R to Python

Leonardo Uchoa

3/31/2020

Contents

1 The table

That's my R to python port. It's intended to make my approach to learning python faster and its mostly composed of data wrangling routine tools. Many of those are already listed in other sources.

R	Python
dim	df.shape (pd)
str	df.dtypes / df.info (pd)
unique	np.unique (np)
summary	df.describe (pd)
`group_by`	df.groupby (pd)
count	`df.value_count` (pd)
—	np.bincount (np)
apply	df.apply (pd)
if.else	df.where[case,true,false] (pd)
table	pd.crosstab
mutate(df, c=a-b)	df.assign(c=df['a']-df['b']) (pd)
colSums(is.na())	df.isnull().sum() (pd)
na.omit	df.dropna(axis=X) (pd)
imputation	df.fillna(df.mean()) (pd)
colnames() <-	df.colnames (pd)

Abreviations

Module Abreviation	Module
pd	Pandas
np	Numpy

2 Usefull examples

These are ilustrations of the commands I use the most when analysing data in R and some additional because they're different from what I'm used to do in R.

```
library(reticulate)
use_python("/home/leonardo/anaconda3/bin/python")
```

```
import numpy as np
import pandas as pd
```

2.1 Counting per column

```
df = pd.DataFrame(np.random.randint(0, 2, (10, 4)), columns=list('abcd'))
df.apply(pd.Series.value_counts)
```

```
##    a  b  c  d
## 0  7  6  6  6
## 1  3  4  4  4
```

2.2 Categorical data encoding

```
df = pd.DataFrame([
['green', 'M', 10.1, 'class2'],
['red', 'L', 13.5, 'class1'],
['blue', 'XL', 15.3, 'class2']])
df.columns = ['color', 'size', 'price', 'classlabel']

df
```

```
##    color size  price classlabel
## 0  green    M   10.1     class2
## 1    red    L   13.5     class1
## 2   blue   XL   15.3     class2
```

In both approaches bellow we use a dictionary the create the mapping identifier for the `map` method. Remember that according to w3schools a dictionary is

> *A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.*

2.2.1 Encoding ordinals - create labels manually

```
#create the dict mapping from ordinal to integer
size_mapping = {'XL': 3,'L': 2,'M': 1}

#use map to in the desired column get the mapped values
df['size'] = df['size'].map(size_mapping)
```

2.2.2 Encoding nominals - creating labels automatically

```
class_mapping = {label: idx for idx, label in enumerate(np.unique(df['classlabel']))}
```

Now what that command is doing is looping through the iterators `idx` and `label` (created by the `enumerate` function) in the unique values of the `classlabel` column and assigning both to `label` and `idx`. Let's see

```
print(list(
    enumerate(np.unique(df['classlabel']))
  ))
```

```
## [(0, 'class1'), (1, 'class2')]
```

So iterating through the list we get to assign "class1"/"class2" to `label` and 0/1 to `idx`[1]. Finally the last step to map

```
df['classlabel'] = df['classlabel'].map(class_mapping)
df
```

```
##    color  size  price  classlabel
## 0  green     1   10.1           1
## 1    red     2   13.5           0
## 2   blue     3   15.3           1
```

Want to get the mapping backwards? Access the `items` method in the `class_mapping` object and loop again

```
inv_class_mapping = {a:b for b,a in class_mapping.items()}
df['classlabel'] = df['classlabel'].map(inv_class_mapping)
df
```

```
##    color  size  price classlabel
## 0  green     1   10.1     class2
## 1    red     2   13.5     class1
## 2   blue     3   15.3     class2
```

Ps.: There's also an object in `skitlearn` module `preprocessing` that does this: `LabelEncoder`

3 References

[1]. Pandas: https://pandas.pydata.org/pandas-docs/stable/getting_started/comparison/comparison_with_r.html#quick-reference

[2]. Raschka, S. and Mirjalili, V., 2019. Python Machine Learning. Birmingham: Packt Publishing, Limited.

[1]Note the inversion in 'label: idx for idx, label'