Digital Skills Training for Student
Central Computer Center, KUET

ICT DIVISION
FUTURE IS HERE

Machine Learning with Python (ML-3)

Session 8: Pandas Series

Contents

Session 8: Pandas Series

- What is Pandas?
- Introduction to Pandas Series
- Series Methods
- Series with Python functionalities
- Boolean Indexing on Series
- Plotting graphs on series

Pandas

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Install it using this command:

```
C:\Users\Your Name>pip install pandas
```

```
import pandas
import pandas as pd
print(pd.__version__)
```

https://pandas.pydata.org/
https://github.com/pandas-dev/pandas

Pandas Series

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

If nothing else is specified, the values are labeled with their index number.

```
print(myvar[0])
```

Pandas DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

```
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
myvar = pd.DataFrame(data)

print(myvar)
```

Pandas Series

With the index argument, you can name your own labels.

```
a = [1, 7, 2]
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
print(myvar["y"])
```

You can also use a key/value object, like a dictionary, when creating a Series.

```
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
myvar = pd.Series(calories, index = ["day1", "day2"])
print(myvar)
```

Pandas Series


```
# Creating empty series
ser = pd.Series()

# simple array
data = np.array(['g','e','e','k','s'])
ser = pd.Series(data)

# providing an index
ser = pd.Series(data, index=[10, 11, 12, 13, 14])

print(ser)
```

Pandas Series


```
# a simple list
list = ['g', 'e', 'e', 'k', 's']

# create series form a list
ser = pd.Series(list)

dict = {'Geeks': 10,
        'for': 20,
        'geeks': 30}
# create series from dictionary
ser = pd.Series(dict)
print(ser)
```

Pandas Series

```
# giving a scalar value with index
ser = pd.Series(10, index=[0, 1, 2, 3, 4, 5])

print(ser)

# series with numpy linspace()
ser1 = pd.Series(np.linspace(3, 33, 3))
print(ser1)

ser=pd.Series(range(10))
print(ser)
```

Practice: Write the code to define a Pandas Series using a range of numbers as data and a string's characters as the index.

Accessing elements of a Pandas Series

There are two ways through which we can access element of series

- Accessing Element from Series with Position
- Accessing Element Using Label (index)

```
data = np.array(['g','e','e','k','s','f',
'o','r','g','e','e','k','s'])
ser = pd.Series(data)

#retrieve the first element
print(ser[:5])
print(ser[-10:])
```

Accessing elements of a Pandas Series


```
# creating simple array
data = np.array(['g','e','e','k','s','f'])
ser = pd.Series(data,index=[10,11,12,13,14,15])

# accessing a element using index element
print(ser[16])
print(ser[[10, 11, 12, 13, 14]])

ser = pd.Series(np.arange(3, 9), index=['a', 'b', 'c',
'd', 'e', 'f'])

print(ser[['a', 'd']])
```

Indexing and Selecting Data in Series


```
# creating simple array
data = np.array(['g','e','e','k','s','f'])
ser = pd.Series(data,index=[10,11,12,13,14,15])

ser[3:6]
ser.loc[13:16]  Label-based indexing [inclusive of both start and stop]
ser.iloc[3:6]   Position-based indexing [exclusive of the stop position,
                                         similar to Python list slicing]
```

Binary Operation on Series

We can perform binary operation on series like addition, subtraction and many other operation. In order to perform binary operation on series we have to use some function.

```
# creating a series
data = pd.Series([5, 2, 3,7], index=['a', 'b', 'c',
'd'])
data1 = pd.Series([1, 6, 4, 9], index=['a', 'b', 'd',
'e'])

data.add(data1, fill_value=0)
data.sub(data1, fill_value=0)
```

Binary Operation on Series

add()	Method is used to add series or list like objects with same length to the caller series
sub()	Method is used to subtract series or list like objects with same length from the caller series
mul()	Method is used to multiply series or list like objects with same length with the caller series
div()	Method is used to divide series or list like objects with same length by the caller series
sum()	Returns the sum of the values for the requested axis
prod()	Returns the product of the values for the requested axis
mean()	Returns the mean of the values for the requested axis
pow()	Method is used to put each element of passed series as exponential power of caller series and returned the results
abs()	Method is used to get the absolute numeric value of each element in Series/DataFrame
cov()	Method is used to find covariance of two series

Conversion Operation on Series

In conversion operation we perform various operation like changing datatype of series, changing a series to list etc. In order to perform conversion operation we have various function which help in conversion like .astype(), .tolist() etc.

```
# reading csv file from url
data = pd.read_csv("nba.csv")

# dropping null value columns to avoid errors
data.dropna(inplace = True)

# storing dtype before converting
before = data.dtypes
```

Conversion Operation on Series


```
# converting dtypes using astype
data["Salary"]= data["Salary"].astype(int)
data["Number"]= data["Number"].astype(str)

# storing dtype after converting
after = data.dtypes

# printing to compare
print("BEFORE CONVERSION\n", before, "\n")
print("AFTER CONVERSION\n", after, "\n")
```

Conversion Operation on Series

```
# converting to list
salary_list = data["Salary"].tolist()

# storing dtype after operation
dtype_after = type(salary_list)

# printing dtype
print("Data type before converting = {}\nData type after
converting = {}"
      .format(dtype_before, dtype_after))

salary_list
```

Pandas series method

Series()	A pandas Series can be created with the Series() constructor method. This constructor method accepts a variety of inputs
combine_first()	Method is used to combine two series into one
count()	Returns number of non-NA/null observations in the Series
size()	Returns the number of elements in the underlying data
name()	Method allows to give a name to a Series object, i.e. to the column
is_unique()	Method returns boolean if values in the object are unique
idxmax()	Method to extract the index positions of the highest values in a Series
idxmin()	Method to extract the index positions of the lowest values in a Series
sort_values()	Method is called on a Series to sort the values in ascending or descending order

Pandas series method

sort_index()	Method is called on a pandas Series to sort it by the index instead of its values
head()	Method is used to return a specified number of rows from the beginning of a Series. The method returns a brand new Series
tail()	Method is used to return a specified number of rows from the end of a Series. The method returns a brand new Series
le()	Used to compare every element of Caller series with passed series.It returns True for every element which is Less than or Equal to the element in passed series
ne()	Used to compare every element of Caller series with passed series. It returns True for every element which is Not Equal to the element in passed series
ge()	Used to compare every element of Caller series with passed series. It returns True for every element which is Greater than or Equal to the element in passed series

Pandas series method

eq()	Used to compare every element of Caller series with passed series. It returns True for every element which is Equal to the element in passed series
gt()	Used to compare two series and return Boolean value for every respective element
lt()	Used to compare two series and return Boolean value for every respective element
clip()	Used to clip value below and above to passed Least and Max value
clip_lower()	Used to clip values below a passed least value
clip_upper()	Used to clip values above a passed maximum value
astype()	Method is used to change data type of a series
tolist()	Method is used to convert a series to list

Pandas series method

get()	Method is called on a Series to extract values from a Series. This is alternative syntax to the traditional bracket syntax
unique()	Pandas unique() is used to see the unique values in a particular column
nunique()	Pandas nunique() is used to get a count of unique values
value_counts()	Method to count the number of the times each unique value occurs in a Series
factorize()	Method helps to get the numeric representation of an array by identifying distinct values
map()	Method to tie together the values from one object to another
between()	Pandas between() method is used on series to check which values lie between first and second argument

Pandas Plotting

Pandas uses the plot() method to create diagrams.

```
df = pd.read_csv('data.csv')

df.plot()

df.plot(kind = 'scatter', x = 'Duration', y = 'Calories’)

df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse’)

df["Duration"].plot(kind = 'hist')
```

Practice Problem

1. Filter and display all the rows where the pulse is greater than 110.
2. Calculate the mean (average) of all column.
3. Identify the row where the Maxpulse is the highest.
4. Count how many exercises had a Duration greater than 50 minutes.
5. Find the minimum and maximum values in the Calories column.
6. Sort the dataset based on the Maxpulse in descending order.

Thank You!