['family_members']
     homelessness.head()
```

	Unnamed: 0	region	state	individuals	family_members	state_pop	total
0	0	East South Central	Alabama	2570.0	864.0	4887681	3434.0
1	1	Pacific	Alaska	1434.0	582.0	735139	2016.0
2	2	Mountain	Arizona	7259.0	2606.0	7158024	9865.0
3	3	West South Central	Arkansas	2280.0	432.0	3009733	2712.0
4	4	Pacific	California	109008.0	20964.0	39461588	129972.0

Next steps: Generate code with homelessness | View recommended plots | New interactive sheet

Funciones de agregación

En esta sección se trabajará con el dataset de ventas de Walmart.

Este dataset contiene ventas semanales de varias tiendas de los Estados Unidos.

Funciones de agregación

En esta sección se trabajará con el dataset de ventas de Walmart.

Este dataset contiene ventas semanales de varias tiendas de los Estados Unidos.

Funciones estadisticas

Importe el dataset ***sales_subset*** e imprima los primeros datos usando la función *head()*.

```
[36] sales_subset = pd.read_csv('https://raw.githubusercontent.com/sofiapapadron/IME_IAA/main/Laboratorio/Practica3_CONCEPTOS%20BASICOS%20DE%20PANDA/datasets/sales_subset.csv?token=GHSAT0AAAAAACWAYALRLRQSL
     sales_subset.head()
```

	Unnamed: 0	store	type	department	date	weekly_sales	is_holiday	temperature_c	fuel_price_usd_per_l	unemployment
0	0	1	A	1	2010-02-05	24924.50	False	5.727778	0.679451	8.106
1	1	1	A	1	2010-03-05	21827.90	False	8.055556	0.693452	8.106
2	2	1	A	1	2010-04-02	57258.43	False	16.816667	0.718284	7.808
3	3	1	A	1	2010-05-07	17413.94	False	22.527778	0.748928	7.808
4	4	1	A	1	2010-06-04	17558.09	False	27.050000	0.714586	7.808

Next steps: Generate code with sales_subset | View recommended plots | New interactive sheet

Obtenga el promedio y la mediana de la columna *weekly_sales*

```
[38] sales_subset['weekly_sales'].mean()
```

```
23843.95014850566
```

```
sales_subset['weekly_sales'].median()
```

```
12049.064999999999
```

Imprima la fecha minima de la columna *date*.

```
[40] sales_subset['date'].min()
```

```
'2010-02-05'
```

Funciones para variables categoricas

Se desea saber cuantos tipos de tiendas diferentes existen, elimine todas las tiendas que tengan el mismo **type** y **store** y almacenelo en una variable llamada *store_types*.

```
[41] store_types = sales_subset.drop_duplicates(subset=['type', 'store'])
     store_types.head()
```

	Unnamed: 0	store	type	department	date	weekly_sales	is_holiday	temperature_c	fuel_price_usd_per_l	unemployment
0	0	1	A	1	2010-02-05	24924.50	False	5.727778	0.679451	8.106
901	901	2	A	1	2010-02-05	35034.06	False	4.550000	0.679451	8.324
1798	1798	4	A	1	2010-02-05	38724.42	False	6.533333	0.686319	8.623
2699	2699	6	A	1	2010-02-05	25619.00	False	4.683333	0.679451	7.259
3593	3593	10	B	1	2010-02-05	40212.84	False	12.411111	0.782478	9.765

Next steps: Generate code with store_types | View recommended plots | New interactive sheet

Cuente el número de tiendas de cada tipo

```
[42] store_types['type'].value_counts()
```

type	count
A	11
B	1

dtype: int64

Agrupamiento

Agrupe las ventas por *type* y obtenga la suma y el promedio de la columna *weekly_sales*.

```
[43] sales_subset.groupby('type')['weekly_sales'].agg(['sum', 'mean'])
```

type	sum	mean
A	2.337163e+08	23674.667242
B	2.317840e+07	25696.678370

Realice la misma operación pero utilizando tablas pivote.

```
[44] sales_subset.pivot_table(values='weekly_sales', index='type', aggfunc=['sum', 'mean'])
```

	sum	mean
	weekly_sales	weekly_sales
type		
A	2.337163e+08	23674.667242
B	2.317840e+07	25696.678370

Ahora obtenga el promedio de *weekly_sales* agrupando por *type* e *is_holiday*.

```
sales_subset.groupby(['type', 'is_holiday'])['weekly_sales'].mean()
```

		weekly_sales
type	is_holiday	
A	False	23768.583523
	True	590.045250
B	False	25751.980533
	True	810.705000

dtype: float64

Subconjuntos con LOC y ILOC

En esta parte se trabajará con el dataset *temperatures*, este dataset la temperatura promedio mensual en distintas ciudades del mundo.

Importe el dataset ***temperatures***

```
temperatures = pd.read_csv('https://raw.githubusercontent.com/sofiapapadron/IME_IAA/main/Laboratorio/Practica3_CONCEPTOS%20BASICOS%20DE%20PANDA/datasets/temperatures.csv?token=GHSAT0AAAAACWAYALRAWOOJ
temperatures.head()
```

	Unnamed: 0	date	city	country	avg_temp_c
0	0	2000-01-01	Abidjan	Côte D'Ivoire	27.293
1	1	2000-02-01	Abidjan	Côte D'Ivoire	27.685
2	2	2000-03-01	Abidjan	Côte D'Ivoire	29.061
3	3	2000-04-01	Abidjan	Côte D'Ivoire	28.162
4	4	2000-05-01	Abidjan	Côte D'Ivoire	27.547

Next steps: Generate code with temperatures | View recommended plots | New interactive sheet

Utilizando la función set_index(), asigne como índice la columna *date*, guarde el resultado en una variable llamada *temp_date*.

```
[48] temp_date = temperatures.set_index('date')
temp_date.head()
```

	Unnamed: 0	city	country	avg_temp_c
date				
2000-01-01	0	Abidjan	Côte D'Ivoire	27.293
2000-02-01	1	Abidjan	Côte D'Ivoire	27.685
2000-03-01	2	Abidjan	Côte D'Ivoire	29.061
2000-04-01	3	Abidjan	Côte D'Ivoire	28.162

Utilizando el método loc, obtenga todos los renglones del 2010 (es importante ordenar el indice de fecha antes de usar el método loc).

```
temp_date = temp_date.sort_index()
temp_date.loc['2010-01-01':'2010-12-31']
```

date	Unnamed: 0	city	country	avg_temp_c
2010-01-01	4905	Faisalabad	Pakistan	11.810
2010-01-01	10185	Melbourne	Australia	20.016
2010-01-01	3750	Chongqing	China	7.921
2010-01-01	13155	São Paulo	Brazil	23.738
2010-01-01	5400	Guangzhou	China	14.136
...	...	...	...	...
2010-12-01	6896	Jakarta	Indonesia	26.602
2010-12-01	5246	Gizeh	Egypt	16.530
2010-12-01	11186	Nagpur	India	19.120
2010-12-01	14981	Sydney	Australia	19.559
2010-12-01	13496	Salvador	Brazil	26.265

1200 rows × 4 columns

Ejercicio integrado

Utilizando el dataset *temperatures*, obtenga una nueva columna llamada *year* a partir de la columna *date*.

Almacene el resultado en una variable llamada *year_temp*

```
[56] temperatures['year']=temperatures['date'].str[:4]
     year_temp = temperatures
     year_temp.head()
```

	Unnamed: 0	date	city	country	avg_temp_c	year
0	0	2000-01-01	Abidjan	Côte D'Ivoire	27.293	2000
1	1	2000-02-01	Abidjan	Côte D'Ivoire	27.685	2000
2	2	2000-03-01	Abidjan	Côte D'Ivoire	29.061	2000
3	3	2000-04-01	Abidjan	Côte D'Ivoire	28.162	2000
4	4	2000-05-01	Abidjan	Côte D'Ivoire	27.547	2000

Next steps: Generate code with year_temp | View recommended plots | New interactive sheet

Asigne el año, el pais y la ciudad como tablas pivote en la variable year_temp para obtener la temperatura en grados Farenheit, el promedio, el valor minimo y maximo de temperaturas.

```
[60] year_temp['avg_temp_f'] = (year_temp['avg_temp_c'] * 9/5) + 32
     year_temp.head()
```

	Unnamed: 0	date	city	country	avg_temp_c	year	avg_temp_f
0	0	2000-01-01	Abidjan	Côte D'Ivoire	27.293	2000	81.1274
1	1	2000-02-01	Abidjan	Côte D'Ivoire	27.685	2000	81.8330
2	2	2000-03-01	Abidjan	Côte D'Ivoire	29.061	2000	84.3098
3	3	2000-04-01	Abidjan	Côte D'Ivoire	28.162	2000	82.6916
4	4	2000-05-01	Abidjan	Côte D'Ivoire	27.547	2000	81.5846

```
[61] year_temp.pivot_table(values='avg_temp_f', index=['year', 'country', 'city'], aggfunc=['mean', 'min', 'max'])
```

			mean	min	max
			avg_temp_f	avg_temp_f	avg_temp_f
year	country	city			
2000	Afghanistan	Kabul	60.48080	37.9868	78.9926
	Angola	Luanda	75.93860	70.0268	79.6928
	Australia	Melbourne	57.77615	47.3684	71.7710
		Sydney	63.62135	58.0172	69.2834
	Bangladesh	Dhaka	78.62945	65.8922	84.6554
...	...	...	...	...	...
2013	United States	Chicago	52.85640	31.0838	72.0140
		Los Angeles	64.61720	47.8634	77.1620
		New York	53.89500	29.5430	76.4996
	Vietnam	Ho Chi Minh City	83.21900	79.8746	85.4384
	Zimbabwe	Harare	67.56170	61.3382	72.9266

1400 rows × 3 columns

Comprensión

1. **¿Qué es una Serie de Pandas?**
   Una serie de Pandas es una estructura de datos similar a un arreglo que almacena datos de diferentes tipos y tiene un índice para identificar cada elemento.

2. **¿Qué es un DataFrame de Pandas y cómo se diferencia de una Serie?**
   Un dataframe de pandas una estructura de datos que es parecida a una tabla con etiquetas en las filas y columnas. La diferencia es que el dataframe puede tener diferentes series(columnas) permitiendo que se trabaje con datos mas complejos.

3. **Explique la diferencia entre los métodos loc y iloc de Pandas**
   Loc se usa para acceder a datos por etiquetas. Iloc se usa para acceder por posiciones.

4. **¿Qué es un índice en un DataFrame de Pandas y cuál es su propósito?**
   Es una etiqueta que identifica a cada fila de manera única y sirve para organizar y manipular los datos con mayor control.

5. **¿Cuál es la importancia de Pandas en el ámbito del análisis de datos y la ciencia de datos?**
   Pandas es muy importante porque permite la manipulación y análisis de datos de manera controlada y eficiente y facilita las tareas relacionadas al análisis de datos.

Conclusiones

En conclusión, Pandas es una biblioteca de Python muy importante para el ámbito de análisis de datos porque nos permite manejar estructuras de datos que facilitan el manejo y análisis de diversos tipos de datos.