Pandas

March 18, 2020

1 Patalagua Suárez Nicolás

1.1 Universidad Sergio Arboleda

1.1.1 Programación cientifica para I.A

1.1.2 Marzo 12, 2020

2 Pandas

En Computación y Ciencia de datos, pandas es una biblioteca de software escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

3 Numpy

NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

[0]:
```
import pandas as pd #Importamos Panda y la asignamos a la variable pd
import numpy as np #Importamos numpy y la asignamos a la variable np
```

[0]:
```
#pd.set_option('max_columns',n,'max_rows',m)
movies=pd.read_csv('movie.csv')#Leemos el archivo  método read_csv de pandas.
movies.head()#Vemos la cabecera adjudicada a movies
```

[0]:

	color	director_name	…	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	…	1.78	33000
1	Color	Gore Verbinski	…	2.35	0
2	Color	Sam Mendes	…	2.35	85000
3	Color	Christopher Nolan	…	2.35	164000
4	NaN	Doug Walker	…	NaN	0

[5 rows x 28 columns]

[0]:
```
#Accediendo a los componentes principales del DataFrame (Atributos)

columns=movies.columns # Todos los objetos columnas son asignados a la variable columnas
index=movies.index # Todos los objetos index son asignados a la variable index
data=movies.values # Todos los objetos valor son asignados a la variable data

columns #Muestra el contenido de la variable columns
index #Muestra el contenido de la variable index
data #Muestra el contenido de la variable data
```

[0]:
```
array([['Color', 'James Cameron', 723.0, …, 7.9, 1.78, 33000],
       ['Color', 'Gore Verbinski', 302.0, …, 7.1, 2.35, 0],
       ['Color', 'Sam Mendes', 602.0, …, 6.8, 2.35, 85000],
       …,
       ['Color', 'Benjamin Roberds', 13.0, …, 6.3, nan, 16],
       ['Color', 'Daniel Hsia', 14.0, …, 6.3, 2.35, 660],
       ['Color', 'Jon Gunn', 43.0, …, 6.6, 1.85, 456]], dtype=object)
```

[0]:
```
#Tipos de datos de cada variable
type(index)
type(columns)
type(data)
```

[0]:
```
numpy.ndarray
```

[0]:
```
#valores
index.values #Valores del index
columns.values #Columnas del dataframe
```

[0]:
```
array(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes',
       'actor_2_name', 'actor_1_facebook_likes', 'gross', 'genres',
       'actor_1_name', 'movie_title', 'num_voted_users',
       'cast_total_facebook_likes', 'actor_3_name',
       'facenumber_in_poster', 'plot_keywords', 'movie_imdb_link',
       'num_user_for_reviews', 'language', 'country', 'content_rating',
       'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score',
       'aspect_ratio', 'movie_facebook_likes'], dtype=object)
```

[0]:
```
movies=pd.read_csv('movie.csv') #Leemos el archivo.
movies.dtypes #Vemos el tipo de datos
movies.get_dtype_counts() #El método get_dtype_counts se puede usar para ver un desglose de los tipos.
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: FutureWarning:
`get_dtype_counts` has been deprecated and will be removed in a future version.
```

```
For DataFrames use `.dtypes.value_counts()
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[0]:
```
float64    13
int64       3
object     12
dtype: int64
```

[0]:
```
#Columna de datos como una serie
movies['director_name']
movies.director_name
```

[0]:
```
0           James Cameron
1          Gore Verbinski
2              Sam Mendes
3       Christopher Nolan
4             Doug Walker
               …
4911          Scott Smith
4912                  NaN
4913     Benjamin Roberds
4914          Daniel Hsia
4915             Jon Gunn
Name: director_name, Length: 4916, dtype: object
```

[0]:
```
type(movies['director_name']) #Tipo de dato de la columna director_name
```

[0]:
```
pandas.core.series.Series
```

[0]:
```
#Series como variables
director=movies['director_name'] #Guardamos la serie director_name en la␣
↪variable director
director.name #Nombre de la columna
director.to_frame().head()#Convierte a director en un frame, con la cabecera␣
↪del mismo
```

[0]:
```
       director_name
0      James Cameron
1     Gore Verbinski
2         Sam Mendes
3  Christopher Nolan
4        Doug Walker
```

[0]:
```
#Series y métodos: llamadas
s_attr_methods=set(dir(pd.Series)) #Todos los atributos y metodos de la serie␣
↪en un conjunto
```

```
len(s_attr_methods) #Cuenta de los atributos y metodos de con funcion DIR
```

[0]: 465

```
df_attr_methods=set(dir(pd.DataFrame)) #Todos los atributo y metodos del␣
 ↪dataFrame en un conjunto
len(df_attr_methods) #CCuenta de los atributos y metodos con la funcion DIR
```

[0]: 451

```
len(s_attr_methods & df_attr_methods) #Realiza la intercepción de los metodos␣
 ↪de la serie y de los metodos del dataframe
```

```
director=movies['director_name'] #Contiene strings
actor_1_fb_likes=movies['actor_1_facebook_likes'] #Contiene numeros
```

```
director.head() #Envia la cabecera de la columna director
```

[0]:
```
0        James Cameron
1       Gore Verbinski
2           Sam Mendes
3    Christopher Nolan
4          Doug Walker
Name: director_name, dtype: object
```

```
actor_1_fb_likes.head() #Envia la cabecera de la columna actor_1_fb_likes
```

[0]:
```
0     1000.0
1    40000.0
2    11000.0
3    27000.0
4      131.0
Name: actor_1_facebook_likes, dtype: float64
```

```
pd.set_option('max_rows',8) #Envia un maximo de 8 impresiones
director.value_counts() #Conteo de los valores que se repiten
```

[0]:
```
Steven Spielberg    26
Woody Allen         22
Martin Scorsese     20
Clint Eastwood      20
                    ..
Dany Boon            1
Conor McPherson      1
Bradley Parker       1
Sol Tryon            1
Name: director_name, Length: 2397, dtype: int64
```

[0]:
```
actor_1_fb_likes.value_counts() #Imprime los valores sin repetición y muestra␣
↪la cantidad de likes
```

[0]:
```
1000.0     436
11000.0    206
2000.0     189
3000.0     150
           …
216.0        1
859.0        1
225.0        1
334.0        1
Name: actor_1_facebook_likes, Length: 877, dtype: int64
```

[0]:
```
director.size #Retorna el tamaño
```

[0]:
```
4916
```

[0]:
```
director.shape #Retorna una tupla que representa la dimensión del DataFrame.
```

[0]:
```
(4916,)
```

[0]:
```
len(director) #Retorna el tamaño
```

[0]:
```
4916
```

[0]:
```
director.count() #Retorna el numero de valores "non-missing"
```

[0]:
```
4814
```

[0]:
```
actor_1_fb_likes.count() #Conteo de los valores
```

[0]:
```
4909
```

[0]:
```
actor_1_fb_likes.quantile() #Representa los datos en quartiles
```

[0]:
```
982.0
```

[0]:
```
#Retornamos medidaas estadisticas centrales como minimo, maximo, media,
#mediana, dispersión (con la cual sabemos la homogneidad), por ultimo envia la␣
↪suma de los datos
actor_1_fb_likes.min(), actor_1_fb_likes.max(), actor_1_fb_likes.mean(),␣
↪actor_1_fb_likes.median(), actor_1_fb_likes.std(), actor_1_fb_likes.sum()
```

[0]:
```
(0.0, 640000.0, 6494.488490527602, 982.0, 15106.986883848309, 31881444.0)
```

[0]:
```
actor_1_fb_likes.describe() #Realiza un resumen estadistico
```

[0]:
```
count      4909.000000
mean       6494.488491
std       15106.986884
min           0.000000
25%         607.000000
50%         982.000000
75%       11000.000000
max      640000.000000
Name: actor_1_facebook_likes, dtype: float64
```

[0]:
```
director.describe()
```

[0]:
```
count                 4814
unique                2397
top       Steven Spielberg
freq                    26
Name: director_name, dtype: object
```

[0]:
```
actor_1_fb_likes.quantile(.2) #Muestra el 20% del quartil
```

[0]:
```
510.0
```

[0]:
```
actor_1_fb_likes.quantile([.1,.2,.3,.4,.5,.6,.7,.8,.9]) #Muestra los␣
↪porcentajes de cada percentil
```

[0]:
```
0.1      240.0
0.2      510.0
0.3      694.0
0.4      854.0
         …
0.6     1000.0
0.7     8000.0
0.8    13000.0
0.9    18000.0
Name: actor_1_facebook_likes, Length: 9, dtype: float64
```

[0]:
```
director.isnull() #Serie de Booleans (tamaño de la serie)
```

[0]:
```
0       False
1       False
2       False
3       False
        …
4912     True
4913    False
4914    False
4915    False
```

```
Name: director_name, Length: 4916, dtype: bool
```

[0]:
```
#El metodo fillna reemplaza los valores perdidos
actor_1_fb_likes_filled=actor_1_fb_likes.fillna(0)
actor_1_fb_likes_filled.count() #Retorna la cantidad de valores luego del␣
 ↪llenado
```

[0]:
```
4916
```

[0]:
```
#El metodo dropna elimina los elementos con valores perdidos
actor_1_fb_likes_dropped=actor_1_fb_likes.dropna()
actor_1_fb_likes_dropped.size #Retorna la cantidad de valores luego de la␣
 ↪eliminación
```

[0]:
```
4909
```

[0]:
```
director.value_counts() #Envia el conteo de la serie
```

[0]:
```
Steven Spielberg    26
Woody Allen         22
Martin Scorsese     20
Clint Eastwood      20
                    ..
Dany Boon            1
Conor McPherson      1
Bradley Parker       1
Sol Tryon            1
Name: director_name, Length: 2397, dtype: int64
```

[0]:
```
director.value_counts(normalize=True) #Normaliza la serie
```

[0]:
```
Steven Spielberg    0.005401
Woody Allen         0.004570
Martin Scorsese     0.004155
Clint Eastwood      0.004155
                      …
Dany Boon           0.000208
Conor McPherson     0.000208
Bradley Parker      0.000208
Sol Tryon           0.000208
Name: director_name, Length: 2397, dtype: float64
```

[0]:
```
director.notnull() #Rastrea columnas llenas
```

[0]:
```
0        True
1        True
2        True
```

```
3        True
        …
4912    False
4913     True
4914     True
4915     True
Name: director_name, Length: 4916, dtype: bool
```

[0]:
```
director.hasnans #Retorna True si encuentra la variable
```

[0]:
```
True
```

[0]:
```
imdb_score=movies['imdb_score'] #Guardamos los valores de score en la variable␣
↪imdb_score
imdb_score #Imprimimos la variable
```

[0]:
```
0       7.9
1       7.1
2       6.8
3       8.5
       …
4912    7.5
4913    6.3
4914    6.3
4915    6.6
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score+1 #Le sumamos 1 al valor de la score
```

[0]:
```
0       8.9
1       8.1
2       7.8
3       9.5
       …
4912    8.5
4913    7.3
4914    7.3
4915    7.6
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score*2.5 #Multiplicamos por 2.5 el valor de la score
```

[0]:
```
0       19.75
1       17.75
2       17.00
3       21.25
        …
```

```
4912    18.75
4913    15.75
4914    15.75
4915    16.50
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score//7 #Dividimos por 7 el valor de score
```

[0]:
```
0       1.0
1       1.0
2       0.0
3       1.0
       …
4912    1.0
4913    0.0
4914    0.0
4915    0.0
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score>7 #Envia un boolean de acuerdo a la condición
```

[0]:
```
0        True
1        True
2       False
3        True
        …
4912     True
4913    False
4914    False
4915    False
Name: imdb_score, Length: 4916, dtype: bool
```

[0]:
```
director=='James Camero'#Retorna True or False en la busqueda del director
```

[0]:
```
0       False
1       False
2       False
3       False
        …
4912    False
4913    False
4914    False
4915    False
Name: director_name, Length: 4916, dtype: bool
```

[0]:
```
imdb_score.add(1) #Sumamos 1 al score
```

[0]:
```
0       8.9
1       8.1
2       7.8
3       9.5
       …
4912    8.5
4913    7.3
4914    7.3
4915    7.6
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score.mul(2.5) #Multiplicamos por 2.5 el valor de la score
```

[0]:
```
0       19.75
1       17.75
2       17.00
3       21.25
        …
4912    18.75
4913    15.75
4914    15.75
4915    16.50
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score.floordiv(7)#Dividimos por 7 el valor de score
```

[0]:
```
0       1.0
1       1.0
2       0.0
3       1.0
       …
4912    1.0
4913    0.0
4914    0.0
4915    0.0
Name: imdb_score, Length: 4916, dtype: float64
```

[0]:
```
imdb_score.gt(7)#Envia un boolean de acuerdo a la condición
```

[0]:
```
0        True
1        True
2       False
3        True
        …
4912     True
4913    False
4914    False
```

```
4915    False
Name: imdb_score, Length: 4916, dtype: bool
```

[0]:
```
director.eq('James Cameron') #Retorna True or False en la busqueda del director
```

[0]:
```
0        True
1       False
2       False
3       False
        …
4912    False
4913    False
4914    False
4915    False
Name: director_name, Length: 4916, dtype: bool
```

[0]:
```
imdb_score.astype(int).mod(5) #Divide y saca el residuo de la operación
```

[0]:
```
0       2
1       2
2       1
3       3
       ..
4912    2
4913    1
4914    1
4915    1
Name: imdb_score, Length: 4916, dtype: int64
```

[0]:
```
#Encadenar metodos de la serie
director.value_counts().head(3) #Agrupa e imprime los 3 primeros valores
```

[0]:
```
Steven Spielberg    26
Woody Allen         22
Martin Scorsese     20
Name: director_name, dtype: int64
```

[0]:
```
actor_1_fb_likes.isnull().sum()#Retorna la cantidad de numeros nulos
```

[0]:
```
actor_1_fb_likes.fillna(0).astype(int).head()#Elementos que no aparecen los␣
 ↪reemplaza por cero, se hace un cambio y envia la cabecera
```

[0]:
```
0     1000
1    40000
2    11000
3    27000
4      131
```

```
Name: actor_1_facebook_likes, dtype: int64
```

[0]:
```
actor_1_fb_likes.isnull().mean() #Retorna la media de los valores nulos
```

[0]:
```
0.0014239218877135883
```

[0]:
```
#Indice significativo
movie2=movies.set_index('movie_title')
movie2
```

[0]:
```
                                            color  … movie_facebook_likes
movie_title                                        …
Avatar                                      Color  …                33000
Pirates of the Caribbean: At World's End    Color  …                    0
Spectre                                     Color  …                85000
The Dark Knight Rises                       Color  …               164000
…                                             …  …                   …
The Following                               Color  …                32000
A Plague So Pleasant                        Color  …                   16
Shanghai Calling                            Color  …                  660
My Date with Drew                           Color  …                  456

[4916 rows x 27 columns]
```

[0]:
```
pd.read_csv('movie.csv',index_col='movie_title')#Almacenamos el indice
```

[0]:
```
                                            color  … movie_facebook_likes
movie_title                                        …
Avatar                                      Color  …                33000
Pirates of the Caribbean: At World's End    Color  …                    0
Spectre                                     Color  …                85000
The Dark Knight Rises                       Color  …               164000
…                                             …  …                   …
The Following                               Color  …                32000
A Plague So Pleasant                        Color  …                   16
Shanghai Calling                            Color  …                  660
My Date with Drew                           Color  …                  456

[4916 rows x 27 columns]
```

[0]:
```
#Renombrar los nombres de las filas y columnas
movies=pd.read_csv('movie.csv',index_col='movie_title')#Almacenamos el indice
idx_rename={'Avatar':'Avatares','Spectre':'Espectro'} # Renombramos datos
col_rename={'director_name':'Nombre del director','num_critic_for_reviews':
 ↪'Revisiones Criticas'} #Renombramos datos
movies.rename(index=idx_rename,columns=col_rename).head() #Visualizamos los␣
 ↪cambios
```

[0]:

movie_title	color	…	movie_facebook_likes
Avatares	Color	…	33000
Pirates of the Caribbean: At World's End	Color	…	0
Espectro	Color	…	85000
The Dark Knight Rises	Color	…	164000
Star Wars: Episode VII - The Force Awakens	NaN	…	0

[5 rows x 27 columns]