Python Libraries Documentation

Pandas Dataframe, NumPy

1. Pandas Dataframe:

1.1. Making Dataframe:

pandas.DataFrame(*data*=*None*) - >to make a dataframe from a dictionary.

- Data: the data you want to change to dataframe as dictionary.

1.2. Reading from CSV:

pandas.read_csv(*filepath*, *sep*=*_NoDefault.no_default*, .. others) -> to read a csv file

- filepath: the path of the csv
- sep: define a separator to separate the data; default ’,’

1.3. Editing row labels:

DataFrame.index ->To show the index of the dataframe and used to edit them.

1.4. Column labels:

DataFrame.columns -> Show the label of the columns

1.5. Column data types:

DataFrame.dtypes -> return list of data type of each column.

1.6. Dataframe info:

DataFrame.info() ->prints information about a DataFrame.

1.7. Getting Dataframe first rows:

DataFrame.head(*n*=*5*) -> get n first rows

1.8. Getting Dataframe last rows:

DataFrame.tail(*n*=*5*) -> getting last n rows.

1.9. Dataframe array dimension:

DataFrame.ndim ->It shows the dimensions of the dataframe.

Return 1 if Series. Otherwise return 2 if Dataframe.

1.10. Dataframe size:

***DataFrame.size* ->** Return size of dataframe.

1.11. Dataframe shape:

***DataFrame.shape* ->** Return a tuple representing the dimension of the Dataframe.

1.12. Dataframe empty:

DataFrame.empty -> To know if dataframe is empty.

1.13. Drop missing values:

DataFrame.dropna(*axis*=*0*, *how*=*_NoDefault.no_ default*) -> Remove missing values.

- Axis:{‘0’ or index, ‘1’ or column} to drop from rows or columns.
- How: remove the row or column if there is {‘any’, ’all’} NA values. Default: any

1.14. Detecting missing values:

DataFrame.isna() **->** Detect missing values.

1.15. Copying Dataframe:

DataFrame.copy(*deep*=*True*) **->** copy the dataframe either deep if ***deep*** = True or shallow if its false.

1.16. loc:

DataFrame.loc -> Select rows and columns by label(s) or a boolean array.

1.17. iloc:

DataFrame.iloc->Select rows and columns by using its index position.

1.18. Insert:

DataFrame.insert(*loc*, *column*, *value*, *allow_duplicates*) ->Insert column into DataFrame at specified location.

- Loc: insertion location
- Column : column name
- Value: column value
- Allow_duplicates: True to allow the insertion of duplicate columns.

1.19. set_index:

DataFrame.set_index(*keys*, *drop*=*True*) -> Set the DataFrame index using existing columns.

- Keys: column label(s).
- Drop: delete column that is used as index.

1.20. fillna:

DataFrame.fillna(*value=None*, *axis=None*) -> Fill NA values.

- Value: the value to fill with the NA values.
- Axis: {0 or ‘index’, 1 or ‘columns’} default : 0.

1.21. group by:

DataFrame.groupby(*by*=*None*, *axis*=*_NoDefault.no_default*) -> Group dataframe using certain values.

- By: the column to group with.
- Axis: {0 or ‘index’, 1 or ‘columns’}, default 0 to Split along rows or columns.

1.22. Aggregate:

DataFrame.aggregate(*func*=*None*, *axis*=*0*) ->Aggregate using one or more operations. as: mean, max,min ...etc.

- Fun: the function(s) to aggregate with.
- Axis: axis{0 or ‘index’, 1 or ‘columns’}, default 0
  If 0 apply to each column. If 1 apply to each row.

1.23. Merge:

DataFrame.merge(*right*, *how*=*'inner'*, *on*=*None*) -> Merge Dataframes

- right : the dataframe on the right to merge with.
- how: the way to merge the dataframes types: {‘left’, ‘right’, ‘outer’, ‘inner’, ‘cross’}, default ‘inner’

1.24. Join:

DataFrame.join(*other*, *on*=*None*, *how*=*'left'*) -> join columns of another DataFrame.

- Other: the other(right) dataframe to join with.
- On: the column to join on.
- how: the way to join the dataframes types: {‘left’, ‘right’, ‘outer’, ‘inner’, ‘cross’}, default ‘left’

1.25. Mathematical Functions:

Parameteres:

- axis: {index (0), columns (1)} the axis for the function be applied to.
-skipna: exclude NA values if True
-numeric_only: include only float, int, boolean columns.

-Max:

DataFrame.max(*axis*=*0*, *skipna*=*True*, *numeric_only*=*False*) -> Return the maximum of the values.

-Mean:

DataFrame.mean(*axis*=*0*, *skipna*=*True*, *numeric_only*=*False*) -> Return the mean of the values.

-Median:

DataFrame.median(*axis*=*0*, *skipna*=*True*, *numeric_only*=*False*) -> Return the median of the values.

-Min:

DataFrame.min(*axis*=*0*, *skipna*=*True*, *numeric_only*=*False*) -> Return the minimum value.

-Mode:

DataFrame.mode(*axis*=*0*, *numeric_only*=*False*, *dropna*=*True*) -> Get the mode(s).

-Count:

DataFrame.count(*axis*=*0*, *numeric_only*=*False*) -> Count non-NA cells for each column or row.

-Sum:

DataFrame.sum(*axis*=*0*, *skipna*=*True*, *numeric_only*=*False*) -> Return the sum of the values.

1.26. Visualization Function:

DataFrame.plot(*kind=line*) ->Make plots of Series or DataFrame.

- Kind : type of plot as histogram {hist}, bar, box{boxplot} ..etc.

1.27. Filter function:

DataFrame.filter(*items*=*None*, *like*=*None*, *regex*=*None*, *axis*=*None*) -> Subset the dataframe rows or columns according to the specified index labels.
-items: items(index ,column name(s)) to filter with
-like: search for label that matches the *like in label*
-regex: search with regular expression.
-axis: {0 or 'index', 1 or 'columns', None}, default None
The axis to filter on.

1.28. nunique:

DataFrame.nunique(*axis*=*0*, *dropna*=*True*)
-> Count number of distinct elements in specified axis.
-axis: {0 or 'index', 1 or 'columns'}, default 0. Axis to use
-dropna: if true don't include NAN.

1.29. drop:

DataFrame.drop(*labels*=*None*, *axis*=*0*,index_*level*=*None*) -> Drop specified labels from rows or columns.
-labels: index or column label to drop.
-axis: {0 or 'index', 1 or 'columns'}, default 0
Whether to drop labels from the index or columns
-index_level: to choose what level to drop from if its multilevel.

1.30. drop duplicates:

DataFrame.drop_duplicates(*subset*=*None*, *keep*=*'first'*)-> Return DataFrame with duplicate rows removed.
-subset: choose certain columns to drop from.

-keep: which duplicates to keep. {first,last,False} , first: to keep the first one only; last; to keep the last one only; False to drop all duplicates.

2. NumPy:

Import nupmy as np

2.1. numpy array creation:

2.1.1. Convert python sequence to numpy array:

numpy.array(*object*, *dtype(optional)*=*None*, *dimension(optional*=*0*))

-object: array to change.

-dtype: the desired data type.

-dimension: the minimum dimensions wanted.

-array: array to convert.

2.1.2. Intrinsic numpy array creation functions:

1- 1D Array:

numpy.arange*(start(optional),stop, step(optional))*->return array with specified interval.

-start: start of the interval. Default: 0.

-stop: end of the interval.

-step: space between values. Default: 1.

2- 2D Array:

numpy.eye(N,M=None(optional), k(optional)=0) -> Return a 2-D array with ones on the diagonal and zeros elsewhere.

-N : no. of rows of output.

-M: no. of columns of output. Default: N.

-k: determine the diagonals with ones.

{0: main, +ve: upper, -ve: lower}

3- General ndarray creation functions:

numpy.ones(*shape*, *dtype*=*None*)

->Return a new array of given shape and type, filled with ones.

-Shape: shape of the output array.
-dtype: data type of the output array. Default: float64.

numpy.zeros(*shape*, *dtype*=*float*)
-> Return a new array of given shape and type, filled with zeros.
-Shape: shape of the output array.
-dtype: data type of the output array. Default: float64.

2.1.3. Replicating, joining or mutating existing arrays:

Copying:

numpy.copy(*a*) -> Return an array copy.

-a: input array

Joining:

numpy.block(*arrays*) -> Assemble an nd-array from nested lists of blocks.
-arrays : arrays to assemble.

2.2. numpy sizing:

numpy.shape(array) -> Return the shape of an array.
numpy.size(array, axis(option)=None) -> Return the number of elements along a given axis.

- Axis: Axis along which the elements are counted. By default: give the total number of elements.

numpy.reshape(*array*,*shape=None*) ->Gives a new shape to an array without changing its data.

- Shape: the new shape of the array.

2.3. Non Zero search:

numpy.nonzero(*array*) -> Return the indices of the elements that are non-zero.

2.5.universal functions (for 2 arrays):

***class* numpy.ufunc()->**Functions that operate element by element on whole arrays.

Parameters:

2.5.1. Arithmetic operations:

-Sum: **numpy.add(*array1*, *array2)***

-Subtract: **numpy.subtract(*array1*, *array2*)**

-Multiply: **numpy.multiply(*array1*, *array2*)**

-Divide: **numpy.divide(*array1*, *array2*)**

-Power: **numpy.power(*array1*, *array2*)->** First array elements raised to powers from second array,

2.5.2. Trig Functions:

-Sin: **numpy.sin(*array*)**

-Cos: **numpy.cos(*array)***

-Tan: **numpy.tan(*array)***

2.5.3. Comparison functions:

-Greater: **numpy.greater(*array1,array2)***

-Greater than or equal to: **numpy.greater_equal(*array1,array2)***

-Less: **numpy.less(*array1,array2)***

-less or equal to:

numpy.less_equal(*array1,array2)*

-not equal to:

numpy.not_equal(*array1,array2)*

2.6.Mathematical functions for one array:

-Sum: **numpy.sum(*array*)**

-Product: **numpy.prod(*array*)**

-Cumulative Sum: **numpy.cumulative_sum(*array*)**

-Maximum: **numpy.max(*array*)**

-Minimum: **numpy.min(*array*)**

2.6.Linear Algebra Functions:

linalg.multi_dot(*arrays*) ->Compute the dot product of two or more arrays.

2.7.argmin:

numpy.argmin(*a*rray)-> Returns the indices of the minimum values.