Pandas práctica

Esta sección proporciona una lista de todos los trucos y consejos.

La **media** (promedio) de un conjunto de datos se encuentra al sumar todos los números en el conjunto de datos y luego al dividir entre el número de valores en el conjunto.

La **mediana** es el valor medio cuando un conjunto de datos se ordena de menor a mayor.

1. Crear una nueva columna a partir de múltiples columnas en tu DataFrame puede ser bastante sencillo. Sin embargo, ¿qué sucede si deseas implementar una función más compleja y utilizarla como lógica para crear la columna? Aquí es donde las cosas pueden volverse un poco desafiantes.

¡Adivina qué...

Usando los métodos `apply` y `lambda`, puedes aplicar fácilmente cualquier lógica a tus columnas siguiendo el siguiente formato:

```
df[new_col] = df.apply(lambda row: func(row), axis=1)
```

donde:

➡️ `df` es tu DataFrame.

➡️ `row` corresponderá a cada fila en tu DataFrame.

➡️ `func` es la función que deseas aplicar a tu DataFrame.

➡️ `axis=1` para aplicar la función a cada fila en tu DataFrame.

💡 A continuación se muestra una ilustración.

```
import pandas as pd

#Creamos el data frame

candidates= {
'Name':["Aida","Mamadou","Ismael","Aicha","Fatou", "Khalil"],
'Degree':['Master','Master','Bachelor', "PhD", "Master", "PhD"],
'From':["Abidjan","Dakar","Bamako", "Abidjan","Konakry", "Lomé"],
'Years_exp': [2, 3, 0, 5, 4, 3],
'From_office(min)': [120, 95, 75, 80, 100, 34]
}
candidates_df = pd.DataFrame(candidates)

"""
----------------Mi funcion personalizada-------------------
"""
def candidate_info(row):

#seleccionamos las columnas de interes:

    name =  row.Name
    is_from = row.From
    year_exp = row.Years_exp
    degree = row.Degree
    from_office = row["From_office(min)"]

    #Generamos la descripcion de variables previas:

    info = f"""{name} from {is_from} holds a {degree} degree
    with {year_exp} year(s) experience
    and lives {from_office} from the office"""

    return info

"""
-------Aplicamos la funcion a los datos------
"""
candidates_df["Description"] = candidates_df.apply(lambda row: candidate_info(row), axis=1)
```

La función `candidate_info` combina la información de cada candidato para crear una columna de descripción única sobre ese candidato.

index	Name	Degree	From	Years_exp	From_office(min)	Description
0	Aida	Master	Abidjan	2	120	Aida from Abidjan holds a Master degree with 2 year(s) experience and lives 120 from the office
1	Mamadou	Master	Dakar	3	95	Mamadou from Dakar holds a Master degree with 3 year(s) experience and lives 95 from the office
2	Ismael	Bachelor	Bamako	0	75	Ismael from Bamako holds a Bachelor degree with 0 year(s) experience and lives 75 from the office
3	Aicha	PhD	Abidjan	5	80	Aicha from Abidjan holds a PhD degree with 5 year(s) experience and lives 80 from the office
4	Fatou	Master	Konakry	4	100	Fatou from Konakry holds a Master degree with 4 year(s) experience and lives 100 from the office
5	Khalil	PhD	Lomé	3	34	Khalil from Lomé holds a PhD degree with 3 year(s) experience and lives 34 from the office

2. Convertir datos categóricos en numéricos
   Este proceso generalmente ocurre en la fase del análisis exploratorio de datos. Algunos de sus beneficios son:

- La identificación de valores atípicos, valores inválidos y valores faltantes en los datos.
- La reducción de la posibilidad de sobreajuste mediante la creación de modelos más robustos.

➡️ Utiliza estas dos funciones de Pandas, dependiendo de tus necesidades. Se proporcionan ejemplos en la imagen a continuación.

1️⃣ `.cut()` para definir específicamente los límites de tus categorías.

Ejemplo:

Supongamos que deseas categorizar a los candidatos según su experiencia en función del número de años, donde:

- Nivel de entrada: 0-1 año.
- Nivel intermedio: 2-3 años.
- Nivel senior: 4-5 años.

```
seniority = ['Entry level', 'Mid level', 'Senior level']
seniority_bins = [0, 1, 3, 5]
candidates_df['Seniority'] = pd.cut(candidates_df['Years_exp'],
bins=seniority_bins,
labels=seniority,
include_lowest=True
candidates_df
```

	Name	Degree	From	Years_exp	From_office(min)	Seniority
0	Aida	Master	Abidjan	2	120	Mid level
1	Mamadou	Master	Dakar	3	95	Mid level
2	Ismael	Bachelor	Bamako	0	75	Entry level
3	Aicha	PhD	Abidjan	5	80	Senior level
4	Fatou	Master	Konakry	4	100	Senior level
5	Khalil	PhD	Lomé	3	34	Mid level

2️⃣ `.qcut()` para dividir tus datos en categorías de tamaño igual.
Esta función utiliza los percentiles subyacentes de la distribución de los datos, en lugar de los límites de las categorías.

Ejemplo:

Supongamos que deseas categorizar el tiempo de viaje al trabajo de los candidatos en "bueno", "aceptable" o "demasiado largo". Puedes utilizar `.qcut()` para dividir los datos en categorías de tamaño igual basadas en los percentiles de la distribución del tiempo

de viaje.

```
commute_time_labels = ["good", "acceptable", "too long"]
candidates_df["Commute_level"] = pd.qcut(
candidates_df["From_office(min)"],
q = 3,
labels=commute_time_labels
)
candidates_df
```

	Name	Degree	From	Years_exp	From_office(min)	Seniority	Commute_level
0	Aida	Master	Abidjan	2	120	Mid level	too long
1	Mamadou	Master	Dakar	3	95	Mid level	acceptable
2	Ismael	Bachelor	Bamako	0	75	Entry level	good
3	Aicha	PhD	Abidjan	5	80	Senior level	acceptable
4	Fatou	Master	Konakry	4	100	Senior level	too long
5	Khalil	PhD	Lomé	3	34	Mid level	good

Cuando usas `.cut()`: el número de bins es igual al número de etiquetas + 1.
Cuando usas `.qcut()`: el número de bins es igual al número de etiquetas.
Con `.cut()`: establece `include_lowest=True`, de lo contrario, el valor más bajo se convertirá en NaN.

3. Seleccionar filas de un DataFrame de Pandas basado en los valores de columna(s)
   ➡️ Utiliza la función `.query()` especificando la condición de filtro.
   ➡️ La expresión de filtro puede contener cualquier operador (<, >, ==, !=, etc.).
   ➡️ Usa el símbolo `@` para usar una variable en la expresión.

```
#Consigue todos los candidatos con título de Máster
ms_candidates = candidates_df.query("Degree == 'Master'")
#Obtener candidatos que no sean de licenciatura
no_bs_candidates = candidates_df.query("Degree != 'Bachelor'")
#Obtener valores de la lista
list_locations = ["Abidjan", "Dakar"]
candiates = candidates_df.query("From in @list_locations")
```

4. Trabajar con archivos comprimidos en formato zip
   A veces puede ser eficiente leer y escribir archivos .zip sin extraerlos de tu disco local. A continuación se muestra una

ilustración.

```
import pandas as pd
"""
------------ READ ZIP FILES -----------
"""
#Case 1: read a single zip file
candidate_df_unzip = pd.read_csv('candidates.csv.zip', compression='zip')
#Case 2: read a file from a folder
from zipfile import ZipFile
#Read the file from a zip folder
sales_df = pd.read_csv(ZipFile("data.zip").open('data/sales_df.csv'))
"""
------------ WRITE ZIP FILES -----------
"""
#Read data from internet
url = " https://raw.githubusercontent.com/keitazoumana/Fastapi-tutorial/master/data/spam.csv "
spam_data = pd.read_csv(url, encoding="ISO-8859-1")
#Save it as a zip file
spam_data.to_csv("spam.csv.zip", compression="zip")
#Check the files sizes
from os import path
path.getsize('spam.csv') / path.getsize('spam.csv.zip')
```

Ir a colab ubicado en:
https://colab.research.google.com/drive/1QHSUx6uh4Ujp6uUech_BEi_gjuU0WNu4#

El archivo excel de prueba con los datos esta en:
https://drive.google.com/drive/folders/1LqE8g4Gk2QMKp2naYmXWbie9t2JejVXy?usp=sharing