

Python para Análisis de Datos

Módulo 05

Pandas

groupby

Como hemos visto, es muy fácil hacer estadísticas sobre columnas de un dataframe. Pero muchas veces estamos interesados en realizar estadísticas a subconjuntos de esos datos según la categoría a la que pertenezcan. Por ejemplo, encontrar el promedio de edad para hombres y mujeres, o para hombres y mujeres divididos también por la clase en la que viajaban.

Para este tipo de operaciones existe el método `groupby`, que permite hacer los agrupamientos de una forma eficiente y sencilla. Este método crea una estructura de datos particular a la que le podemos aplicar diferentes operaciones estadísticas. Sólo cuando aplicamos estas operaciones se devuelve un dataframe con la información.



groupby

Típicamente vamos a usar el `groupby` para agrupar los datos según los valores de una o más columnas y lo vamos a combinar con algún método de `data` para realizar las

estadísticas sobre el resto de las columnas si es posible. El método `groupby` sólo crea la estructura de datos a la que se le puede aplicar la función de agregación.

```
data.groupby("Sex")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f802a7542b0>
```

```
data.groupby("Sex").count()
```

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Sex											
female	314	314	314	314	261	314	314	314	314	97	312
male	577	577	577	577	453	577	577	577	577	107	577

groupby

Si la operación a realizar solamente admite números, se aplica a las columnas numéricas exclusivamente.

```
data.groupby("Sex").mean()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
Sex							
female	431.028662	0.742038	2.159236	27.915709	0.694268	0.649682	44.479818
male	454.147314	0.188908	2.389948	30.726645	0.429809	0.235702	25.523893

groupby

Si queremos restringir el resultado a ciertas columnas podemos hacerlo de varias maneras:

- Una es seleccionar la columna de interés luego de hacer las operaciones.
- Otra es seleccionar las columnas necesarias antes de hacer el groupby (se debe incluir la columna con la que agrupar).
- Y la otra opción es seleccionar la columna de interés después de hacer el groupby, pero antes de aplicar la función de agregación.



groupby

El tipo de estructura a devolver puede variar según el método de selección:

```
data.groupby("Sex").mean()["Age"]
```

```
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

```
data[["Age", "Sex"]].groupby("Sex").mean()
```

	Age
Sex	
female	27.915709
male	30.726645

```
data.groupby("Sex")["Age"].mean()
```

```
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

groupby

```
# promedio de edad según sexo  
data[["Sex", "Age"]].groupby("Sex").mean()
```

Age	
Sex	
female	27.915709
male	30.726645

```
# máximo de edad en cada clase  
data[["Age", "Pclass"]].groupby("Pclass").max()
```

Age	
Pclass	
1	80.0
2	70.0
3	74.0

groupby

```
# cantidad de valores no nulos por columna agrupados por sobrevivencia  
data.groupby("Survived").count()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Survived											
0	549	549	549	549	424	549	549	549	549	68	549
1	342	342	342	342	290	342	342	342	342	136	340

```
# cantidad de filas agrupadas por sobrevivencia  
data.groupby("Survived").size()
```

```
Survived  
0      549  
1      342  
dtype: int64
```

groupby

En caso de querer realizar varias operaciones a los mismos grupos se puede utilizar aggregate (o su alias, agg).

```
data[["Age", "Pclass"]].groupby("Pclass").agg(["min", "max", "mean", "std"])
```

	Age			
	min	max	mean	std
Pclass				
1	0.92	80.0	38.233441	14.802856
2	0.67	70.0	29.877630	14.001077
3	0.42	74.0	25.140620	12.495398

groupby

También se puede agrupar por varias categorías, haciendo agrupamientos más específicos. Por ejemplo, el promedio de edad discriminando tanto por clase como por sexo.

```
data[["Pclass", "Sex", "Age"]].groupby(["Pclass", "Sex"]).mean()
```

Age		
Pclass	Sex	
1	female	34.611765
	male	41.281386
2	female	28.722973
	male	30.740707
3	female	21.750000
	male	26.507589

groupby

```
# proporción de supervivencia por clase y por sexo  
data[["Pclass", "Sex", "Survived"]].groupby(["Pclass", "Sex"]).mean()
```

		Survived
Pclass	Sex	
1	female	0.968085
	male	0.368852
2	female	0.921053
	male	0.157407
3	female	0.500000
	male	0.135447



¡Muchas gracias!

¡Sigamos trabajando!