EI320A(3) 深度學習使用 Python

Instructors

Tipajin Thaipisutikul (t.greentip@gmail.com)

Prof. Huang-Chia Shih (hcshih@Saturn.yzu.edu.tw)

Course Syllabus

Evaluation Criteria

Tasks	Percentage
In Class Hands-on	60
Project Proposal Presentation	10
Project Final Presentation	30
Bonus (In Class Participation)	5
Total	110/100

Week	Date	Content	Note	Total
1	2/26	Welcome to the course	Download & Install Anaconda Homework (1)	1
2	3/5	Crash Course of Python, Numpy, Pandas, and Matplotlib	In class hands-on (4)	5
3	3/12	Get to know about Data ML:Classification Models	In class hands-on (5)	10
4	3/19	ML:Regression Models	In class hands-on (5)	15
5	3/26	ML:Clustering /Apriori Models	In class hands-on (5)	20
6	4/2	Holiday		
7	4/9	Introduction to Deep Learning (ANN)	In class hands-on (5)	25
8	4/16	Convolutional Neural Network (CNN)	In class hands-on (5)	30
9	4/23	Convolutional Neural Network (CNN)	In class hands-on (5)	35
10	4/30	Recurrent Neural Network (RNN)	In class hands-on (5)	40
11	5/7	Recurrent Neural Network (RNN)	In class hands-on (5)	45
12	5/14	Project Proposal Presentation	Proposal Presentation (10)	55
13	5/21	Time series with DNN, CNN, RNN	In class hands-on (5)	60
14	5/28	Attention Neural Network	In class hands-on (5)	65
15	6/4	Generative Adversarial Network (GAN)	In class hands-on (5)	70
16	6/11	Reinforcement Learning (RL)	In class hands-on (5)	75
17	6/18	Final Project Presentation	Final Presentation (30)	105

Bonus: 5 For class participation.

Preface

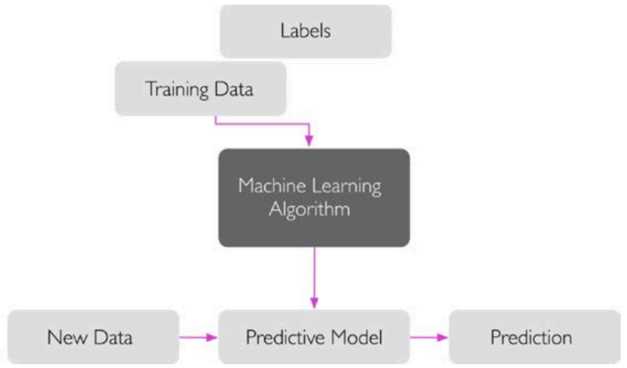
ARTIFICIAL INTELLIGENCE

Science thae enables computers to mimic human imtelligence.

Subfields: Machine Learning, robotics, and computer vision

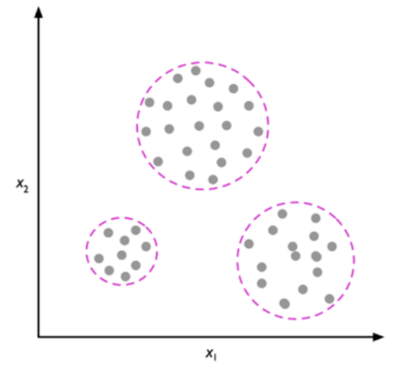
MACHINE LEARNING

Subset of AI that enable maches to improve at tasks with experience

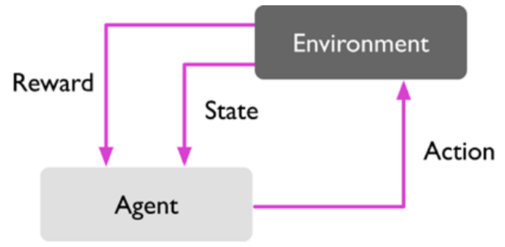


The three different types of machine learning

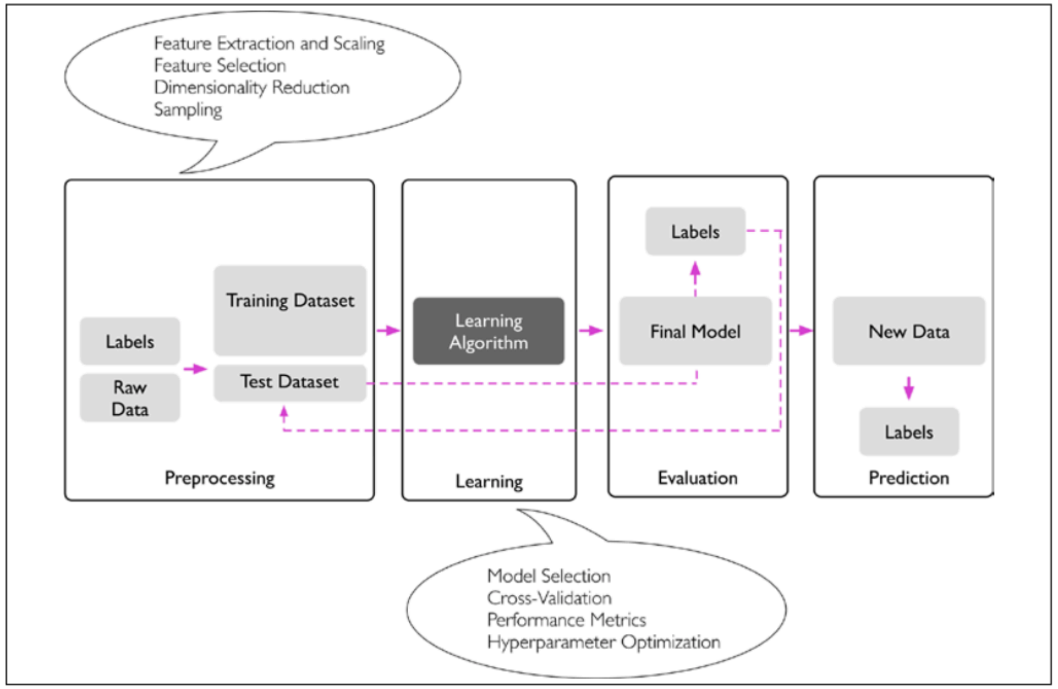
Supervised Learning	> Labeled data > Direct feedback > Predict outcome/future
Unsupervised Learning	> No labels > No feedback > Find hidden structure in data
Reinforcement Learning	> Decision process > Reward system > Learn series of actions



- Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned
- Deep learning structures algorithms in layers to create an artificial "neural network" that can learn and make intelligent decisions on its own
- Deep learning is a subfield of machine learning. While both fall under the broad category of artificial intelligence, deep learning is usually what's behind the most human-like artificial intelligence



A Roadmap for building ML Systems



A predictive modeling machine learning project can be broken down into **6 top-level tasks:**

1. **Define Problem**: Investigate and characterize the problem in order to better understand the goals of the project.
2. **Analyze Data**: Use **descriptive statistics** and **visualization** to better understand the data you have available.
3. **Prepare Data**: Use **data transforms** in order to better expose the structure of the prediction problem to modeling algorithms.
4. **Evaluate Algorithms:** Design a test harness to evaluate a number of standard algorithms on the data and select the top few to investigate further.
5. **Improve Results:** Use algorithm tuning and ensemble methods to get the most out of well-performing algorithms on your data.
6. **Present Results:** Finalize the model, make predictions and present results.

A Roadmap for building ML Systems

	ML Lesson
	Lesson 1: Python Ecosystem for Machine Learning. Lesson 2: Python and SciPy Crash Course. (Numpy, Pandas, Matplotlib)
Define Problem & Analyze Data	Lesson 3: Understand Data With Descriptive Statistics. Lesson 4: Understand Data With Visualization.
Prepare Data	Lesson 5: Pre-Process Data.
Evaluate Algorithms	Lesson 6: Resampling Methods. Lesson 7: Algorithm Evaluation Metrics. Lesson 8: Spot-Check Classification Algorithms. Lesson 9: Spot-Check Regression Algorithms. Lesson 10: Model Selection.
Improve Results	Lesson 11: Ensemble Methods. Lesson 12: Algorithm Parameter Tuning.
Present Results	Lesson 13: Model Finalization.

1. Python Ecosystem for Machine Learning

Python is a general purpose interpreted programming language. It is easy to learn and use primarily because the language focuses on readability. The philosophy of Python is captured in the Zen of Python which includes phrases like:

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
```

It is a popular language in general, consistently appearing in the top 10 programming languages in surveys on StackOverflow. It's a dynamic language and very suited to interactive development and quick prototyping with the power to support the development of large applications. It is also widely used for machine learning and data science because of the excellent library support and because it is a general purpose programming language (unlike R or Matlab).

1. Python Ecosystem for Machine Learning

SciPy

is an ecosystem of Python libraries for mathematics, science and engineering.
It is an add-on to Python that you will need for machine learning.
The SciPy ecosystem is comprised of the following core modules relevant to machine learning:

1. **NumPy**: A foundation for SciPy that allows you to efficiently work with data in arrays.
2. **Matplotlib**: Allows you to create 2D charts and plots from data.
3. **Pandas**: Tools and data structures to organize and analyze your data.

To be effective at machine learning in Python you must install and become familiar with SciPy. Specifically:

- You will prepare your data as **NumPy arrays for modeling in machine learning algorithms.**
- You will use Matplotlib (and wrappers of Matplotlib in other frameworks) **to create plots and charts of your data.**
- You will use Pandas to **load, explore, and better understand your data.**

scikit-learn library

is how you can develop and practice machine learning in Python.
It is built upon and requires the SciPy ecosystem. The name scikit suggests that it is a SciPy plug-in or toolkit.
The focus of **the library is machine learning algorithms for classification, regression, clustering and more.**
It also provides **tools for related tasks such as evaluating models, tuning parameters and pre-processing data.**

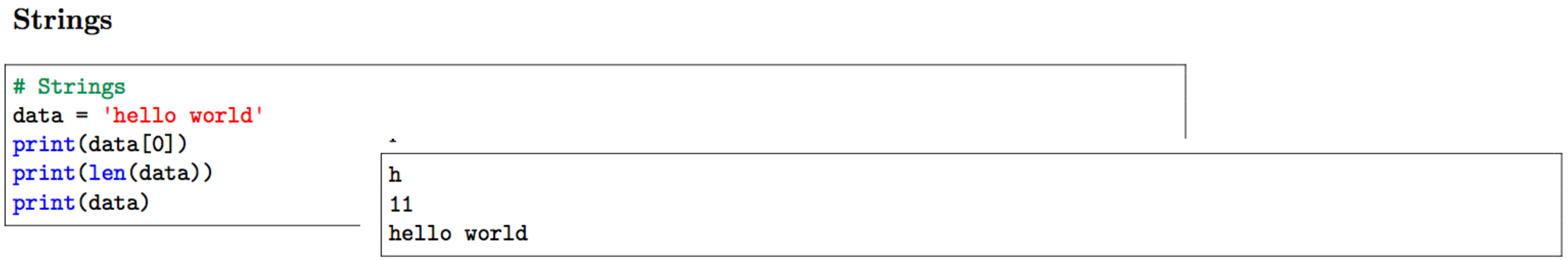
2. Crash Course in Python and SciPy

When getting started in **Python** you need to know a few key details about the language syntax to be able to **read and understand Python code.**
This includes:

- Assignment.
- Flow Control.
- Data Structures.
- Functions.

We will cover each of these topics in turn with small standalone examples that you can type and run.
Remember, whitespace has meaning in Python.

Assignment



Strings

```
# Strings
data = 'hello world'
print(data[0])
print(len(data))
print(data)
```

```
h
11
hello world
```

2. Crash Course in Python and SciPy

Assignment

Numbers

```python
# Numbers
value = 123.1
print(value)
value = 10
print(value)
```

```
123.1
10
```

Boolean

```python
# Boolean
a = True
b = False
print(a, b)
```

```
True, False
```

Multiple Assignment

```python
# Multiple Assignment
a, b, c = 1, 2, 3
print(a, b, c)
```

```
1, 2, 3
```

No Value

```python
# No value
a = None
print(a)
```

```
None
```

2. Crash Course in Python and SciPy

Flow Control : There are three main types of flow control that you need to learn: If-Then-Else conditions, For-Loops and While-Loops.

If-Then-Else Conditional

```
value = 99
if value == 99:
  print('That is fast')
elif value > 200:
  print('That is too fast')
else:
  print('That is safe')
```

Notice the colon (:) at the end of the condition and the meaningful tab intend for the code block under the condition. Running the example prints:

```
That is fast
```

For-Loop

```
# For-Loop
for i in range(10):
  print(i)
```

Running the example prints:

```
0
1
2
3
4
5
6
7
8
9
```

2. Crash Course in Python and SciPy

Flow Control : There are three main types of flow control that you need to learn: If-Then-Else conditions, For-Loops and While-Loops.

While-Loop

```
# While-Loop
i = 0
while i < 10:
  print(i)
  i += 1
```

Running the example prints:

```
0
1
2
3
4
5
6
7
8
9
```

2. Crash Course in Python and SciPy

Data Structures : There are three data structures in Python that you will find the most used and useful. They are tuples, lists and dictionaries.

Tuple

Tuples are read-only collections of items.

```
a = (1, 2, 3)
print(a)
```

Running the example prints:

```
(1, 2, 3)
```

List

Lists use the square bracket notation and can be indexed using array notation.

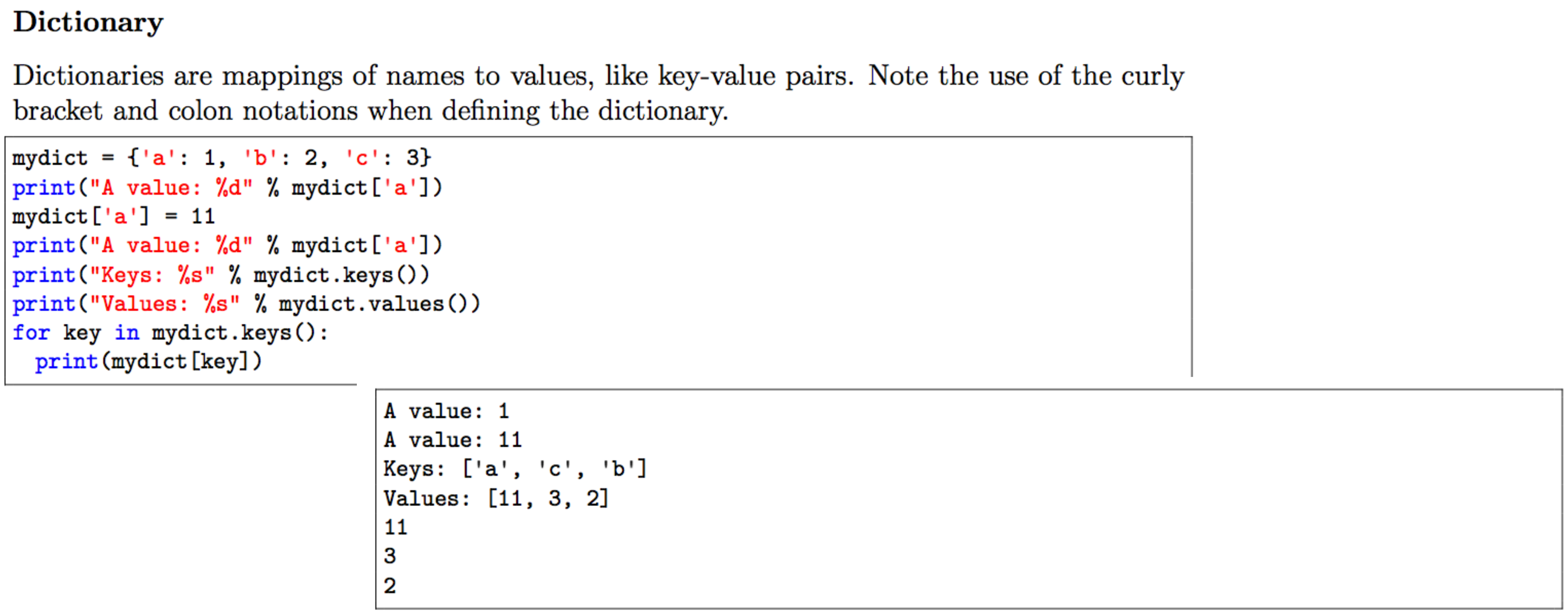
```
mylist = [1, 2, 3]
print("Zeroth Value: %d" % mylist[0])
mylist.append(4)
print("List Length: %d" % len(mylist))
for value in mylist:
  print(value)
```

Notice that we are using some simple **printf**-like functionality to combine strings and variables when printing. Running the example prints:

```
Zeroth Value: 1
List Length: 4
1
2
3
4
```

2. Crash Course in Python and SciPy

Data Structures : There are three data structures in Python that you will find the most used and useful. They are tuples, lists and dictionaries.



Dictionary

Dictionaries are mappings of names to values, like key-value pairs. Note the use of the curly bracket and colon notations when defining the dictionary.

```
mydict = {'a': 1, 'b': 2, 'c': 3}
print("A value: %d" % mydict['a'])
mydict['a'] = 11
print("A value: %d" % mydict['a'])
print("Keys: %s" % mydict.keys())
print("Values: %s" % mydict.values())
for key in mydict.keys():
  print(mydict[key])
```

```
A value: 1
A value: 11
Keys: ['a', 'c', 'b']
Values: [11, 3, 2]
11
3
2
```

2. Crash Course in Python and SciPy

Functions : The biggest gotcha with Python is the whitespace. Ensure that you have an empty new line after indented code. The example below defines a new function to calculate the sum of two values and calls the function with two arguments.

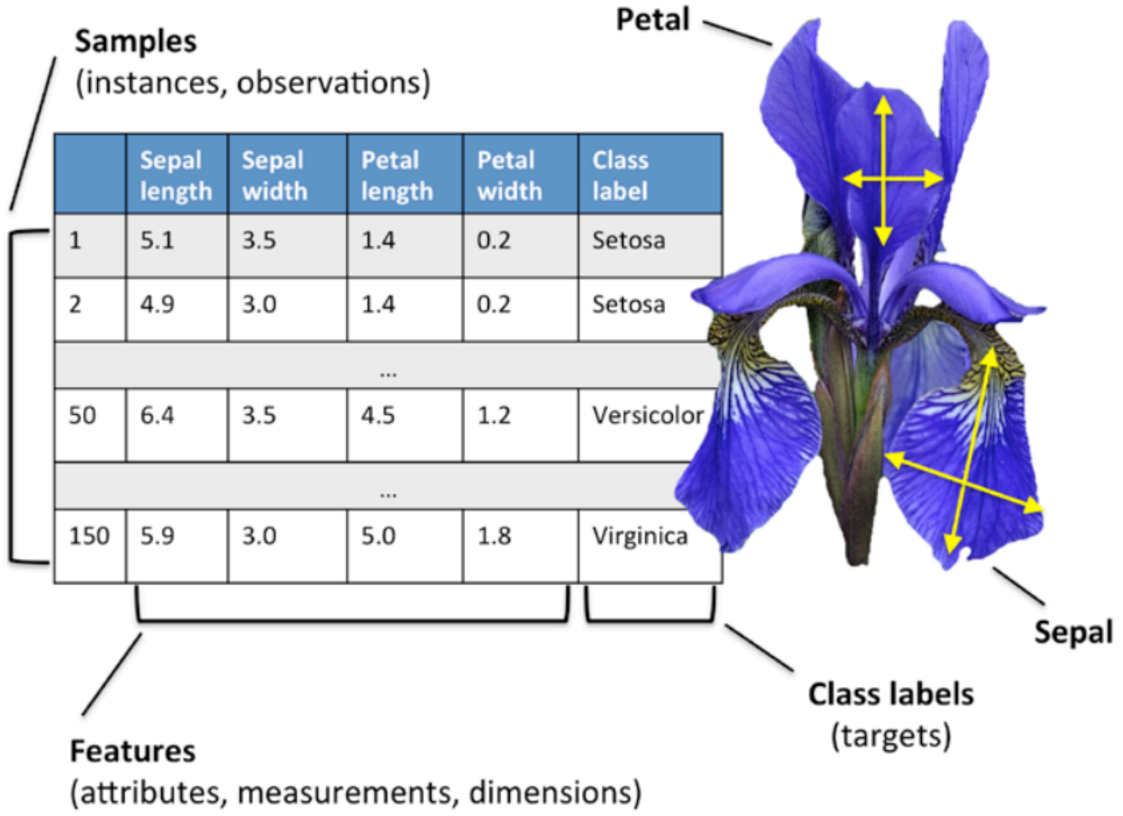
```
# Sum function
def mysum(x, y):
  return x + y

# Test sum function
result = mysum(1, 3)
print(result)
```

Running the example prints:

```
4
```

Recall: Tabular Data


Samples
(instances, observations)
Petal
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Row: samples, records, instances, observations
Column: attributes, features, measurements, dimensions

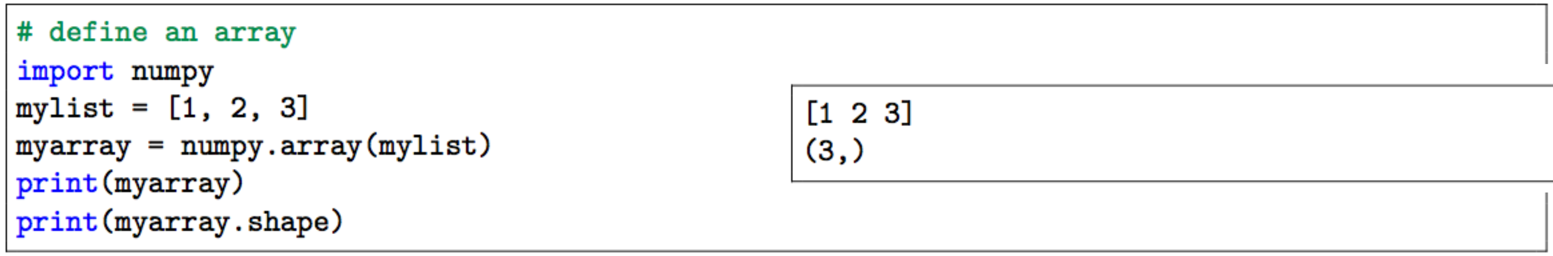
2. Crash Course in Python and SciPy

NumPy provides the foundational data structures and operations for SciPy.
These are arrays (ndarrays) that are efficient to define and manipulate.

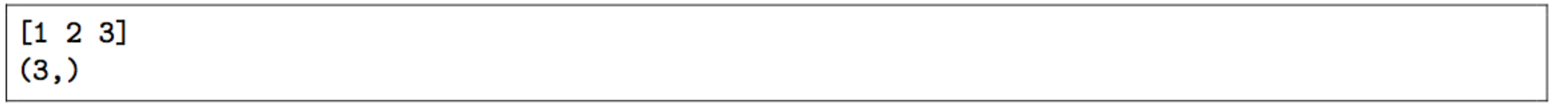
- Why we use NumPy
    - **NumPy Arrays and Matrices**
    - **Indexing and Slicing**
    - **Operations**
    - Exercise Questions and Solutions

2.2.1. Creating NumPy Array

- Converting existing objects to arrays
- Using numpy functions to generate arrays
- Creating arrays and matrices of random values
- Basic array attributes



```
# define an array
import numpy
mylist = [1, 2, 3]
myarray = numpy.array(mylist)
print(myarray)
print(myarray.shape)
```



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

2. Crash Course in Python and SciPy

2.2.2. Indexing and Selection

Array notation and ranges can be used to efficiently access data in a NumPy array.

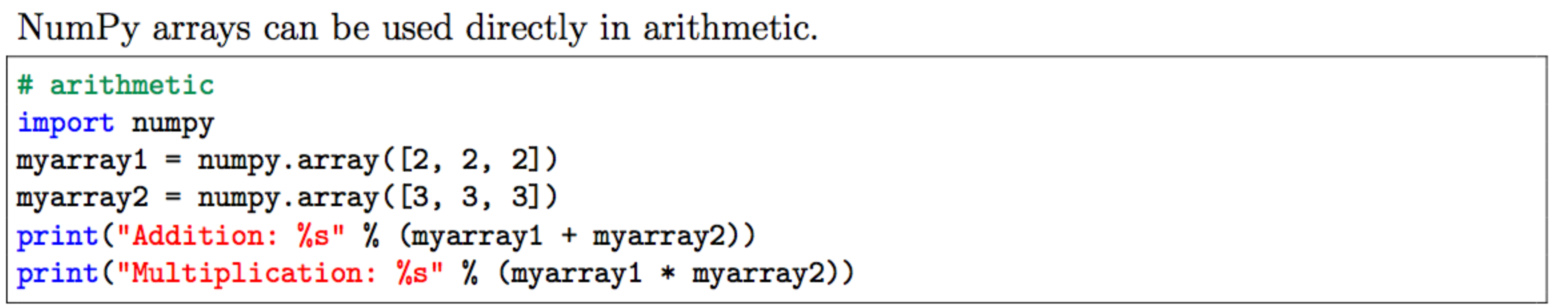
```
# access values
import numpy
mylist = [[1, 2, 3], [3, 4, 5]]
myarray = numpy.array(mylist)
print(myarray)
print(myarray.shape)
print("First row: %s" % myarray[0])
print("Last row: %s" % myarray[-1])
print("Specific row and col: %s" % myarray[0, 2])
print("Whole col: %s" % myarray[:, 2])
```

Running the example prints:

```
[[1 2 3]
 [3 4 5]]
(2, 3)
First row: [1 2 3]
Last row: [3 4 5]
Specific row and col: 3
Whole col: [3 5]
```

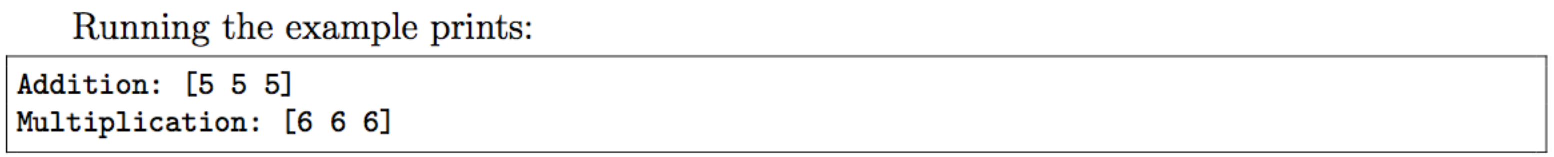
2. Crash Course in Python and SciPy

2.2.3. Operation



NumPy arrays can be used directly in arithmetic.

```
# arithmetic
import numpy
myarray1 = numpy.array([2, 2, 2])
myarray2 = numpy.array([3, 3, 3])
print("Addition: %s" % (myarray1 + myarray2))
print("Multiplication: %s" % (myarray1 * myarray2))
```



Running the example prints:

```
Addition: [5 5 5]
Multiplication: [6 6 6]
```

2.3. Pandas Crash Course

2. Crash Course in Python and SciPy

Pandas provides data structures and functionality to quickly manipulate and analyze data. The key to understanding Pandas for machine learning is understanding the Series and DataFrame data structures.

2.3.1. Series

A series is a one dimensional array of data where the rows are labeled using a time axis.

```
# series
import numpy
import pandas
myarray = numpy.array([1, 2, 3])
rownames = ['a', 'b', 'c']
myseries = pandas.Series(myarray, index=rownames)
print(myseries)
```

Running the example prints:

```
a    1
b    2
c    3
```

You can access the data in a series like a NumPy array and like a dictionary, for example:

```
print(myseries[0])
print(myseries['a'])
```

Running the example prints:

```
1
1
```

2. Crash Course in Python and SciPy

2.3.2. Dataframe

A data frame is a multi-dimensional array where the rows and the columns can be labeled.

```
# dataframe
import numpy
import pandas
myarray = numpy.array([[1, 2, 3], [4, 5, 6]])
rownames = ['a', 'b']
colnames = ['one', 'two', 'three']
mydataframe = pandas.DataFrame(myarray, index=rownames, columns=colnames)
print(mydataframe)
```

Running the example prints:

```
   one  two  three
a    1    2      3
b    4    5      6
```

Data can be indexed using column names.

```
print("method 1:")
print("one column:\n%s" % mydataframe['one'])
print("method 2:")
print("one column:\n%s" % mydataframe.one)
```

```
method 1:
one column:
a    1
b    4
method 2:
one column:
a    1
b    4
```

2. Crash Course in Python and SciPy

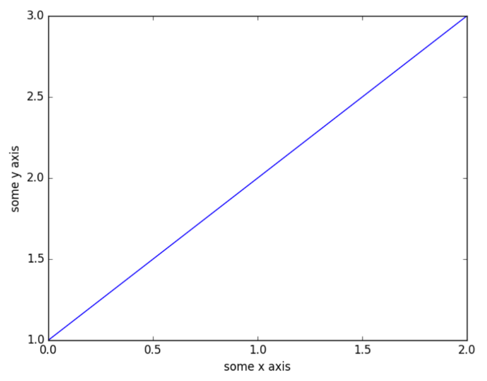
Matplotlib can be used for creating plots and charts. The library is generally used as follows:

- Call a plotting function with some data (e.g. .plot()).
- Call many functions to setup the properties of the plot (e.g. labels and colors).
- Make the plot visible (e.g. .show()).

Line Plot

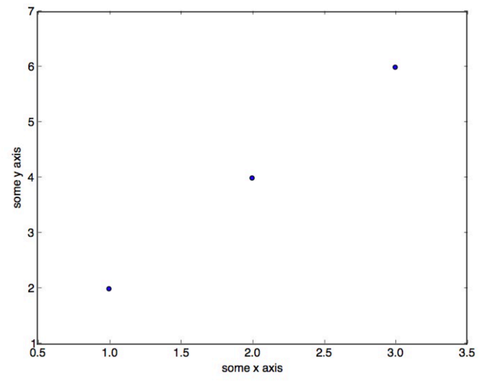
The example below creates a simple line plot from one dimensional data.

```
# basic line plot
import matplotlib.pyplot as plt
import numpy
myarray = numpy.array([1, 2, 3])
plt.plot(myarray)
plt.xlabel('some x axis')
plt.ylabel('some y axis')
plt.show()
```



Below is a simple example of creating a scatter plot from two dimensional data.

```
# basic scatter plot
import matplotlib.pyplot as plt
import numpy
x = numpy.array([1, 2, 3])
y = numpy.array([2, 4, 6])
plt.scatter(x,y)
plt.xlabel('some x axis')
plt.ylabel('some y axis')
plt.show()
```



2. Crash Course in Python and SciPy

Mean

The ‘Mean” is the average of a set of numbers.

The "Mean" is computed by adding all of the numbers in the data together and dividing by the number of elements contained in the data set.

Example: Data Set = 2, 5, 9, 7, 5, 4, 3
Number of Elements in Data Set = 7
Mean = (2 + 5 + 9 + 7 + 5 + 4 + 3) / 7 = 5

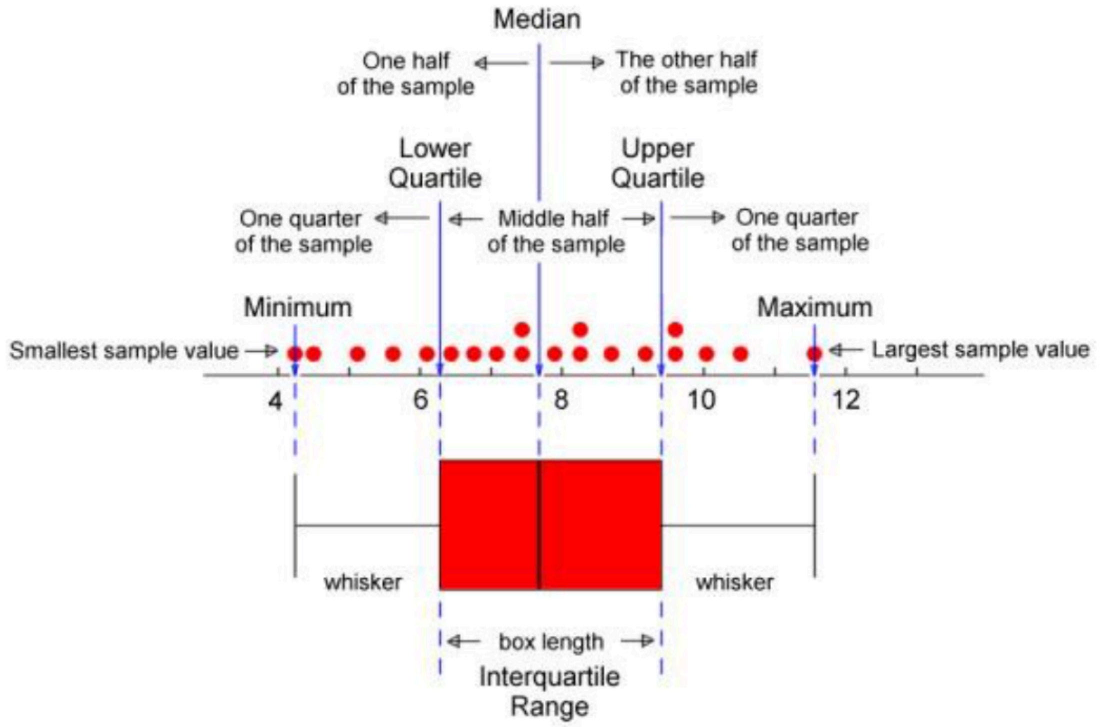
Median

The “Median” is the middle value of a set of **ordered** numbers.

The "Median" of a data set is dependent on whether the number of elements in the data set is odd or even. First reorder the data set from the smallest to the largest. If the number of elements is odd, then the Median is the element in the middle of the data set. If the number of elements is even, then the Median is the average of the two middle terms.

Example: Odd Number of Elements
Data Set = 2, 5, 9, 7, 5, 4, 3
Reordered = 2, 3, 4, 5, 5, 7, 9 - the middle term is 5
Median = 5

Example: Even Number of Elements
Data Set = 2, 5, 9, 3, 5, 4
Reordered = 2, 3, 4, 5, 5, 9 - the middle terms are 4 and 5
Median = (4 + 5) / 2 = 4.5 - the median is the average of the two middle terms



2. Crash Course in Python and SciPy

Mode

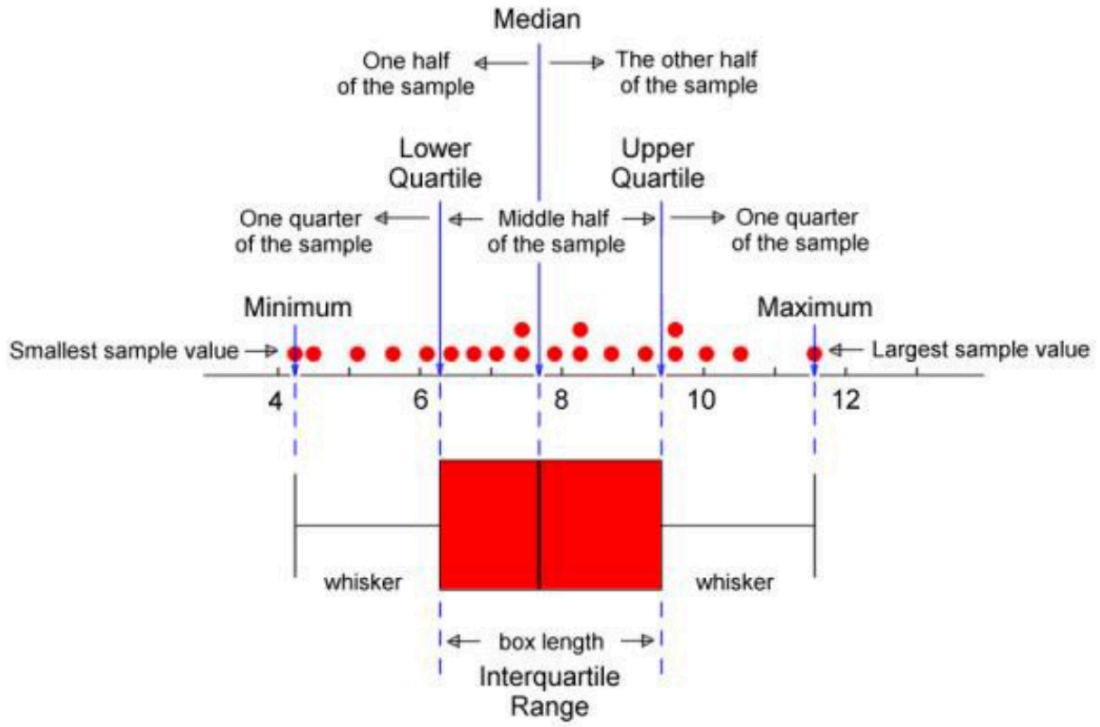
The "Mode" for a set of data is the value that occurs most often.

It is not uncommon for a data set to have more than one mode. This happens when two or more elements occur with equal frequency in the data set.

Example: One Mode
Data Set = 2, 5, 9, 7, 5, 4, 3
Mode = 5

Examples: Two Modes
Data Set = 2, 5, 2, 3, 5, 4, 7
Modes = 2 and 5

Example: Three Modes
Data Set = 2, 5, 2, 7, 5, 4, 7
Modes = 2, 5, and 7



Range

The "Range" is the difference between the largest value and smallest value in a set of data.

First reorder the data set from smallest to largest then subtract the first element from the last element.

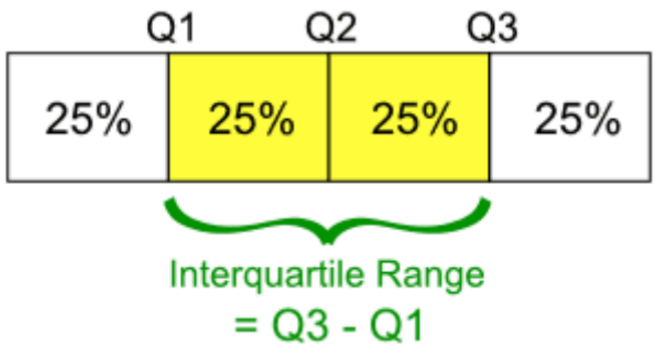
Example:
Data Set = 2, 5, 9, 7, 5, 4, 3
Reordered = 2, 3, 4, 5, 5, 7, 9
Range = (9 - 2) = 7

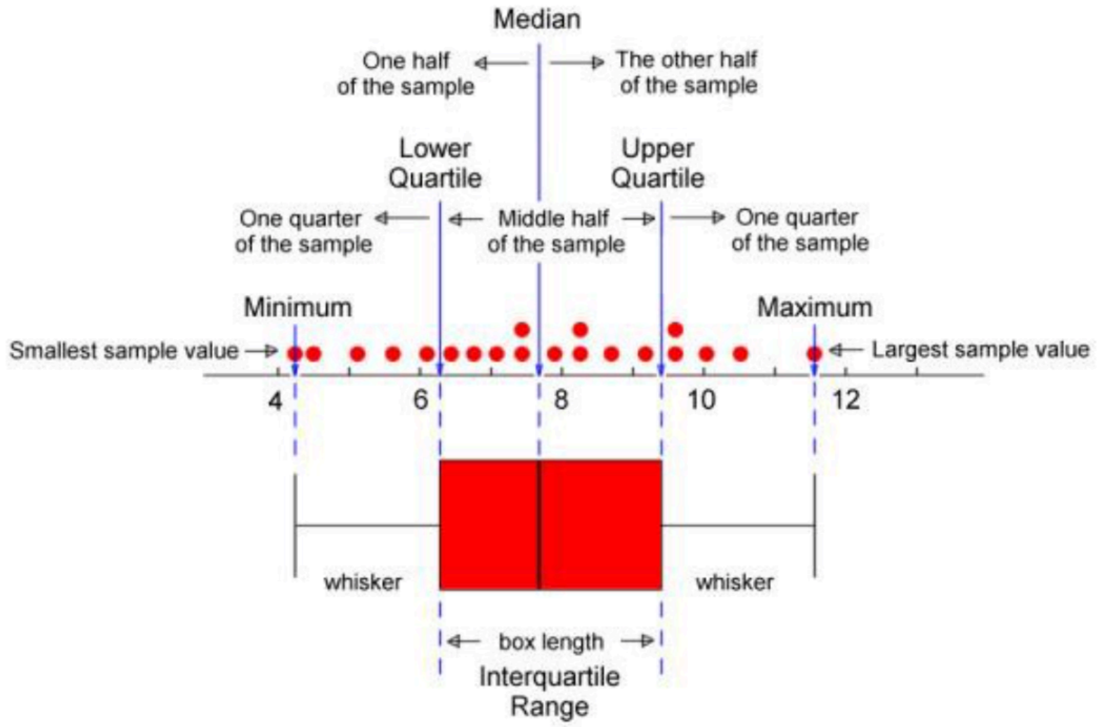
2. Crash Course in Python and SciPy

Interquartile Range

The "Interquartile Range" is the difference between smallest value and the largest value of the middle 50% of a set of data.

The "Interquartile Range" is from Q1 to Q3:





To find the interquartile range of a set of data:

- First put the list of numbers in order
- Then cut the list into four equal parts
- The quartiles are the "cuts"
- The interquartile range is the distance between the two middle sets of data

Summary

You have covered a lot of ground in this lesson. You discovered the basic syntax and usage of Python and three key **Python libraries used for machine learning:**

1. NumPy.
2. Matplotlib.
3. Pandas.