Universidad Católica

San Pablo

Pandas

Curso

Graciela Meza Lovon, Yessenia Yari Ramos, Alvaro Mamani Aliaga

Contenido

- Qué es Pandas?
- Series
- Archivos CSV
- Análisis Básico de los datos
- Consultas Básicas en un dataframe

¿Qué es Pandas?

- Python Data Analysis Library (alias Pandas).
- Pandas es una biblioteca de software escrita como extensión de Numpy para manipulación y análisis de datos para el lenguaje de programación Python.
- Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

¿Qué es Pandas?

- Las características de la biblioteca son:
  - El tipo de datos son DataFrame para manipulación de datos con indexación integrada. Tiene herramientas para leer y escribir datos entre estructuras de dato en memoria y formatos de archivos variados

¿Qué es Pandas?

- Las características de la biblioteca son:
  - Permite la alineación de dato y manejo integrado de datos faltantes, la reestructuración y segmentación de conjuntos de datos, la segmentación vertical basada en etiquetas, indexación elegante, y segmentación horizontal de grandes conjuntos de datos, la inserción y eliminación de columnas en estructuras de datos.

¿Qué es Pandas?

- Las características de la biblioteca son:
  - Permite realizar cadenas de operaciones, dividir, aplicar y combinar sobre conjuntos de datos, la mezcla y unión de datos.

¿Qué es Pandas?

- Las características de la biblioteca son:
  - Permite realizar indexación jerárquica de ejes para trabajar con datos de altas dimensiones en estructuras de datos de menor dimensión, la funcionalidad de series de tiempo: generación de rangos de fechas y conversión de frecuencias, desplazamiento de ventanas estadísticas y de regresiones lineales, desplazamiento de fechas y retrasos.

Series

- La serie es una de las estructuras de datos centrales en pandas.
- Puede pensar en ello como un cruce entre una lista y un diccionario.
  - Todos los elementos se almacenan en orden y hay etiquetas con las que puede recuperarlos.
- Una forma fácil de visualizar esto son dos columnas de datos.
  - El primero es el índice especial, muy parecido a las claves de un diccionario.
  - El segundo son los datos reales.

Creación

- Importar la librería

```
import pandas as pd
```

Creación

- Una serie se puede crear a partir de una lista de la forma siguiente:

```
import pandas as pd
estudiantes = ["Graciela", "Luisa", "Jorge"]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

Creación

- Una serie se puede crear a partir de una lista de la forma siguiente:

```
import pandas as pd
estudiantes = ["Graciela", "Luisa", "Jorge"]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
0    Graciela
1       Luisa
2       Jorge
dtype: object
```

Creación

- Una serie se puede crear a partir de una lista de la forma siguiente:

```
import pandas as pd

num = [8, 10, 12]
s = pd.Series(num)
print(type(s))
print(s)
```

Creación

- Una serie se puede crear a partir de una lista de la forma siguiente:

```
import pandas as pd

num = [8, 10, 12]
s = pd.Series(num)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
0     8
1    10
2    12
dtype: int64
```

Creación

- Una serie puede manipular datos con información faltante:

```
import pandas as pd
estudiantes = ["Graciela", "Luisa", None]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

Creación

- Una serie puede manipular datos con información faltante:

```
import pandas as pd
estudiantes = ["Graciela", "Luisa", None]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
0    Graciela
1       Luisa
2        None
dtype: object
```

Creación

- Una serie puede manipular datos con información faltante:

```
import pandas as pd

num = [8, 10, None]
s = pd.Series(num)
print(type(s))
print(s)
```

Creación

- Una serie se puede puede manipular datos con información faltante:

```
import pandas as pd

num = [8, 10, None]
s = pd.Series(num)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
0     8.0
1    10.0
2     NaN
dtype: float64
```

Creación

- Una serie se puede crear a partir de un diccionario de la forma siguiente:

```
import pandas as pd
estudiantes = {"clave1":"Graciela",
"clave2":"Gonzalo", "clave3":"Laura" }
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

Creación

- Una serie se puede crear a partir de un diccionario de la forma siguiente:

```
import pandas as pd
estudiantes = {"clave1":"Graciela",
"clave2":"Gonzalo", "clave3":"Laura" }
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
clave1    Graciela
clave2     Gonzalo
clave3       Laura
dtype: object
```

Creación

- Una serie se puede crear a partir de un lista de tuplas de la forma siguiente:

```
import pandas as pd
estudiantes = [("Graciela", "Meza"),("Laura2",
"Flores"),("Patricio", "Mitchel")]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

Creación

- Una serie se puede crear a partir de un lista de tuplas de la forma siguiente:

```
import pandas as pd
estudiantes = [("Graciela", "Meza"),("Laura2",
"Flores"),("Patricio", "Mitchel")]
s = pd.Series(estudiantes)
print(type(s))
print(s)
```

```
Salida:
<class 'pandas.core.series.Series'>
0        (Graciela, Meza)
1        (Laura2, Flores)
2    (Patricio, Mitchel)
dtype: object
```

Importando un CSV en Python

- La adquisición de datos es un proceso de carga y lectura de datos desde varias fuentes.
- Para leer cualquier dato usando el paquete pandas de Python, hay dos factores importantes a considerar:
  - El formato
  - La ruta del archivo.

Importando un CSV en Python

- La adquisición de datos es un proceso de carga y lectura de datos desde varias fuentes.
- Para leer cualquier dato usando el paquete pandas de Python, hay dos factores importantes a considerar:
  - El formato: El formato es la forma en que se codifican los datos.
    - Diferentes esquemas de codificación observando el final del nombre del archivo.
    - Algunas codificaciones habituales son: CSV, JSON, XLSX, HDF, etc.

Importando un CSV en Python

- La adquisición de datos es un proceso de carga y lectura de datos desde varias fuentes.
- Para leer cualquier dato usando el paquete pandas de Python, hay dos factores importantes a considerar:
  - La ruta: La ruta nos dice dónde se almacenan los datos.
    - Computadora.
    - Dirección web.

Importando un CSV en Python

- Usando el método read_csv

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
```

Importando un CSV en Python

- Usando el método read_csv

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
```

- Pandas asume que CSV tiene una cabecera

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv",header = None)
```

Imprimiendo el Dataframe en Python

- Usando el método head()

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df.head()
```

Imprimiendo el Dataframe en Python

- Usando el método head()

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	...	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible		2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible		2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback		3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan		3.4	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan		3.4	8.0	115	5500	18	22	17450

Imprimiendo el Dataframe en Python

- Usando el método tail()

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df.tail()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	2952	ohc	four	141	mpfi	3.78	3.15
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.8	55.5	3049	ohc	four	141	mpfi	3.78	3.15
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3012	ohcv	six	173	mpfi	3.58	2.87
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3217	ohc	six	145	idi	3.01	3.4
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3062	ohc	four	141	mpfi	3.78	3.15

Imprimiendo el Dataframe en Python

- Usando el método tail()

```
import pandas as pd
df = pd.read_csv("Automobile_data.csv")
df.tail()
```

Reemplazando la cabecera

- Sobreescribiendo df.columns

```
n_cabecera = ['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
'num-of-doors', 'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length',
'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders', 'engine-size',
'fuel-system', 'bore', 'stroke', 'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
'highway-mpg', 'precio']
df.columns = n_cabecera
```

Reemplazando la cabecera

- Sobreescribiendo df.columns

```
n_cabecera = ['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
'num-of-doors', 'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length',
'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders', 'engine-size',
'fuel-system', 'bore', 'stroke', 'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
'highway-mpg', 'precio']
df.columns = n_cabecera
```

compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	precio
9.0	111	5000	21	27	13495
9.0	111	5000	21	27	16500
9.0	154	5000	19	26	16500
10.0	102	5500	24	30	13950
8.0	115	5500	18	22	17450

Exportando un dataframe to CSV

- Se puede guardar los cambios en un archivo csv

```
ruta=”./Autos_modificado.csv”
df.to_csv(ruta)
```

Otros formatos en Python

❖

Data Format	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
sql	pd.read_sql()	df.to_sql()

Análisis básico de los Datos

- Entender los datos antes de empezar con el análisis
- Verificar:
  - Tipos de Datos
  - Distribución de los Datos
- Identificar problemas con los datos

Análisis básico de los Datos

- Tipos de Datos

Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	time data.

Análisis básico de los Datos

- Tipos de Datos: usando df.dtypes

```
df.dtypes
```

Análisis básico de los Datos

- Tipos de Datos: usando df.dtypes

```
df.dtypes
```

```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
precio                object
dtype: object
```

Análisis básico de los Datos

- Resumen Estadístico : df.describe()

```
df.describe()
```

Análisis básico de los Datos

- Resumen Estadístico : df.describe()

```
df.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Análisis básico de los Datos

- Resumen Estadístico Completo (include =”all”)

```
df.describe(include = "all")
```

Análisis básico de los Datos

- Resumen Estadístico Completo (include =”all”)

```
df.describe(include = "all")
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	e
count	205.000000	205	205	205	205	205	205	205	205	205.000000	205.000000	205.000000	205.000000	205.000000	
unique	NaN	52	22	2	2	3	5	3	2	NaN	NaN	NaN	NaN	NaN	
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	NaN	NaN	NaN	NaN	
freq	NaN	41	32	185	168	114	96	120	202	NaN	NaN	NaN	NaN	NaN	
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	174.049268	65.907805	53.724878	2555.565854	
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	12.337289	2.145204	2.443522	520.680204	
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	141.100000	60.300000	47.800000	1488.000000	
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	166.300000	64.100000	52.000000	2145.000000	
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	173.200000	65.500000	54.100000	2414.000000	
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	183.100000	66.900000	55.500000	2935.000000	
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	208.100000	72.300000	59.800000	4066.000000	

Consultas Básicas en un Dataframe

- Visualizar alguna columna en especial

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

```
0    3
1    3
2    1
3    2
4    2
Name: symboling, dtype: int64
```

Consultas Básicas en un Dataframe

- Visualizar varias columnas

```
df[['symboling', 'price', 'width']].head()
```

	symboling	price	width
0	3	13495	64.1
1	3	16500	64.1
2	1	16500	65.5
3	2	13950	66.2
4	2	17450	66.4

Consultas Básicas en un Dataframe

- Visualizar alguna fila en especial

```
df.iloc[0, :]
```

Consultas Básicas en un Dataframe

- Visualizar alguna fila en especial

```
df.iloc[0, :]
```

```
symboling                3
normalized-losses        ?
make                 alfa-romero
fuel-type                gas
aspiration               std
num-of-doors             two
body-style           convertible
drive-wheels             rwd
engine-location        front
wheel-base              88.6
length                 168.8
width                   64.1
height                  48.8
curb-weight             2548
engine-type             dohc
num-of-cylinders        four
engine-size              130
fuel-system             mpfi
bore                    3.47
stroke                  2.68
compression-ratio          9
horsepower               111
peak-rpm                5000
city-mpg                  21
highway-mpg               27
price                  13495
Name: 0, dtype: object
```

Consultas Básicas en un Dataframe

- Visualizar algunas filas

```
df.iloc[0:4, :]
```

Consultas Básicas en un Dataframe

- Visualizar algunas filas

```
df.iloc[0:4, :]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	...	city-mpg	highway-mpg		price
0	3	?	alfa-romero	gas	std	two		21	27	0	13495
1	3	?	alfa-romero	gas	std	two		21	27	1	16500
2	1	?	alfa-romero	gas	std	two		19	26	2	16500
3	2	164	audi	gas	std	four		24	30	3	13950

Consultas Básicas en un Dataframe

- Usando máscaras booleanas

```
mask = df['width']>70
df[mask]
```

Consultas Básicas en un Dataframe

- Usando máscaras booleanas

```
mask = df['width']>70
df[mask]
```

	symboling	normalized-losses	make	fuel-type	aspiration	...	length	width	highway-mpg	price
6	1	158	audi	gas	std		192.7	71.4	25	17710
7	1	?	audi	gas	std		192.7	71.4	25	18920
8	1	158	audi	gas	turbo		192.7	71.4	20	23875
17	0	?	bmw	gas	std		197.0	70.9	20	36880
49	0	?	jaguar	gas	std		191.7	70.6	17	36000
67	-1	93	mercedes-benz	diesel	turbo		190.9	70.3	25	25552
68	-1	93	mercedes-benz	diesel	turbo		190.9	70.3	25	28248
69	0	93	mercedes-benz	diesel	turbo		187.5	70.3	25	28176
70	-1	93	mercedes-benz	diesel	turbo		202.6	71.7	25	31600
71	-1	?	mercedes-benz	gas	std		202.6	71.7	18	34184
72	3	142	mercedes-benz	gas	std		180.3	70.5	18	35056
73	0	?	mercedes-benz	gas	std		208.1	71.7	16	40960
74	1	?	mercedes-benz	gas	std		199.2	72.0	16	45400
129	1	?	porsche	gas	std		175.7	72.3	28	?

Consultas Básicas en un Dataframe

- Usando máscaras booleanas

```
mask = (df['width']>70) & (df['width']<71)
df[mask]
```

Consultas Básicas en un Dataframe

- Usando máscaras booleanas

```
mask = (df['width']>70) & (df['width']<71)
df[mask]
```

	symboling	normalized-losses	make	fuel-type	aspiration	...	width	height	curb-weight	engine-type	price
17	0	?	bmw	gas	std		70.9	56.3	3505	ohc	36880
49	0	?	jaguar	gas	std		70.6	47.8	3950	ohcv	36000
67	-1	93	mercedes-benz	diesel	turbo		70.3	56.5	3515	ohc	25552
68	-1	93	mercedes-benz	diesel	turbo		70.3	58.7	3750	ohc	28248
69	0	93	mercedes-benz	diesel	turbo		70.3	54.9	3495	ohc	28176
72	3	142	mercedes-benz	gas	std		70.5	50.8	3685	ohcv	35056

¡Gracias!