Bootcamp Training

Python

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

Authorized & published by Summitworks Technologies Inc

Agenda: Day5

✓Introduction to Machine Learning

What is Machine Learning

- Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.
- The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.
- Machine learning - is the science of creating algorithms and program which learn on their own. Once designed, they do not need a human to become better. Some of the common applications of machine learning include following: Web Search, spam filters, recommender systems, ad placement, credit scoring, fraud detection, stock trading, computer vision and drug design. An easy way to understand is this - it is humanly impossible to create models for every possible search or spam, so you make the machine intelligent enough to learn by itself. When you automate the later part of data mining - it is known as machine learning.

Examples of Machine Learning Problems

- Machine Learning problems are abound. They make up core or difficult parts of the software you use on the web or on your desktop everyday. Think of the “do you want to follow” suggestions on twitter and the speech understanding in Apple’s Siri.

Examples of Machine Learning Problems

- **Spam Detection**: Given email in an inbox, identify those email messages that are spam and those that are not. Having a model of this problem would allow a program to leave non-spam emails in the inbox and move spam emails to a spam folder. We should all be familiar with this example.
- **Credit Card Fraud Detection**: Given credit card transactions for a customer in a month, identify those transactions that were made by the customer and those that were not. A program with a model of this decision could refund those transactions that were fraudulent.
- **Digit Recognition**: Given a zip codes hand written on envelops, identify the digit for each hand written character. A model of this problem would allow a computer program to read and understand handwritten zip codes and sort envelops by geographic region

Examples of Machine Learning Problems

- **Speech Understanding**: Given an utterance from a user, identify the specific request made by the user. A model of this problem would allow a program to understand and make an attempt to fulfil that request. The iPhone with Siri has this capability.
- **Face Detection**: Given a digital photo album of many hundreds of digital photographs, identify those photos that include a given person. A model of this decision process would allow a program to organize photos by person. Some cameras and software like iPhoto has this capability.

Examples of Machine Learning Problems

- **Product Recommendation**: Given a purchase history for a customer and a large inventory of products, identify those products in which that customer will be interested and likely to purchase. A model of this decision process would allow a program to make recommendations to a customer and motivate product purchases. Amazon has this capability. Also think of Facebook, GooglePlus and Facebook that recommend users to connect with you after you sign-up.
- **Medical Diagnosis**: Given the symptoms exhibited in a patient and a database of anonymized patient records, predict whether the patient is likely to have an illness. A model of this decision problem could be used by a program to provide decision support to medical professionals.
- **Stock Trading**: Given the current and past price movements for a stock, determine whether the stock should be bought, held or sold. A model of this decision problem could provide decision support to financial analysts.

Examples of Machine Learning Problems

- **Online Search:** Perhaps the most famous use of machine learning, Google and its competitors are constantly improving what the search engine understands. Every time you execute a search on Google, the program watches how you respond to the results. If you click the top result and stay on that web page, we can assume you got the information you were looking for and the search was a success. If, on the other hand, you click to the second page of results, or type in a new search string without clicking any of the results, we can surmise that the search engine didn't serve up the results you wanted — and the program can learn from that mistake to deliver a better result in the future.

- **Numpy**. Adds Python support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

- **SciPy** is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

- **Pandas.** Software library written for data manipulation and analysis in Python. Offers data structures and operations for manipulating numerical tables and time series.

Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Numpy

```
import numpy as np
L = [1,2,3]
A = np.array(L)
print(A)
print(L)
```

```
[1 2 3]
[1, 2, 3]
```

```
for e in L:
    print(e)
```

```
1
2
3
```

```
for e in A:
    print(e)
```

```
1
2
3
```

```
L.append(4)
L
```

```
[1, 2, 3, 4]
```

```
A.append(4)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-25-7b48164c1d55> in <module>()
----> 1 A.append(4)

AttributeError: 'numpy.ndarray' object has no attribute 'append'
```

```
L = L + [5,6]  # its giving a new list with all elements concatinated with given list
```

```
L
```

```
[1, 2, 3, 5, 6]
```

Numpy

```
A = A+[5,6]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-28-e574a2111219> in <module>()
----> 1 A = A+[5,6]

ValueError: operands could not be broadcast together with shapes (3,) (2,)
```

```
L2 = []
for e in L:
    L2.append(e+e)
```

```
L2
```

```
[2, 4, 6, 10, 12]
```

```
L+L # elements will be appended
```

```
[1, 2, 3, 5, 6, 1, 2, 3, 5, 6]
```

```
A
```

```
array([1, 2, 3])
```

Numpy

```
A+A  # will add each element
```

```
array([2, 4, 6])
```

```
2*L
```

```
[1, 2, 3, 5, 6, 1, 2, 3, 5, 6]
```

```
2*A # another way to to add elements
```

```
array([2, 4, 6])
```

```
A
```

```
array([1, 2, 3])
```

```
A**2 # most operations are element wise in numpy
```

```
array([1, 4, 9])
```

```
L**2 # mostly list cant be on element
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-38-9c714a1bea7b> in <module>()
----> 1 L**2
```

Numpy

```
np.sqrt(A)
```

```
array([ 1.        ,  1.41421356,  1.73205081])
```

```
np.log(A)
```

```
array([ 0.        ,  0.69314718,  1.09861229])
```

```
np.exp(A)
```

```
array([  2.71828183,   7.3890561 ,  20.08553692])
```

all above operations if you want to do in list, we have to use for loop,but numpy will do without for loop. for loops in python are very slow, but numpy will be very fast , if you want to do operations on list of data.

```
a = np.arange(15).reshape(3, 5)
a
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

Numpy

```
c = np.array( [ [1,2], [3,4] ], dtype=complex )
c
```

```
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

```
c = np.arange(24).reshape(2,3,4)          # 3d array
c
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

```
print(np.arange(10000).reshape(100,100))
```

```
[[   0    1    2 ...,   97   98   99]
 [ 100  101  102 ...,  197  198  199]
 [ 200  201  202 ...,  297  298  299]
 ...,
 [9700 9701 9702 ..., 9797 9798 9799]
 [9800 9801 9802 ..., 9897 9898 9899]
 [9900 9901 9902 ..., 9997 9998 9999]]
```

Scipy

SciPy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the SciPy library is that, it should work with NumPy arrays. It provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization

Numpy VS SciPy

Numpy:

- Numpy is written in C and use for mathematical or numeric calculation.
- It is faster than other Python Libraries
- Numpy is the most useful library for Data Science to perform basic calculations.
- Numpy contains nothing but array data type which performs the most basic operation like sorting, shaping, indexing, etc.

SciPy:

- SciPy is built in top of the NumPy
- SciPy is a fully-featured version of Linear Algebra while Numpy contains only a few features.
- Most new Data Science features are available in Scipy rather than Numpy.

Scipy: Cubic Root Function

Cubic Root function finds the cube root of values.

Syntax:

scipy.special.cbrt(x)
Example:

```
from scipy.special import cbrt
#Find cubic root of 27 & 64 using cbrt() function
cb = cbrt([27, 64])
#print value of cb
print(cb)
```

Output: array([3., 4.])

Scipy: Exponential

Exponential Function:
Exponential function computes the 10**x element-wise.

Example:

```
from scipy.special import exp10
#define exp10 function and pass value in its
exp = exp10([1,10])
print(exp)
```

Output: [1.e+01 1.e+10]

Scipy: Numerical Integration

SciPy provides functionality to integrate function with numerical integration.
scipy.integrate library has single integration, double, triple, multiple, Gaussian quadrate, Romberg, Trapezoidal and Simpson's rules.
Example: Now take an example of Single Integration

$$\int_a^b f(x)dx$$

Here a is the upper limit and b is the lower limit

Scipy: Numerical Integration

```
from scipy import integrate
# take f(x) function as f
f = lambda x : x**2
#single integration with a = 0 & b = 1
integration = integrate.quad(f, 0 , 1)
print(integration)
```

Output:

(0.33333333333333337, 3.700743415417189e-15)

Here function returns two values, in which the first value is integration and second value is estimated error in integral.

matplotlib

Matplotlib is the most used Python package for 2D-graphics. It provides both a quick way to visualize data from Python and publication-quality figures in many formats. We are going to explore matplotlib in interactive mode covering most common cases.

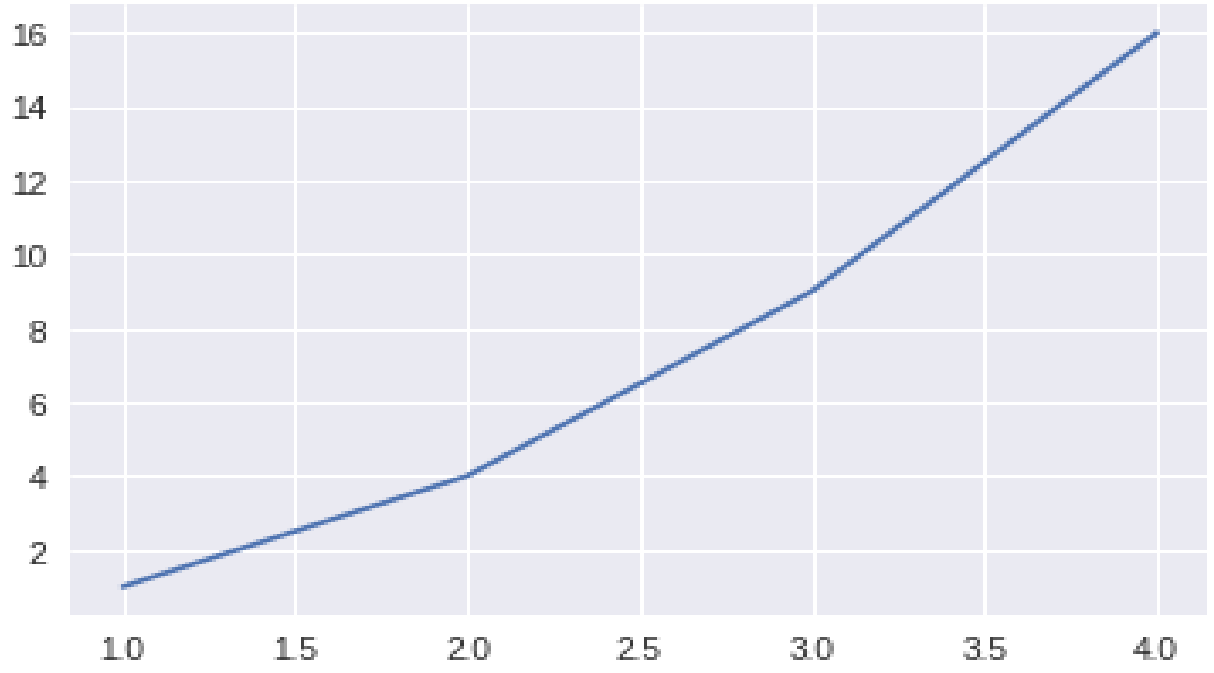
matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axe

Matplotlib: Make a simple plot

Here we import Matplotlib's Pyplot module and Numpy library as most of the data that we will be working with will be in the form of arrays only.

```
import matplotlib.pyplot as plt
import numpy as np

plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```

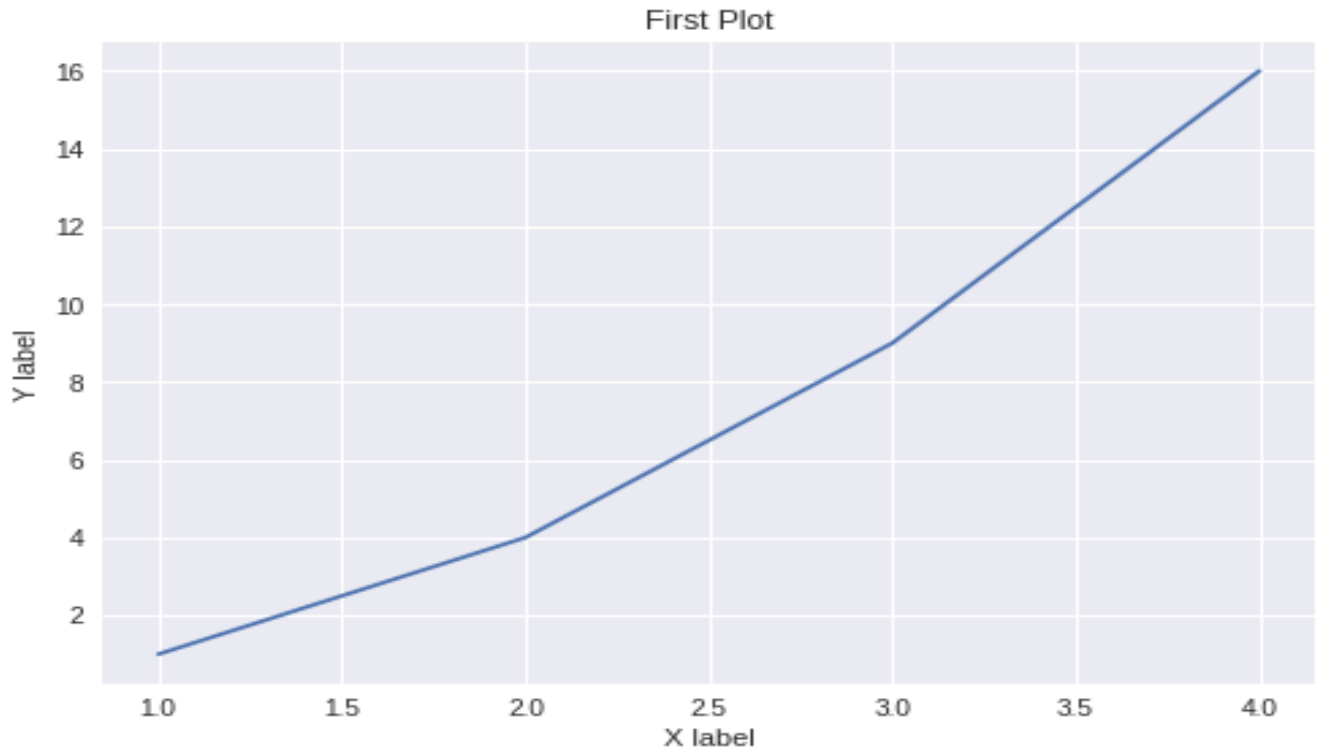


We pass two arrays as our input arguments to Pyplot's plot() method and use show() method to invoke the required plot. Here note that the first array appears on the x-axis and second array appears on the y-axis of the plot.

Matplotlib: Make a simple plot

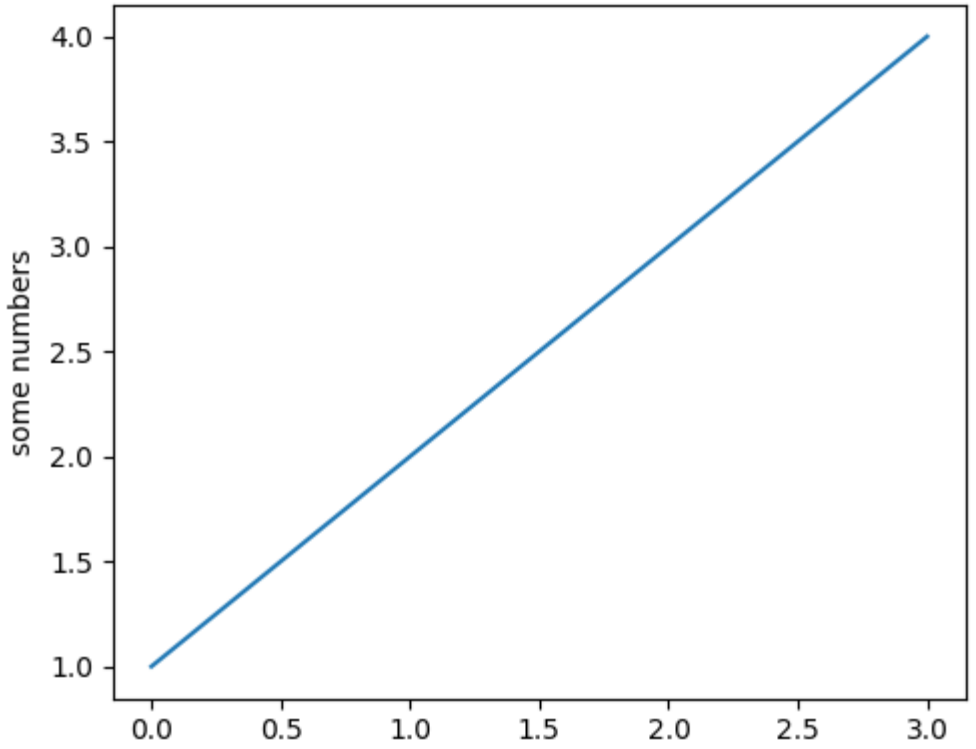
Now that our first plot is ready, let us add the title, and name x-axis and y-axis using methods title(), xlabel() and ylabel() respectively.

```
plt.plot([1,2,3,4],[1,4,9,16])
plt.title("First Plot")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```



Matplotlib: Make a simple plot

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



You may be wondering why the x-axis ranges from 0-3 and the y-axis from 1-4. If you provide a single list or array to the plot() command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you. Since python ranges start with 0, the default x vector has the same length as y but starts with 0. Hence the x data are [0,1,2,3].

Matplotlib: Make a simple plot

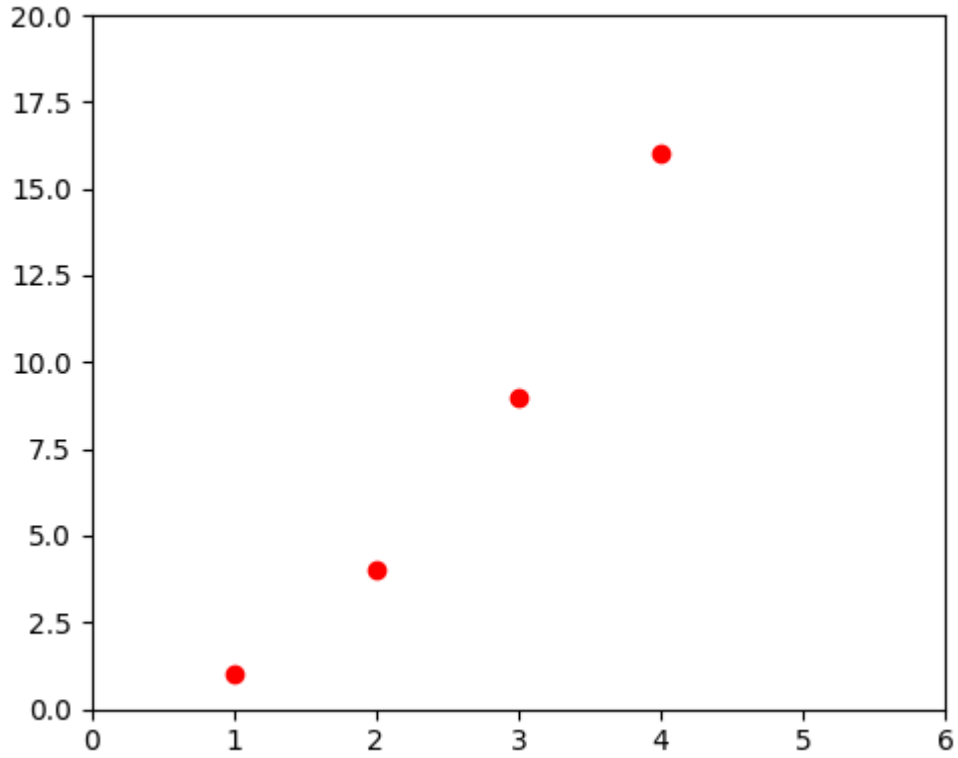
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```

For every x, y pair of arguments, there is an optional third argument which is the format string that indicates the color and line type of the plot. The letters and symbols of the format string are from MATLAB, and you concatenate a color string with a line style string. The default format string is ‘b-‘, which is a solid blue line. For example, to plot the above with red circles, you would issue

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

Matplotlib: Make a simple plot

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics

Pandas: Pandas DataFrames

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas: Pandas DataFrames

```
dict = {"country": ["Brazil", "Russia", "India", "China",
"South Africa"], "capital": ["Brasilia", "Moscow",
"New Dehli", "Beijing", "Pretoria"], "area": [8.516,
17.10, 3.286, 9.597, 1.221], "population": [200.4,
143.5, 1252, 1357, 52.98] }

import pandas as pd

brics = pd.DataFrame(dict)
print(brics)
```

area	capital	country	population	0
8.516	**Brasilia**	Brazil	200.401	17.100
Moscow		Russia	143.502	3.286
New Dehli		India	1252.003	9.597
Beijing		China	1357.004	1.221
Pretoria		South Africa	52.98	

Pandas: Querying

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(10, 2), columns=list('ab'))
>>> df
          a         b
0  1.518318 -0.670897
1 -0.374006 -0.927971
2 -0.150112  0.056623
3  1.061960  0.585305
4 -1.551948 -0.699413
5  1.560866 -2.189432
6  0.488731  0.831204
7  0.118696  0.765931
8 -0.969283  0.572119
9 -1.059981 -0.098627
>>> df.query('a > b')
          a         b
0  1.518318 -0.670897
1 -0.374006 -0.927971
3  1.061960  0.585305
5  1.560866 -2.189432
```

Pandas: Missing Data In pandas Dataframes

```
import pandas as pd
import numpy as np
raw_data = {'first_name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'],
        'age': [42, np.nan, 36, 24, 73],
        'sex': ['m', np.nan, 'f', 'm', 'f'],
        'preTestScore': [4, np.nan, np.nan, 2, 3],
        'postTestScore': [25, np.nan, np.nan, 62, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'sex', 'preTestScore', 'postTestScore'])
df
```

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	NaN	NaN
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

Pandas: Drop missing observations

```
df_no_missing = df.dropna()
df_no_missing
```

first_name	last_name	age	sex	preTestScore	postTestScore	
0	Jason	Miller	42.0	m	4.0	25.0
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

Any Queries