AN7914 Week 9 Python

April 2, 2024

1 Week 9 Python

[1]:
```
import pandas as pd
import numpy as np
```

2 Cleaning Data

Let's create a fake dataset

[2]:
```
people = {
    'first': ['Sakib', 'Jane', 'John', 'Chris', np.nan, None, 'NA'],
    'last': ['Anwar', 'Doe', 'Doe', 'Schafer', np.nan, np.nan, 'Missing'],
    'email': ['SakibAnwar@email.com', 'JaneDoe@email.com', 'JohnDoe@email.com',␣
 ↪None, np.nan, 'Anonymous@email.com', 'NA'],
    'age': ['100', '55', '63', '36', None, None, 'Missing']
}
```

[3]:
```
df = pd.DataFrame(people) # Convert it to a dataframe
```

[4]:
```
df
```

[4]:

	first	last	email	age
0	Sakib	Anwar	SakibAnwar@email.com	100
1	Jane	Doe	JaneDoe@email.com	55
2	John	Doe	JohnDoe@email.com	63
3	Chris	Schafer	None	36
4	NaN	NaN	NaN	None
5	None	NaN	Anonymous@email.com	None
6	NA	Missing	NA	Missing

[5]:
```
df.isna()
```

[5]:

	first	last	email	age
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	True	False

```
4   True   True   True   True
5   True   True  False   True
6  False  False  False  False
```

`df.isna()` is a method in pandas, a popular Python library for data manipulation and analysis. This method is used to detect missing or NA (Not Available) values in a DataFrame (df). It returns a DataFrame of the same shape as the input DataFrame (df), where each element is either True if it's NA or False if it's not NA. This allows users to easily identify missing data within their datasets for further analysis or handling.

[6]:
```
df.dropna()
```

[6]:
```
    first     last                  email      age
0  Sakib    Anwar  SakibAnwar@email.com      100
1   Jane      Doe     JaneDoe@email.com       55
2   John      Doe     JohnDoe@email.com       63
6     NA  Missing                    NA  Missing
```

`df.dropna()` is a method in the pandas library used to remove rows or columns from a DataFrame that contain missing values (NA values). When you call `df.dropna()`, it returns a new DataFrame with rows or columns containing NA values removed. By default, it drops rows containing any NA values.

Here's a brief breakdown:

- If you call `df.dropna(axis=0)`, it drops rows containing any NA values.
- If you call `df.dropna(axis=1)`, it drops columns containing any NA values.
- If you want to customize the behavior, you can use additional parameters such as `subset` to specify particular columns or rows to consider for dropping NA values, and `thresh` to specify a threshold for the number of NA values allowed before dropping a row or column.

[7]:
```
df.dropna(axis='index', how='any')
```

[7]:
```
    first     last                  email      age
0  Sakib    Anwar  SakibAnwar@email.com      100
1   Jane      Doe     JaneDoe@email.com       55
2   John      Doe     JohnDoe@email.com       63
6     NA  Missing                    NA  Missing
```

The code `df.dropna(axis='index', how='any')` is using the `dropna()` method in pandas to remove rows from the DataFrame (`df`) that contain any missing values (NA values). Here's a breakdown of the parameters used:

- `axis='index'`: This specifies that the operation should be applied along the index axis, which means rows will be considered.
- `how='any'`: This parameter specifies the condition for dropping rows. In this case, it's set to 'any', meaning that any row containing at least one NA value will be dropped.

So, when you call `df.dropna(axis='index', how='any')`, it returns a new DataFrame with rows containing any NA values removed.

Here's another example:

```
# Create a DataFrame with NA values
df = pd.DataFrame({'A': [1, None, 3, 4],
                   'B': [None, 2, 3, None]})

# Drop rows containing any NA values
df_cleaned = df.dropna(axis='index', how='any')

print("DataFrame after dropping rows with any NA values:")
print(df_cleaned)
```

This code will remove rows 1 and 3 from the original DataFrame because they contain NA values in either column 'A' or column 'B'.

In pandas, the `how` parameter in the `dropna()` method allows you to specify different conditions for dropping rows or columns based on the presence of NA values. Here are the available options for the `how` parameter and their explanations:

1. `how='any'`: This is the default option. It drops rows or columns containing any NA values.
2. `how='all'`: This option drops rows or columns only if all values are NA. In other words, it removes rows or columns that are entirely composed of NA values.

[8]:
```
df.dropna(axis='index', how='all', subset=['last', 'email'])
```

[8]:

	first	last	email	age
0	Sakib	Anwar	SakibAnwar@email.com	100
1	Jane	Doe	JaneDoe@email.com	55
2	John	Doe	JohnDoe@email.com	63
3	Chris	Schafer	None	36
5	None	NaN	Anonymous@email.com	None
6	NA	Missing	NA	Missing

The code `df.dropna(axis='index', how='all', subset=['last', 'email'])` is using the `dropna()` method in pandas to remove rows from the DataFrame (`df`) based on specific columns (`'last'` and `'email'`) where all values are missing. Here's a breakdown of the parameters used:

- `axis='index'`: This specifies that the operation should be applied along the index axis, which means rows will be considered for removal.
- `how='all'`: This parameter specifies the condition for dropping rows. In this case, it's set to 'all', meaning that rows will only be dropped if all values in the specified subset are missing.
- `subset=['last', 'email']`: This parameter specifies the subset of columns to consider when checking for missing values. In this case, only the columns `'last'` and `'email'` are considered. If all values in both of these columns are missing for a particular row, that row will be dropped.

So, when you call `df.dropna(axis='index', how='all', subset=['last', 'email'])`, it returns a new DataFrame with rows removed where all values in the specified subset of columns are

missing.

Here's another example:

```
# Create a DataFrame with missing values
data = {'first': ['John', None, 'Jane'],
        'last': [None, None, None],
        'email': ['john@example.com', None, None]}

df = pd.DataFrame(data)

# Drop rows where both 'last' and 'email' columns are missing
df_cleaned = df.dropna(axis='index', how='all', subset=['last', 'email'])

print("DataFrame after dropping rows with missing 'last' and 'email' values:")
print(df_cleaned)
```

This code will remove the second and third rows because both the 'last' and 'email' columns have missing values in those rows.

[9]:
```
df.replace('NA', np.nan, inplace=True)
df.replace('Missing', np.nan, inplace=True)
df
```

[9]:

	first	last	email	age
0	Sakib	Anwar	SakibAnwar@email.com	100
1	Jane	Doe	JaneDoe@email.com	55
2	John	Doe	JohnDoe@email.com	63
3	Chris	Schafer	None	36
4	NaN	NaN	NaN	None
5	None	NaN	Anonymous@email.com	None
6	NaN	NaN	NaN	NaN

1. **Replacing 'NA' and 'Missing' Values with `np.nan`:**
   - After creating the DataFrame, `df.replace()` method is used to replace the string values `'NA'` and `'Missing'` with `np.nan`, which represents missing values in pandas.
   - This is done to standardize missing value representation in the DataFrame. By replacing these strings with `np.nan`, it ensures that all missing values are represented consistently as `np.nan` throughout the DataFrame.
2. **`inplace=True` argument:**
   - The `inplace=True` argument is used with the `replace()` method to modify the DataFrame `df` in place. When `inplace=True`, the modifications are applied directly to the DataFrame `df`, and it is updated without the need to assign the result back to `df`.

In summary, replaces string values `'NA'` and `'Missing'` with `np.nan` to represent missing values consistently, and updates `df` in place. This ensures that the DataFrame `df` has standardized missing value representation for further data analysis and processing.

[10]:
```
df.dropna(axis='index', how='any')
#This code removes rows containing any missing values (NaN) in the DataFrame␣
 ↪(`df`).
```

[10]:
```
   first   last                email  age
0  Sakib  Anwar  SakibAnwar@email.com  100
1   Jane    Doe     JaneDoe@email.com   55
2   John    Doe     JohnDoe@email.com   63
```

[11]:
```
df.dropna(axis='index', how='all', subset=['last', 'email'])
#This code removes rows from the DataFrame (`df`) where all values in the␣
 ↪'last' and 'email' columns are missing.
```

[11]:
```
   first     last                 email   age
0  Sakib    Anwar  SakibAnwar@email.com   100
1   Jane      Doe     JaneDoe@email.com    55
2   John      Doe     JohnDoe@email.com    63
3  Chris  Schafer                  None    36
5   None      NaN   Anonymous@email.com  None
```

[12]:
```
df.fillna(0)
```

[12]:
```
   first     last                 email  age
0  Sakib    Anwar  SakibAnwar@email.com  100
1   Jane      Doe     JaneDoe@email.com   55
2   John      Doe     JohnDoe@email.com   63
3  Chris  Schafer                     0   36
4      0        0                     0    0
5      0        0   Anonymous@email.com    0
6      0        0                     0    0
```

The `df.fillna(0)` method is used to fill missing (NaN) values in the DataFrame (`df`) with a specified value, in this case, 0. This means that wherever there is a missing value in the DataFrame, it will be replaced with 0. The original DataFrame (`df`) remains unchanged unless the `inplace=True` parameter is used.

[13]:
```
df.dtypes
```

[13]:
```
first    object
last     object
email    object
age      object
dtype: object
```

`df.dtypes` is an attribute in pandas used to display the data types of each column in the DataFrame (`df`). It returns a Series where the index is the column names and the values are the corresponding data types of each column.

Let's try to get mean age.

[14]:
```
df['age'].mean()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 df['age'].mean()

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪generic.py:11556, in NDFrame._add_numeric_operations.<locals>.mean(self, axis␣
 ↪skipna, numeric_only, **kwargs)
  11539 @doc(
  11540     _num_doc,
  11541     desc="Return the mean of the values over the requested axis.",
   (…)
  11554     **kwargs,
  11555 ):
> 11556     return NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪generic.py:11201, in NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)
  11194 def mean(
  11195     self,
  11196     axis: Axis | None = 0,
   (…)
  11199     **kwargs,
  11200 ) -> Series | float:
> 11201     return self._stat_function(
  11202         "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  11203     )

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪generic.py:11158, in NDFrame._stat_function(self, name, func, axis, skipna,␣
 ↪numeric_only, **kwargs)
  11154     nv.validate_stat_func((), kwargs, fname=name)
  11156 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 11158 return self._reduce(
  11159     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_onl
  11160 )

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪series.py:4670, in Series._reduce(self, op, name, axis, skipna, numeric_only,␣
 ↪filter_type, **kwds)
   4665     raise TypeError(
   4666         f"Series.{name} does not allow {kwd_name}={numeric_only} "
   4667         "with non-numeric dtypes."
   4668     )
```

```
   4669 with np.errstate(all="ignore"):
-> 4670     return op(delegate, skipna=skipna, **kwds)

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪nanops.py:96, in disallow.__call__.<locals>._f(*args, **kwargs)
     94 try:
     95     with np.errstate(invalid="ignore"):
---> 96         return f(*args, **kwargs)
     97 except ValueError as e:
     98     # we want to transform an object array
     99     # ValueError message to the more typical TypeError
    100     # e.g. this is normally a disallowed function on
    101     # object arrays that contain strings
    102     if is_object_dtype(args[0]):

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪nanops.py:158, in bottleneck_switch.__call__.<locals>.f(values, axis, skipna,
 ↪**kwds)
    156         result = alt(values, axis=axis, skipna=skipna, **kwds)
    157 else:
--> 158     result = alt(values, axis=axis, skipna=skipna, **kwds)
    160 return result

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪nanops.py:421, in _datetimelike_compat.<locals>.new_func(values, axis, skipna␣
 ↪mask, **kwargs)
    418 if datetimelike and mask is None:
    419     mask = isna(values)
--> 421 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    423 if datetimelike:
    424     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/pandas/core/
 ↪nanops.py:727, in nanmean(values, axis, skipna, mask)
    724     dtype_count = dtype
    726 count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
--> 727 the_sum = _ensure_numeric(values.sum(axis, dtype=dtype_sum))
    729 if axis is not None and getattr(the_sum, "ndim", False):
    730     count = cast(np.ndarray, count)

File ~/opt/anaconda3/envs/Decarbon/lib/python3.8/site-packages/numpy/core/
 ↪_methods.py:48, in _sum(a, axis, dtype, out, keepdims, initial, where)
     46 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     47          initial=_NoValue, where=True):
---> 48     return umr_sum(a, axis, dtype, out, keepdims, initial, where)

TypeError: can only concatenate str (not "int") to str
```

We ran into a problem. `TypeError: can only concatenate str (not "int") to str`.

We have to do something with how the nan values are stored. That is conver them into float. But first let's what type of data is `np.nan`.

[15]:
```
type(np.nan)
```

[15]:
```
float
```

[16]:
```
df['age'] = df['age'].astype(float)
```

The code `df['age'] = df['age'].astype(float)` is used to convert the data type of the 'age' column in the DataFrame (`df`) to float. This means that each value in the 'age' column will be converted to a floating-point number data type. This can be useful for mathematical operations or when you need to work with numerical data in the 'age' column.

The `.astype()` method in pandas is used to change the data type of a Series or DataFrame column to another specified data type. It allows you to convert the values in a column to a different data type, such as integer, float, string, etc., depending on your requirements.

Now we can use `mean()` method.

[17]:
```
df['age'].mean()
```

[17]:
```
63.5
```

Le'ts look at a real dataset and try to clean one column. We will import Stackoverflow developer survey data. If you look at the CSV file there is a natural index column : `ResponseId`. So we pass `index_col='ResponseId'` as an argument in `pd.read_csv`.

[18]:
```
df = pd.read_csv('survey_results_public.csv',index_col='ResponseId')
df
```

[18]:
```
                                                MainBranch  \
ResponseId
1                                            None of these
2                              I am a developer by profession
3           I am not primarily a developer, but I write co…
4                              I am a developer by profession
5                              I am a developer by profession
…                                                         …
73264                          I am a developer by profession
73265                          I am a developer by profession
73266       I am not primarily a developer, but I write co…
73267                          I am a developer by profession
73268       I used to be a developer by profession, but no…

                                                Employment  \
ResponseId
1                                                      NaN
```

```
2                                        Employed, full-time
3                                        Employed, full-time
4                                        Employed, full-time
5                                        Employed, full-time
…                                                       …
73264                                    Employed, full-time
73265                                    Employed, full-time
73266                                    Employed, full-time
73267                                    Employed, full-time
73268       Independent contractor, freelancer, or self-em…

                                     RemoteWork  \
ResponseId
1                                           NaN
2                                  Fully remote
3           Hybrid (some remote, some in-person)
4                                  Fully remote
5           Hybrid (some remote, some in-person)
…                                           …
73264                              Fully remote
73265                             Full in-person
73266       Hybrid (some remote, some in-person)
73267       Hybrid (some remote, some in-person)
73268                              Fully remote

                                          CodingActivities  \
ResponseId
1                                                    NaN
2                   Hobby;Contribute to open-source projects
3                                                  Hobby
4                               I don’t code outside of work
5                                                  Hobby
…                                                       …
73264                              Freelance/contract work
73265                                              Hobby
73266                          Hobby;School or academic work
73267                                              Hobby
73268       Hobby;Contribute to open-source projects;Boots…

                                                    EdLevel  \
ResponseId
1                                                       NaN
2                                                       NaN
3           Master’s degree (M.A., M.S., M.Eng., MBA, etc.)
4              Bachelor’s degree (B.A., B.S., B.Eng., etc.)
5              Bachelor’s degree (B.A., B.S., B.Eng., etc.)
…                                                       …
```

```
73264          Bachelor's degree (B.A., B.S., B.Eng., etc.)
73265       Master's degree (M.A., M.S., M.Eng., MBA, etc.)
73266          Bachelor's degree (B.A., B.S., B.Eng., etc.)
73267          Bachelor's degree (B.A., B.S., B.Eng., etc.)
73268          Bachelor's degree (B.A., B.S., B.Eng., etc.)

                                                    LearnCode  \
ResponseId
1                                                         NaN
2                                                         NaN
3           Books / Physical media;Friend or family member…
4           Books / Physical media;School (i.e., Universit…
5           Other online resources (e.g., videos, blogs, f…
…                                                         …
73264       Books / Physical media;Other online resources …
73265       Other online resources (e.g., videos, blogs, f…
73266       Books / Physical media;Other online resources …
73267              Books / Physical media;On the job training
73268       Books / Physical media;Friend or family member…

                                               LearnCodeOnline  \
ResponseId
1                                                         NaN
2                                                         NaN
3           Technical documentation;Blogs;Programming Game…
4                                                         NaN
5           Technical documentation;Blogs;Stack Overflow;O…
…                                                         …
73264       Technical documentation;Blogs;Written Tutorial…
73265       Technical documentation;Blogs;Written Tutorial…
73266       Technical documentation;Programming Games;Stac…
73267                                                     NaN
73268       Technical documentation;Blogs;Programming Game…

                        LearnCodeCoursesCert YearsCode YearsCodePro  …  \
ResponseId                                                           …
1                                        NaN       NaN          NaN  …
2                                        NaN       NaN          NaN  …
3                                        NaN        14            5  …
4                                        NaN        20           17  …
5                                        NaN         8            3  …
…                                          …         …            …  …
73264                                 Udemy          8            5  …
73265                 Coursera;Udemy;Udacity         6            5  …
73266       Udemy;Codecademy;Pluralsight;edX        42           33  …
73267                                    NaN        50           31  …
73268                    Udemy;Pluralsight        16            5  …
```

```
                 TimeSearching               TimeAnswering  Onboarding  \
ResponseId
1                          NaN                         NaN         NaN
2                          NaN                         NaN         NaN
3                          NaN                         NaN         NaN
4                          NaN                         NaN         NaN
5                          NaN                         NaN         NaN
…                          …                           …           …
73264       30-60 minutes a day  Less than 15 minutes a day  Just right
73265       15-30 minutes a day        60-120 minutes a day   Very long
73266       30-60 minutes a day        60-120 minutes a day  Just right
73267                      NaN                         NaN         NaN
73268                      NaN                         NaN         NaN

                                          ProfessionalTech TrueFalse_1  \
ResponseId
1                                                      NaN         NaN
2                                                      NaN         NaN
3                                                      NaN         NaN
4                                                      NaN         NaN
5                                                      NaN         NaN
…                                                      …           …
73264       DevOps function;Microservices;Developer portal…         Yes
73265                                        None of these          No
73266                                        None of these          No
73267                                                  NaN         NaN
73268                                                  NaN         NaN

           TrueFalse_2  TrueFalse_3           SurveyLength  \
ResponseId
1                  NaN          NaN                    NaN
2                  NaN          NaN               Too long
3                  NaN          NaN  Appropriate in length
4                  NaN          NaN  Appropriate in length
5                  NaN          NaN               Too long
…                  …            …                      …
73264              Yes          Yes               Too long
73265              Yes          Yes               Too long
73266               No           No  Appropriate in length
73267              NaN          NaN  Appropriate in length
73268              NaN          NaN  Appropriate in length

                         SurveyEase ConvertedCompYearly
ResponseId
1                               NaN                NaN
2                         Difficult                NaN
```

```
3           Neither easy nor difficult             40205.0
4                                 Easy            215232.0
5                                 Easy                 NaN
…                                 …                    …
73264                             Easy                 NaN
73265                             Easy                 NaN
73266                             Easy                 NaN
73267                             Easy                 NaN
73268                             Easy                 NaN

[73268 rows x 78 columns]
```

We are interested in cleaning the `YearsCode` Column. So lets look at the first 10 values.

[19]:
```
df['YearsCode'].head(10)
```

[19]:
```
ResponseId
1     NaN
2     NaN
3      14
4      20
5       8
6      15
7       3
8       1
9       6
10     37
Name: YearsCode, dtype: object
```

Now look at all the unique values. This will help us identify any odd values. We expect it to numerical. So let's see if we have any non-numerical values.

[20]:
```
df['YearsCode'].unique()
```

[20]:
```
array([nan, '14', '20', '8', '15', '3', '1', '6', '37', '5', '12', '22',
       '11', '4', '7', '13', '36', '2', '25', '10', '40', '16', '27',
       '24', '19', '9', '17', '18', '26', 'More than 50 years', '29',
       '30', '32', 'Less than 1 year', '48', '45', '38', '39', '28', '23',
       '43', '21', '41', '35', '50', '33', '31', '34', '46', '44', '42',
       '47', '49'], dtype=object)
```

As you can see ther couple of non-numerical values in this column: `'Less than 1 year'` and `'More than 50 years'`. We need to replace them with appropriate values.

[21]:
```
df['YearsCode'].replace('Less than 1 year', 0, inplace=True)
```

[22]:
```
df['YearsCode'].replace('More than 50 years',51, inplace=True)
```

Finally convert the values into float.

[23]:
```
df['YearsCode']=df['YearsCode'].astype(float)
```

[24]:
```
df['YearsCode'].median()
```

[24]:
```
9.0
```

[ ]:
```

```