rp

June 29, 2023

[1]:
```
#Reshaping a pandas dataframe is one of the most common data wrangling tasks in␣
 ↪the data analysis world.
#It is also referred to as transposing or pivoting/unpivoting a table from long␣
 ↪to wide or from wide to long format.
#So what is a long data format vs. a wide data format and how do we reshape a␣
 ↪dataframe from long-to-wide and vice versa?
```

[2]:
```
#Syntax: Pandas.Series.values.reshape((dimension))

#Return: return an ndarray with the values shape if the specified shape matches␣
 ↪exactly the current shape,
#then return self (for compat)
```

[7]:
```
# import pandas library
import pandas as pd

# make an array
array = [2, 4, 6, 8, 10, 12]

# create a series
series_obj = pd.Series(array)

# convert series object into array
arr = series_obj.values

arr
```

[7]:
```
array([ 2,  4,  6,  8, 10, 12], dtype=int64)
```

[17]:
```
# reshaping series
reshaped_arr = arr.reshape((3, 2))

# show
reshaped_arr
```

[17]:
```
array([[ 2,  4],
       [ 6,  8],
```

```
       [10, 12]], dtype=int64)
```

[18]:
```
# import pandas library
import pandas as pd

# make an array
array = ["ankit","shaurya",
        "shivangi", "priya",
        "jeet","ananya"]

# create a series
series_obj = pd.Series(array)

print("Given Series:\n", series_obj)

# convert series object into array
arr = series_obj.values

arr
```

```
Given Series:
 0       ankit
1     shaurya
2    shivangi
3       priya
4        jeet
5      ananya
dtype: object
```

[18]:
```
array(['ankit', 'shaurya', 'shivangi', 'priya', 'jeet', 'ananya'],
      dtype=object)
```

[19]:
```
# reshaping series
reshaped_arr = arr.reshape((2, 3))

# show
print("After Reshaping: \n", reshaped_arr)
```

```
After Reshaping:
 [['ankit' 'shaurya' 'shivangi']
 ['priya' 'jeet' 'ananya']]
```

[20]:
```
# The pivot() function is used to reshaped a given DataFrame organized by given␣
 ↪index / column values.
#This function does not support data aggregation, multiple values will result␣
 ↪in a MultiIndex in the columns.
```

```
#Syntax: DataFrame.pivot(self, index=None, columns=None, values=None)␣
 ↪Parameters: Name
```

[33]:
```
# pandas.pivot(index, columns, values) function produces pivot table based on 3␣
 ↪columns of the DataFrame.
#Uses unique values from index / columns and fills with values.

#Parameters:
#index[ndarray] : Labels to use to make new frame's index
#columns[ndarray] : Labels to use to make new frame's columns
#values[ndarray] : Values to use for populating new frame's values

#Returns: Reshaped DataFrame
#Exception: ValueError raised if there are any duplicates.
```

[34]:
```
# Create a simple dataframe

# importing pandas as pd
import pandas as pd

# creating a dataframe
df = pd.DataFrame({'A': ['John', 'Boby', 'Mina'],
    'B': ['Masters', 'Graduate', 'Graduate'],
    'C': [27, 23, 21]})

df
```

[34]:

	A	B	C
0	John	Masters	27
1	Boby	Graduate	23
2	Mina	Graduate	21

[35]:
```
# values can be an object or a list
df.pivot('A', 'B', 'C')
```

```
<ipython-input-35-790241475f3a>:2: FutureWarning: In a future version of pandas
all arguments of DataFrame.pivot will be keyword-only.
  df.pivot('A', 'B', 'C')
```

[35]:

B	Graduate	Masters
A		
Boby	23.0	NaN
John	NaN	27.0
Mina	21.0	NaN

[36]:
```
# value is a list
df.pivot(index ='A', columns ='B', values =['C', 'A'])
```

[36]:

	C		A	
B	Graduate	Masters	Graduate	Masters
A				
Boby	23	NaN	Boby	NaN
John	NaN	27	NaN	John
Mina	21	NaN	Mina	NaN

[42]:
```
#the pivot table has been created for the given dataset where the gender␣
↪percentage has been calculated.
```

[43]:
```
# importing pandas library
import pandas as pd

# creating dataframe
df = pd.DataFrame({'Name': ['John', 'Sammy', 'Stephan', 'Joe', 'Emily', 'Tom'],
                   'Gender': ['Male', 'Female', 'Male',
                              'Female', 'Female', 'Male'],
                   'Age': [45, 6, 4, 36, 12, 43]})
print("Dataset")
print(df)
print("-"*40)

# categorizing in age groups
def age_bucket(age):
    if age <= 18:
        return "<18"
    else:
        return ">18"

df['Age Group'] = df['Age'].apply(age_bucket)

# calculating gender percentage
gender = pd.DataFrame(df.Gender.value_counts(normalize=True)*100).reset_index()
gender.columns = ['Gender', '%Gender']
df = pd.merge(left=df, right=gender, how='inner', on=['Gender'])

# creating pivot table
table = pd.pivot_table(df, index=['Gender', '%Gender', 'Age Group'],
                       values=['Name'], aggfunc={'Name': 'count',})

# display table
print("Table")
print(table)
```

```
Dataset
      Name  Gender  Age
0     John    Male   45
```

```
1    Sammy  Female    6
2  Stephan    Male    4
3      Joe  Female   36
4    Emily  Female   12
5      Tom    Male   43
---------------------------------------
Table
                           Name
Gender %Gender Age Group
Female 50.0    <18           2
               >18           1
Male   50.0    <18           1
               >18           2
```