

# Introducción a Pandas

Pandas es una librería de Python que proporciona herramientas para manipulación y análisis de datos de manera eficiente y fácil de usar. Es una de las librerías más populares en la comunidad de científicos de datos y es ampliamente utilizada para tareas como limpieza y preparación de datos, análisis exploratorio de datos, modelado y visualización.

La biblioteca de Pandas les ayuda a explotar los datos y visualmente ver la estructura de la salida mientras transforman sus datos. DataFrames de Pandas trabaja muy bien con las bibliotecas de aprendizaje automático en Python (scikit-learn, statsmodels) y las bibliotecas de visualización de datos (matplotlib, seaborn).

Las dos estructuras de datos principales en pandas son Series y DataFrame.

Una serie es un objeto similar a una matriz unidimensional que puede contener cualquier tipo de datos, como números enteros, números de coma flotante, cadenas y objetos de Python.

Un DataFrame es una estructura bidimensional similar a una tabla, donde cada columna puede tener un tipo de datos diferente.

Pandas proporciona funciones para leer y escribir datos en varios formatos de archivo, como CSV, Excel, bases de datos SQL y JSON. También incluye herramientas para la limpieza, transformación y agregación de datos, como filtrado, agrupación, fusión y rotación.

## **Aquí hay algunos de los métodos más utilizados en la librería pandas:**

1. **read\_csv():** Este método se utiliza para leer archivos CSV y crear un objeto DataFrame. El objeto DataFrame es una estructura de datos tabular que se utiliza para almacenar datos en filas y columnas. Puedes leer otros tipos de archivos como Excel, JSON, HTML, entre otros.
2. **head() y tail():** Estos métodos muestran las primeras o últimas filas de un objeto.

**Ejemplo:**

**df.head()**

Por defecto, muestran las primeras o últimas 5 filas, pero puedes especificar el número de filas que deseas mostrar.

**Ejemplo:**

**df.tail(3)**

Este código imprime las últimas 3 filas del documento, agregando la cantidad de filas que deseas mostrar.

3. **info()**: Este método muestra información sobre el objeto DataFrame, incluyendo el tipo de datos de cada columna y el número de valores no nulos.

**Ejemplo:**

**df.info()**

```
: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                  Non-Null Count  Dtype  
---  --
 0   Booking_ID                             36275 non-null  object 
 1   no_of_adults                           36275 non-null  int64  
 2   no_of_children                          36275 non-null  int64  
 3   no_of_weekend_nights                    36275 non-null  int64  
 4   no_of_week_nights                       36275 non-null  int64  
 5   type_of_meal_plan                       36275 non-null  object 
 6   required_car_parking_space              36275 non-null  int64  
 7   room_type_reserved                      36275 non-null  object 
 8   lead_time                              36275 non-null  int64  
 9   arrival_year                           36275 non-null  int64  
10  arrival_month                           36275 non-null  int64  
11  arrival_date                            36275 non-null  int64  
12  market_segment_type                     36275 non-null  object 
13  repeated_guest                          36275 non-null  int64  
14  no_of_previous_cancellations            36275 non-null  int64  
15  no_of_previous_bookings_not_canceled    36275 non-null  int64  
16  avg_price_per_room                      36275 non-null  float64 
17  no_of_special_requests                  36275 non-null  int64  
18  booking_status                          36275 non-null  object 
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

#### 4. Comprobar los tipos de datos de las columnas

Queremos ver los tipos de datos de las columnas para ver con qué tipo de datos estamos trabajando. Una vez que empecemos a construir modelos, necesitamos que todos

nuestros datos sean de algún un tipo de columna numérica (int o flotante), así que presten mucha atención a cualquier columna del tipo “objeto”.

**Ejemplo:**

```
df.dtypes
```

```
RM float64
```

```
LSTAT float64
```

```
PTRATIO float64
```

```
target float64
```

```
dtype: object
```

5. **shape:** Este método permite visualizar cuantas filas y columnas contiene el documento.

**Ejemplo:**

```
df.shape
```

```
(500, 4)
```

Este código imprimió la cantidad de filas en este caso 500 y 4 columnas.

6. **describe():** Este método calcula estadísticas descriptivas para cada columna del objeto DataFrame, incluyendo la media, la desviación estándar, el mínimo, el máximo, y los cuartiles.

**Ejemplo:**

```
df.describe()
```

```
df.describe()
```

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	required_car_parking_space	lead_time	arrival_year	arrival_month	arrival_
count	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.000000	36275.00
mean	1.844962	0.105279	0.810724	2.204300	0.030986	85.232557	2017.820427	7.423653	15.59
std	0.518715	0.402648	0.870644	1.410905	0.173281	85.930817	0.383836	3.069894	8.74
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2017.000000	1.000000	1.00
25%	2.000000	0.000000	0.000000	1.000000	0.000000	17.000000	2018.000000	5.000000	8.00
50%	2.000000	0.000000	1.000000	2.000000	0.000000	57.000000	2018.000000	8.000000	16.00
75%	2.000000	0.000000	2.000000	3.000000	0.000000	126.000000	2018.000000	10.000000	23.00
max	4.000000	10.000000	7.000000	17.000000	1.000000	443.000000	2018.000000	12.000000	31.00

7. **loc[] y iloc[]**: Estos métodos se utilizan para acceder a datos en un objeto DataFrame.

**loc[]** se utiliza para acceder a filas y columnas por etiquetas, mientras que **iloc[]** se utiliza para acceder a filas y columnas por índice numérico.

En Pandas, **loc** es una función utilizada para seleccionar filas y columnas de un **DataFrame** por etiquetas o nombres de índices.

La sintaxis básica de **loc** es:

**df.loc[filas, columnas]**

**Ejemplo:**

**df.loc[3,:]**

```
df.loc[3,:]
Booking_ID                INN00004
no_of_adults                2
no_of_children              0
no_of_weekend_nights        0
no_of_week_nights           2
type_of_meal_plan           Meal Plan 1
required_car_parking_space  0
room_type_reserved          Room_Type 1
lead_time                  211
arrival_year                2018
arrival_month                5
arrival_date                 20
market_segment_type         Online
repeated_guest              0
no_of_previous_cancellations 0
no_of_previous_bookings_not_canceled 0
avg_price_per_room          100.0
no_of_special_requests       0
booking_status              1
Name: 3, dtype: object
```

**Este código imprimió todas las columnas que contiene la fila 3.**

En Pandas, **iloc** es una función utilizada para seleccionar filas y columnas de un **DataFrame** por índices enteros.

La sintaxis básica de **iloc** es:

**df.iloc[filas, columnas]**

Donde **filas** y **columnas** son índices enteros o listas de índices enteros que representan las filas y columnas a seleccionar. Si se omite **columnas**, se seleccionarán todas las columnas.

```
df.iloc[3,5]
```

```
'Meal Plan 1'
```

Este código solo imprimió el valor o contenido de la fila tres de la columna 5.

```
In [167]: df.iloc[0:2, 1:3]
```

```
Out[167]:
```

	no_of_adults	no_of_children
0	2	0
1	2	0

Este código selecciona las primeras dos filas y la segunda y tercera columnas.

8. **Cortes**, a menudo vamos a querer acceder solo a ciertas partes de nuestro conjunto de datos, ya sea si es solo una columna, unas pocas columnas o una porción de nuestro conjunto de datos en base a alguna condición.

#### Seleccionar columnas usando corchetes

Con los corchetes pueden seleccionar una o más columnas. Los corchetes simples devolverán una serie Pandas, mientras que los corchetes dobles devolverán un dataframe de Pandas.

# Seleccionar una columna usando doble corchetes.

```
df[['nombre_columna']]
```

```
df[['type_of_meal_plan']].head()
```

```
In [174]: df[['type_of_meal_plan']].head()
```

```
Out[174]:
```

	type_of_meal_plan
0	Meal Plan 1
1	Not Selected
2	Meal Plan 1
3	Meal Plan 1
4	Not Selected

Este código imprime solo los primeros valores de la columna de nombre 'type\_of\_meal\_plan'.

9. **drop():** Este método se utiliza para eliminar filas o columnas de un objeto DataFrame.

**Ejemplo:**

```
df.drop("type_of_meal_plan")
```

Este código elimina la columna 'type\_of\_meal\_plan'.

10. **groupby():** Este método se utiliza para agrupar filas de un objeto DataFrame por una o varias columnas y aplicar una función a cada grupo.

En Pandas, groupby es una función utilizada para agrupar datos en un DataFrame según una o varias columnas y aplicar una función de agregación a cada grupo.

La sintaxis básica de groupby es:

```
df.groupby(columnas).funcion_agregacion()
```

Donde columnas son las columnas por las que se desea agrupar los datos y funcion\_agregacion es la función que se aplicará a cada grupo. Algunas funciones de agregación comunes incluyen sum, mean, max, min, count, std, var, entre otras. Algunos ejemplos de cómo se puede utilizar groupby en Pandas son los siguientes:

**OJO:** Las funciones de agregación las variables deben ser numéricas y se colocan en la última parte del código.

**Ejemplo:**

```
df.groupby('arrival_year')['lead_time'].mean()

Out[177]: arrival_year
2017      58.886859
2018      90.999026
Name: lead_time, dtype: float64
```

Este código imprime la variable "arrival\_year" con el promedio del "lead time".

## Usar GroupBy con múltiples columnas

```
df.groupby(['arrival_year', 'room_type_reserved'])['lead_time'].max()
```

arrival_year	room_type_reserved	
2017	Room_Type 1	327
	Room_Type 2	296
	Room_Type 3	66
	Room_Type 4	221
	Room_Type 5	96
	Room_Type 6	66
	Room_Type 7	110
2018	Room_Type 1	443
	Room_Type 2	381
	Room_Type 3	180
	Room_Type 4	355
	Room_Type 5	297
	Room_Type 6	346
	Room_Type 7	213

Este código imprimió las agrupaciones de las variables “arrival\_year”, “room\_type\_reserved” con los valores max cumplidos en esas dos condiciones de la variable “lead\_time”.

OJO: Fijarse en el cambio de los corchetes cuando se hacen agrupaciones con multivariabes.

El dataset utilizado previamente es: <https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset>

## Ejercicio Practico

### Operaciones básicas

El objetivo de este ejercicio es:

1. Asegurar que hayan cargado correctamente los datos.
2. Ver qué tipo de datos tienen.
3. Comprobar la validez de los datos

1. Descargar la data del link:

[https://drive.google.com/file/d/1X6570Wm0vp8EkM\\_Civf3jN\\_zHJnHNxSw/view](https://drive.google.com/file/d/1X6570Wm0vp8EkM_Civf3jN_zHJnHNxSw/view)

Subir la data con el metodo: `df = pd.read_csv('nombre_archivo')`.

2. Imprimir la data descriptiva de los datos y comentar que datos contienen y cuales columnas.
3. Con el método `iloc` buscar las 6 primeras filas de las 3 primeras columnas.
4. Con el método `Groupby` encontrar la media de la variable "Seat Comfort" agrupados por la variable "Class"
5. Con el mismo ejercicio pasado agrega la variable "Gender" a la agrupación, explique qué resultados pudo obtener? .