AI4D LAB TRAINING

[Zephania Reuben (https://nsoma.me)](https://nsoma.me)

July 17, 2023

DATA SCIENCE | PANDAS

What is Pandas?

- **Pandas** is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.
- The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
- Data frames are tabular, meaning that they are based on rows and columns like you would see in a spreadsheet.
- pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user cansimply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets

Pandas Installation

- conda environment conda install pandas
- Installing from PyPI
  - python -m pip install pandas
- **Installing pandas on Linux**
  - In the following table, we will present some of the common Linux distributions package names for Matplotlib and the tools we can use to install the package:

Distribution	Package Name
Debian or Ubuntu (And other Debian derivatives)	`sudo apt-get install python3-pandas`
Fedora	`sudo dnf install python3-pandas`
Red hat	`sudo yum install python3-pandas`
Centos/RHEL	`sudo dnf install python3-pandas`

1.Understanding a pandas DataFrame

- a pandas DataFrame (in a Jupyter Notebook) appears to be nothing more than an ordinary table of data consisting of rows and columns. Hiding beneath the surface are the three components--the index, columns, and data (also known as values) that you must be aware of in order to maximize the DataFrame's full potential.
- Analyze the labeled anatomy of the DataFrame:
- **Note**
  - In this Notebook we will be using a **Titanic** dataset.A dataset about passengers in Titanic.

The variables that describe the passengers are:

- **PassengerId**: and id given to each traveller on the boat.
- **Pclass**: the passenger class. It has three possible values: 1,2,3.
- **The Name**: a word or set of words by which a person or thing is usually known.
- **The Sex**: males or females considered as separate groups.
- **The Age**: the number of years that someone has lived.
- **SibSp**: number of siblings and spouses traveling with the passenger.
- **Parch**: number of parents and children traveling with the passenger.
- **The ticket number**: a number (identifier) piece of paper that shows you have paid for a journey.
- **The ticket Fare**: amount paid for a ticket.
- **The cabin number**: a number for private room on a ship for a passenger.
- **The embarkation**: It has three possible values S,C,Q

- A DataFrame has two axes: a **vertical axis** (the index) and a **horizontal axis**(the columns). Pandas borrows convention from NumPy and uses the integers 0/1 as another way of referring to the vertical/horizontal axis.

In [2]:
```
#Load library
import pandas as pd
```

In [3]:
```
import seaborn as sns
```

In [4]:
```
df = sns.load_dataset('titanic')
```

In [11]:
```
df.head(10)
```

Out[11]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	
6	0	1	male	54.0	0	0	51.8625	S	First	man	
7	0	3	male	2.0	3	1	21.0750	S	Third	child	
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	
9	1	2	female	14.0	1	0	30.0708	C	Second	child	

In [5]:
```
#Create url

url = './Titanic.csv'
```

In [6]:
```
pd.read_clipboard()
```

Out[6]:

	base_entity_id	systolic	diastolic	gest_age	protein_in_urine
0	4g3h2a1b6e7f9d	120	80	32	negative
1	8f1c5b9e4h3a6g	130	85	28	positive
2	2d7e4h6b9f1c8a	115	75	34	negative
3	7c8h6e9f3d2b4a	140	90	30	positive
4	5b2a1c8g4f6e3h	125	80	29	negative
5	1d6e9a5h2c7b4f	135	85	31	positive
6	6f4b9h3g8a1d5e	120	75	33	negative
7	3g7f4h1b6a5c9e	130	85	27	positive
8	8h5d9a7b3f6g2c	115	80	32	negative
9	2e6f8g4c3h1a9b	140	90	30	positive
10	9a5e3d7c6f8b2h	130	85	29	negative
11	4g3h2a1b6e7f9d	120	80	31	positive
12	8f1c5b9e4h3a6g	135	90	28	negative
13	2d7e4h6b9f1c8a	125	85	33	positive
14	7c8h6e9f3d2b4a	115	75	30	negative
15	5b2a1c8g4f6e3h	140	85	32	positive
16	1d6e9a5h2c7b4f	130	85	33	negative
17	6f4b9h3g8a1d5e	120	80	30	positive
18	3g7f4h1b6a5c9e	135	90	32	negative
19	8h5d9a7b3f6g2c	125	85	27	positive
20	2e6f8g4c3h1a9b	115	75	31	negative
21	9a5e3d7c6f8b2h	140	85	29	positive
22	4g3h2a1b6e7f9d	130	80	34	negative
23	8f1c5b9e4h3a6g	120	85	31	positive

In [7]:

```
# Load data as a DataFrame
dataframe = pd.read_csv(url)
```

In [10]:
```
dataframe.head(10)
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.45
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.07
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07

Things to notice in this DataFrame

- First, in a data frame each row corresponds to one observation (e.g., a passenger) and each column corresponds to one feature (gender, age, etc.). For example, bylooking at the first observation we can see that **Heikkinen, Miss. Laina** stayed in first class, was 26 years old, was female, and survived the disaster.
- Second, each column contains a name (e.g., Name, PClass, Age) and each rowcontains an index number (e.g., 0 for the lucky Miss Elisabeth Walton Allen). We will use these to select and manipulate observations and features.

2.Creating a DataFrame

- First method :
  - Create a dataframe and add columns independently.

In [14]:
```
df = pd.DataFrame()
```

In [15]:
```
df['names'] = ['John','Rose','Jack']
```

In [16]:
```
df['age'] = [15,19,46]
```

In [17]:
```
df['country'] = ['Kenya','Uganda','Malawi']
```

In [18]:
```
df
```

Out[18]:

	names	age	country
0	John	15	Kenya
1	Rose	19	Uganda
2	Jack	46	Malawi

- Second method :
  - Create a dataframe and add columns at the same time.

In [19]:
```
#Load library
import pandas as pd

# Create a DataFrame
df = pd.DataFrame(columns=['Name','Age','Country'],
                  data=[
                       ['John',19,'Kenya'],
                       ['Rebecca',16,'Uganda'],
                       ['Lisa',19,'Rwanda'],
                       ['Godfrey',19,'Tanzania'],
                       ['Vivian',19,'Burundi']
                      ])

#show DataFrame
df
```

Out[19]:

	Name	Age	Country
0	John	19	Kenya
1	Rebecca	16	Uganda
2	Lisa	19	Rwanda
3	Godfrey	19	Tanzania
4	Vivian	19	Burundi

In [23]:
```
df.iloc[0]
```

Out[23]:
```
Name          John
Age             19
Country      Kenya
Name: 0, dtype: object
```

In [20]:
```
type(df)
```

Out[20]:
```
pandas.core.frame.DataFrame
```

3.Creating a Series

In [21]:
```
#Load library
import pandas as pd

#Create a Series
series = pd.Series(index=['Name','Age','Country'],data=['John',19,'
Uganda'])

#show series
series
```

Out[21]:
```
Name          John
Age             19
Country     Uganda
dtype: object
```

In [22]:
```
type(series)
```

Out[22]:
```
pandas.core.series.Series
```

A series can be used to create a DataFrame as follows

In [28]:
```
df = pd.DataFrame()
```

In [25]:
```
df
```

Out[25]:

	0
Name	John
Age	19
Country	Uganda

4.Describing a DataFrame

- Describing a DataFrame involve looking at its short summary of descriptive statistical measures.

In [30]:
```
dataframe.describe()
```

Out[30]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.0000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.2042
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.6934
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.0000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.9104
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.4542
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.0000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.3292

- We can also take a look at the number of row and colums

In [31]:
```
dataframe.shape
```

Out[31]:
```
(891, 12)
```

In [32]:
```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

- DataFrame has 891 rows(instances/samples) and 12 colums(features)

5.Navigating DataFrames

- You need to select individual data or slices of a DataFrame
  - **loc**
    - is useful when the index of the DataFrame is a label (e.g., a string).
  - **iloc**
    - works by looking for the position in the DataFrame. For example, iloc[0] will return the first row regardless of whether the index is an integer or a label.

In [33]:
```
dataframe.head()
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [34]:
```
# Select three rows
dataframe.iloc[1:4] # also dataframe.iloc[:4]
```

Out[34]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100

- DataFrames do not need to be numerically indexed. We can set the index of a DataFrame to any value where the value is unique to each row. For example, we can set the index to be passenger names and then select rows using a name:

In [35]:
```
#set index

dataframe = dataframe.set_index(dataframe['Name'])
```

In [36]:
```
dataframe.head()
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticke
Name									
Braund, Mr. Owen Harris	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 2117
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 1759
Heikkinen, Miss. Laina	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O 310128
Futrelle, Mrs. Jacques Heath (Lily May Peel)	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	11380
Allen, Mr. William Henry	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	37345

In [37]:
```
#use index to slice and show row
dataframe.loc['Heikkinen, Miss. Laina']
```

Out[37]:
```
PassengerId                          3
Survived                             1
Pclass                               3
Name            Heikkinen, Miss. Laina
Sex                             female
Age                               26.0
SibSp                                0
Parch                                0
Ticket                STON/O2. 3101282
Fare                             7.925
Cabin                              NaN
Embarked                             S
Name: Heikkinen, Miss. Laina, dtype: object
```

6.Selecting Rows Based on Conditionals

- Suppose we want to select all women in Titanic

In [40]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Show top two rows where column 'sex' is 'female'
dataframe[ dataframe['Sex']=='female' ].head(2)
```

Out[40]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925

- Multiple conditions are easy as well. For example, here we select all the rows where the passenger is a female 65 or older:

In [41]:
```
# Show top two rows where column 'sex' is 'female' and 'age' >=27
dataframe[(dataframe['Sex'] == 'female') & (dataframe['Age'] >= 27)
].head(10)
```

Out[41]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333
				Bonnell,						

11	12	1	1	Miss. Elizabeth	female	58.0	0	0	113783	26.5500
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875
40	41	0	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	female	40.0	1	0	7546	9.4750
41	42	0	2	Turpin, Mrs. William John Robert (Dorothy Ann ...	female	27.0	1	0	11668	21.0000
52	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292

7.Replacing Values

- pandas’ replace is an easy way to find and replace values. For example, we can replace any instance of "female" in the Sex column with "Woman":

In [42]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Replace values, show two rows
dataframe[dataframe['Sex']=='female'].replace("female", "F").head(1
0)
```

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	F	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	F	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	F	35.0	1	0	113803	53.100
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	F	27.0	0	2	347742	11.133
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	F	14.0	1	0	237736	30.070
10	11	1	3	Sandstrom, Miss. Marguerite Rut	F	4.0	1	1	PP 9549	16.700
11	12	1	1	Bonnell, Miss. Elizabeth	F	58.0	0	0	113783	26.550
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	F	14.0	0	0	350406	7.854
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	F	55.0	0	0	248706	16.000
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	F	31.0	1	0	345763	18.000

- We can also replace multiple values at the same time:

In [43]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Replace "female" and "male with "Woman" and "Man"
dataframe[(dataframe['Sex'] == 'female') | (dataframe['Sex'] == 'ma
le')].replace(["female", "male"], [0, 1]).head(5)
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500

In [44]:
```
dataframe['Ticket'][1].split()[1]
```

Out[44]:
```
'17599'
```

In [45]:
```
dataframe.Ticket[1].split()
```

Out[45]:
```
['PC', '17599']
```

In [ ]:
```
#Passenger ID
```

8.Renaming Columns

- Rename columns using the rename method:

In [67]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Rename column, show two rows
dataframe.rename(columns={'Pclass': 'p_class'}).head(2)
```

Out[67]:

	PassengerId	Survived	p_class	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833

- Notice that the rename method can accept a dictionary as a parameter. We can use the dictionary to change multiple column names at once:

In [47]:
```
# Rename columns, show two rows
dataframe.rename(columns={'Pclass': 'p_class', 'Sex': 'sex'}).head(
2)
```

Out[47]:

	PassengerId	Survived	p_class	Name	sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833

9.Finding the Minimum, Maximum, Sum,Average, and Count

In [48]:
```
# Load library
import pandas as pd
# Create URL
url = './Titanic.csv'
# Load data
dataframe = pd.read_csv(url)
# Calculate statistics
print('Maximum:', dataframe['Age'].max())
print('Minimum:', dataframe['Age'].min())
print('Mean:', dataframe['Age'].mean())
print('Sum:', dataframe['Age'].sum())
print('Count:', dataframe['Age'].count())
```

```
Maximum: 80.0
Minimum: 0.42
Mean: 29.69911764705882
Sum: 21205.17
Count: 714
```

In [53]:
```
dataframe['Age'].agg('median')
```

Out[53]:
```
28.0
```

In [54]:
```
dataframe['Age'].agg('mode')
```

Out[54]:
```
0    24.0
Name: Age, dtype: float64
```

10.Finding Unique Values

- Use unique to view an array of all unique values in a column:

```
In [68]: # Load library
         import pandas as pd

         # Create URL
         url = './Titanic.csv'

         # Load data
         dataframe = pd.read_csv(url)

         # Select unique values
         dataframe['Pclass'].unique()
```

```
Out[68]: array([3, 1, 2])
```

```
In [51]: dataframe['Pclass'].value_counts()
```

```
Out[51]: Pclass
         3    491
         1    216
         2    184
         Name: count, dtype: int64
```

- Alternatively, value_counts will display all unique values with the number of times each value appears:

```
In [52]: dataframe['Sex'].value_counts()
```

```
Out[52]: Sex
         male      577
         female    314
         Name: count, dtype: int64
```

11.Handling Missing Values

- isnull and notnull return booleans indicating whether a value is missing:

In [69]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

## Select missing values, show two rows
dataframe[dataframe['Age'].isnull()].head(2)
```

Out[69]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000

In [59]:
```
(dataframe.isna().sum()/dataframe.shape[0])*100
```

Out[59]:
```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [72]:
```
dataframe['Age_filled'] = dataframe['Age'].fillna(value=dataframe['
Age'].mean())
```

In [73]:
```
dataframe['Age_filled'].isna().sum()
```

Out[73]:
```
0
```

In [ ]:
```
dataframe.shape
```

In [ ]:
```
```

12.Deleting a Column

- The best way to delete a column is to use drop with the parameter axis=1 (i.e., the column axis):

In [74]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Delete column
dataframe = dataframe.drop('Age', axis=1).head(2)
```

In [75]:
```
dataframe.columns
```

Out[75]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'SibSp'
, 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

- You can also use a list of column names as the main argument to drop multiple columns at once:

In [76]:
```
# Drop columns
dataframe.drop(['Pclass', 'Sex'], axis=1)
```

Out[76]:

	PassengerId	Survived	Name	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	0	PC 17599	71.2833	C85	C

In [ ]:
```
[dataframe['PassengerId'] != 2].head()
```

13.Deleting a Row

- Use a boolean condition to create a new DataFrame excluding the rows you want to delete:

In [78]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Delete rows, show first two rows of output
dataframe[dataframe['PassengerId'] != 3].head()
```

Out[78]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583

14.Dropping Duplicate Rows

- Use drop_duplicates, but be mindful of the parameters:

In [79]:
```python
# Load library
import pandas as pd
# Create URL
url = './Titanic.csv'
# Load data
dataframe = pd.read_csv(url)
```

In [80]:
```python
dataframe.duplicated().sum()
```

Out[80]:
```
0
```

In [81]:
```python
# Drop duplicates, show first two rows of output
dataframe.drop_duplicates(keep='first').head(2)
```

Out[81]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833

15.Grouping Rows by Values

- groupby is one of the most powerful features in pandas:

In [83]:
```
# Load library
import pandas as pd

# Create URL
url = './Titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Group rows by the values of the column 'Sex', calculate mean
# of each group
dataframe[['Sex','Age']].groupby('Sex').mean().rename(columns={'Age
':'mean_age'})
```

Out[83]:

Sex	mean_age
female	27.915709
male	30.726645

15.Concatenating DataFrames

- Use concat with axis=0 to concatenate along the row axis:

In [99]:
```
# Load library
import pandas as pd

# Create DataFrame
data_a = {'id': ['1', '2', '3'],
'first': ['Alex', 'Amy', 'Allen']}
dataframe_a = pd.DataFrame(data = data_a, columns = ['id', 'first']
)
```

In [100]:
```
dataframe_a
```

Out[100]:

	id	first
0	1	Alex
1	2	Amy
2	3	Allen

In [108]:
```
# Create DataFrame
data_b = {'id': ['1', '2', '3','4'],'last': ['Anderson', 'Ackerman'
, 'Ali','Juma']}
dataframe_b = pd.DataFrame(data = data_b, columns = ['id', 'last'])
```

In [109]:
```
dataframe_b
```

Out[109]:

	id	last
0	1	Anderson
1	2	Ackerman
2	3	Ali
3	4	Juma

In [110]:
```
pd.merge(dataframe_a,dataframe_b, how='outer',on='id')
```

Out[110]:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
3	4	NaN	Juma

In [90]:
```
pd.concat([dataframe_a,dataframe_b], axis=0)
```

Out[90]:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
0	4	Billy	Bonder
1	5	Brian	Black
2	6	Bran	Balwner

In [ ]:
```

```

Referencies

- Machine Learning with Python Cookbook,Chris Albon, O’Reilly Media, Inc,2018