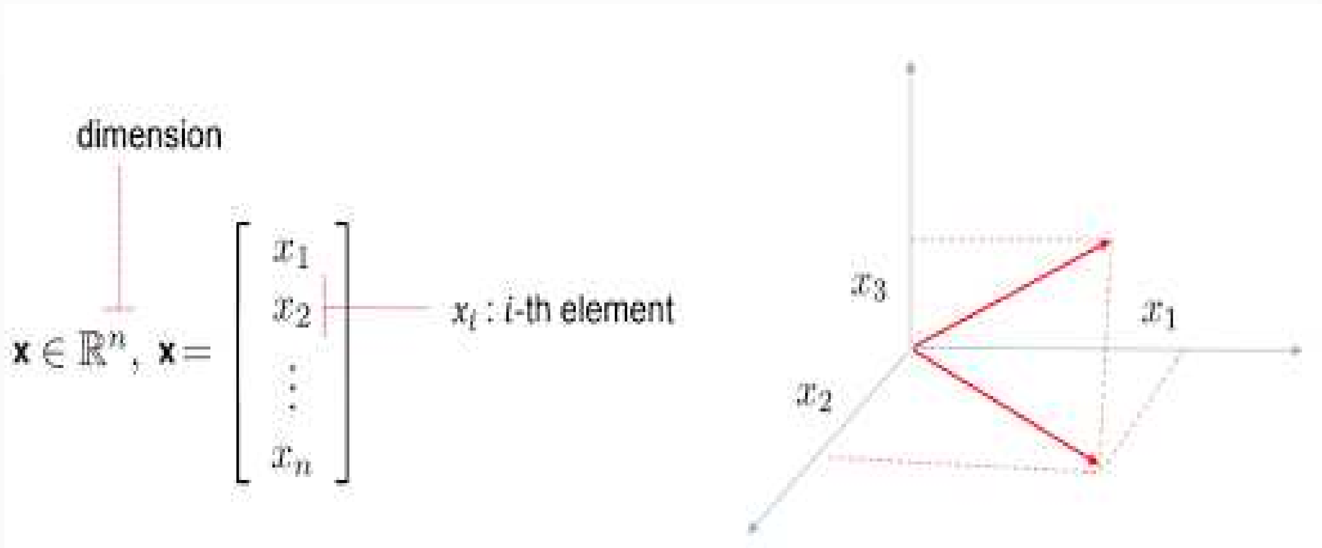
2. Feature Vector & Matrix

Vector

- A vector quantity has both magnitude and directions
  - In machine learning algorithms a data instance is represented by a vector, more precisely, by a feature vector



Vector operations

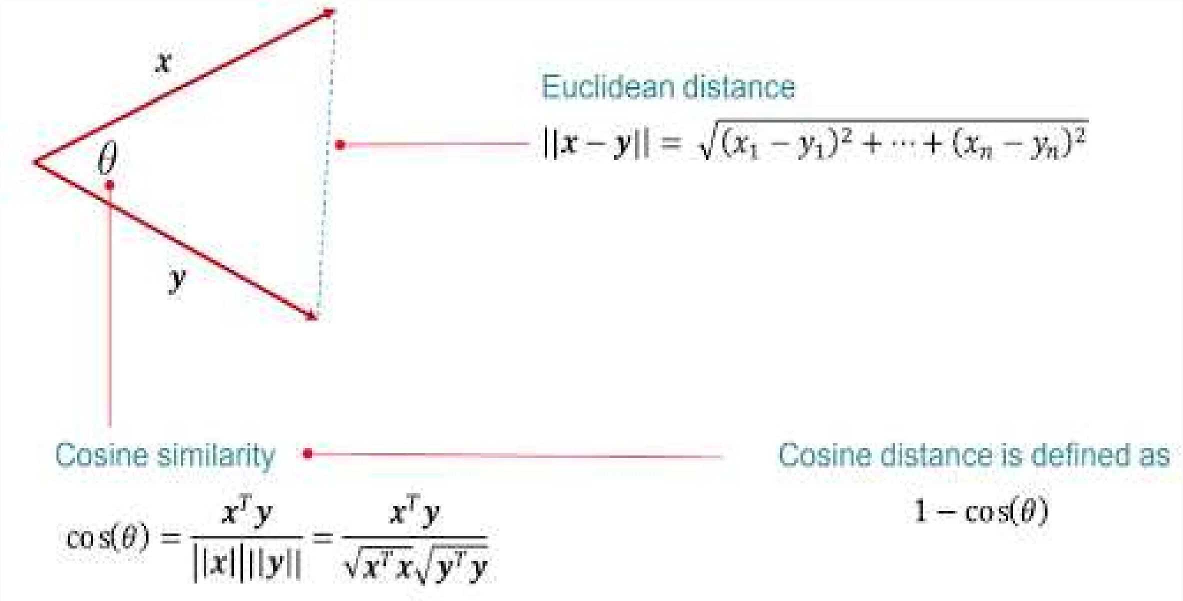
- Three main operations in vectors -
  - Transpose
  - Addition
  - Inner product
- Let's take two simple matrices x and y: $X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{bmatrix}$ $Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_n \end{bmatrix}$
- Transpose: $X^T = [x_1 \quad x_2 \quad .. \quad x_n]$

Vector operations...

- Addition: $$X + Y = \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ . \\ x_n + y_n \end{bmatrix}$$
- Inner product: $X^T Y = [x_1 y_1 + x_2 y_2 + .. + x_n y_n]$
- Magnitude of length of a vector: $length(X) = \sqrt{x_1^2 + x_2^2 + .. + x_n^2}$
- Above length known as 2-norm of vector: $$\|\boldsymbol{X}\|_2 = \sqrt{\boldsymbol{x}_1^2 + \boldsymbol{x}_2^2 + .. + \boldsymbol{x}_n^2}$$ $$\|\boldsymbol{X}\|_2 = \left(\boldsymbol{x}_1^2 + \boldsymbol{x}_2^2 + .. + \boldsymbol{x}_n^2\right)^{\frac{1}{2}}$$
- generalised to define a p-norm of a vector: $$\|X\|_p = \left(|x_1|^p + |x_2|^p + .. + |x_n|^p\right)^{\frac{1}{p}}$$

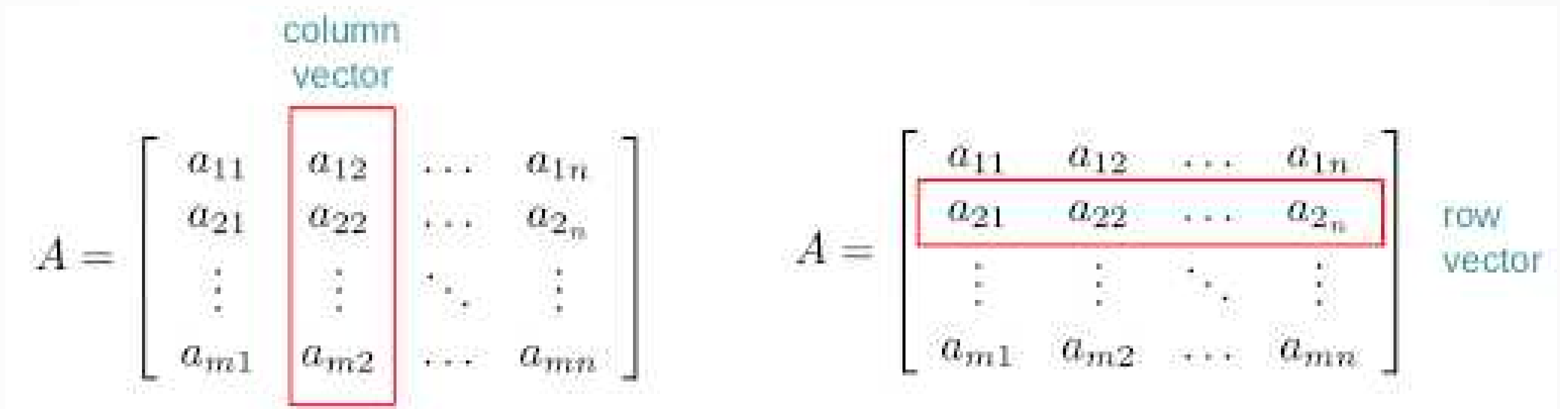
Distances between vectors

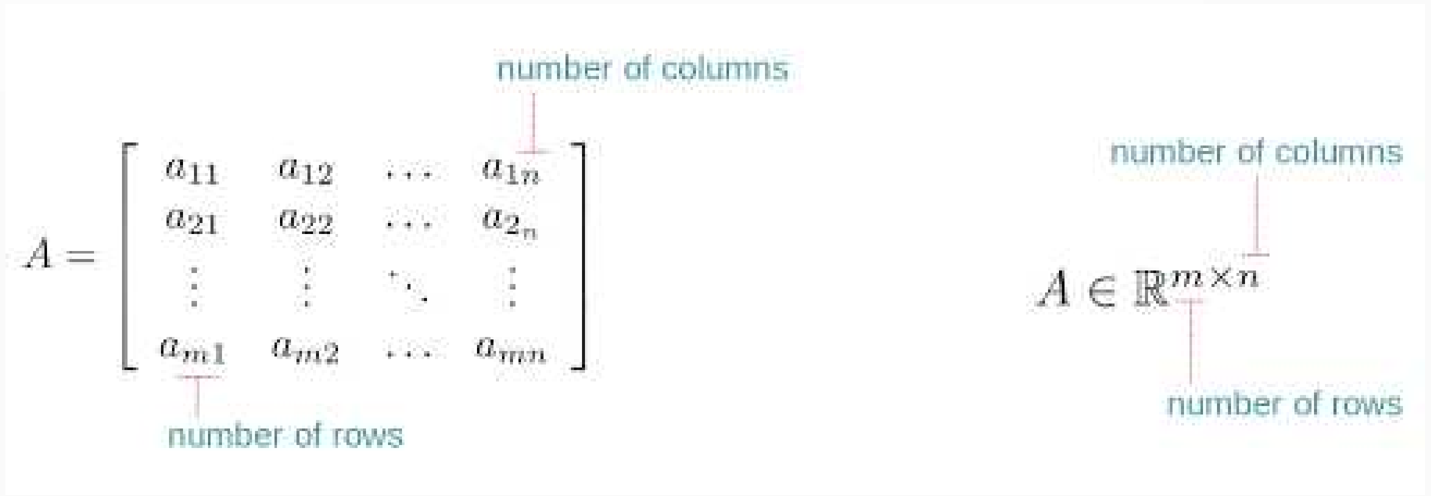
- Cosine similarity - measures the cosine of the angle between two vectors
  - measure of similarity between two vectors of an inner product space



Matrix

- Matrix has number of rows and columns





Matrix types

- Rectangular and Square Matrices
  - If a matrix A has size $m \times n$ such that $m=n$, then it is called a square matrix; otherwise it is a rectangular matrix

$$\begin{bmatrix} 1 & 6 \\ 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 5 \\ 6 & 2 & 4 \end{bmatrix}$$

square matrix

rectangular matrix

Matrix types

- Symmetric Matrices
  - a matrix is symmetric if it is equal to its transpose, $A = A^T$ that is

$$\begin{bmatrix} 1 & 6 \\ 6 & 7 \end{bmatrix}$$

- Symmatric matrices are always square.

Matrix types

- Diagonal Matrix
  - A matrix A is called a diagonal matrix if A(i,j)=0 for all i≠j.
  - Diagonal matrix is always a square matrix.

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

- Identity Matrix
  - A matrix I is called an identity matrix if it is a diagonal matrix and *I*(i,i)=1
  - I_{nxn} denotes nxn identity matrix.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix operations

- Transpose of a Matrix
  - putting all the matrix elements on rows on its columns. Lets say B is transpose of A, then B(i,j)=A(j,i)

$$\begin{bmatrix} 1 & 6 & 7 \\ 2 & 3 & 8 \end{bmatrix} \xleftrightarrow{\text{transpose}} \begin{bmatrix} 1 & 2 \\ 6 & 3 \\ 7 & 8 \end{bmatrix}$$

Matrix operations

- Matrix Addition/Subtraction
  - two matrices of same size

$$X+Y=\begin{bmatrix}2 & 4\\3 & 1\\8 & 5\end{bmatrix}+\begin{bmatrix}6 & 7\\4 & 4\\1 & 3\end{bmatrix}=\begin{bmatrix}8 & 11\\7 & 5\\9 & 8\end{bmatrix}$$

- Scalar Multiplication/Division
  - to multiply a matrix A with scalar c, multiply each element of A with c

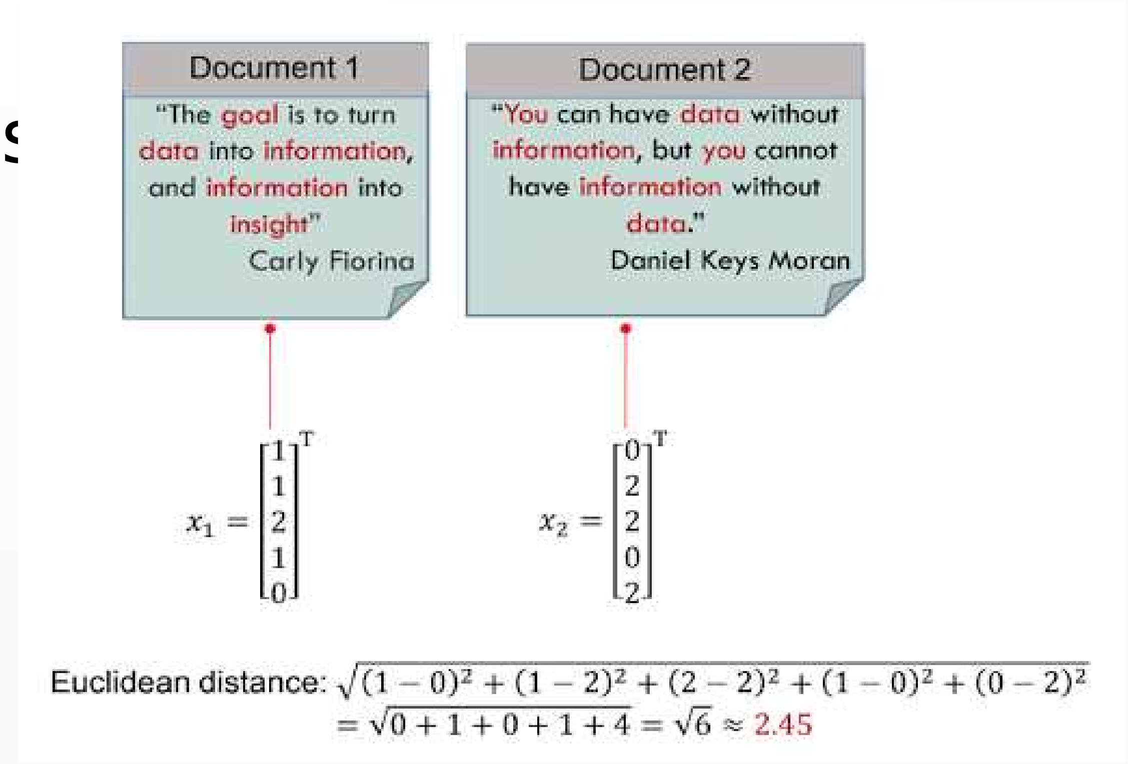
$$3\,x\begin{bmatrix}6 & 7\\4 & 4\\1 & 3\end{bmatrix}=\begin{bmatrix}18 & 21\\12 & 12\\3 & 9\end{bmatrix}$$

- Element wise Matrix Multiplication
  - matrices have the same size

$$\begin{bmatrix}2 & 4\\3 & 1\\8 & 5\end{bmatrix}\odot\begin{bmatrix}6 & 7\\4 & 4\\1 & 3\end{bmatrix}=\begin{bmatrix}12 & 28\\12 & 4\\8 & 15\end{bmatrix}$$

Feature Vectors

- Vector space model is representation of set of documents as vectors.
- It is a fundamental step in information retrieval operations
- Text data representation as **Feature Vectors**

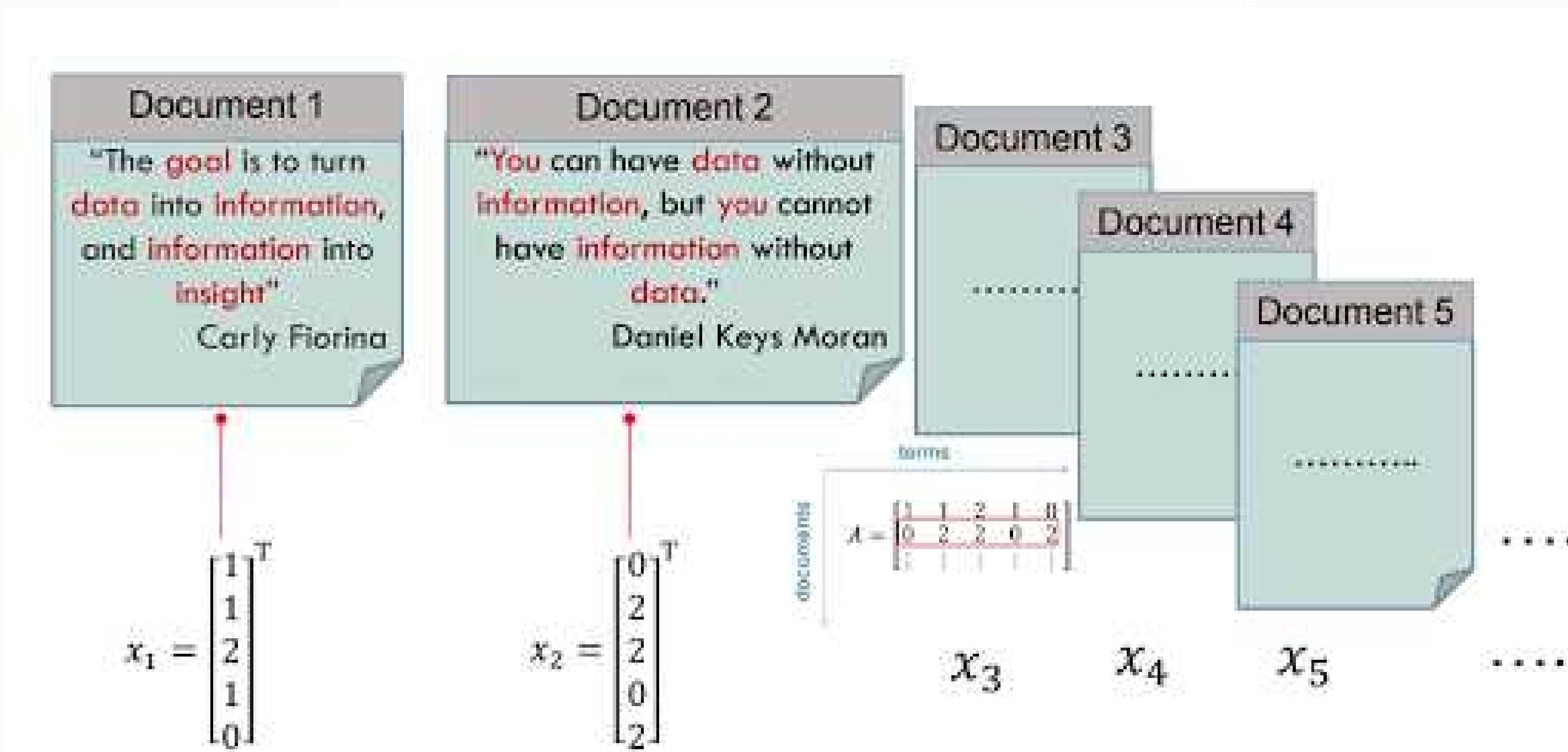


Feature matrix

- We can extend the concept of the feature vector towards a feature matrix by stacking feature vectors as a matrix X
  - We create a vocabulary of features for all the instances in the dataset
  - Represent each instance as a vector on features listed in the vocabulary
  - If our dataset has N instances, we create N vectors x1,x2,,...,xN
  - Each of these vectors is called a feature vector
  - We stack these vectors as a matrix X and call it a feature matrix

Feature matrix

- Example of steps mentioned earlier

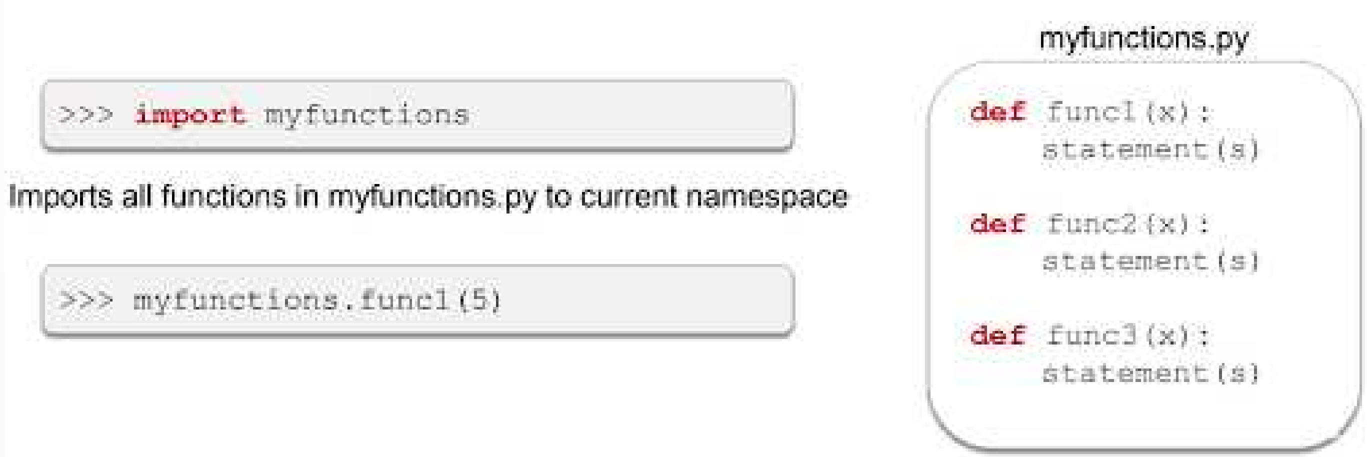


Python Programming

- Modules & Packages
  - Matplotlib
  - Numpy
  - Scipy
  - Scikit-learn
- Modelling using Scikit-learn
  - Linear
- Text analysis

Modules & Packages

- you can store your useful function definitions in a file
- then import it as a module in your current program



- Python comes with many libraries as standard packages
- Additional library packages can be downloaded over time (e.g., numpy)

Matplotlib

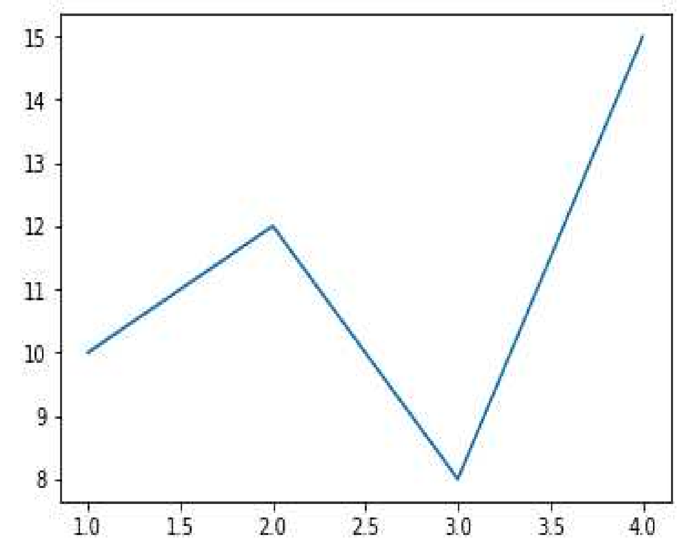
- Plotting with Matplotlib:
  - very versatile tool and is capable of generating high quality, cross-platforms graph images

[2…

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

x = np.array([1, 2, 3, 4])
y = np.array([10, 12, 8, 15])

plt.plot(x, y)
plt.show()
```



Numpy

- NumPy package
  - to create vectors and matrices and then perform some common linear algebra operations on these data

```
x = np.array([1,2,3])
print('An example of vector is:')
print(x)
A = np.array([(1,2),(3,4)])
print('An example of matrix is:')
print(A)

A = np.zeros([3,3])
print('An example of all zero matrix is:')
print(A)

A = np.ones([3,3])
print('An example of all one matrix is:')
print(A)

A = np.identity(3)
print('An example of an identity matrix is:')
print(A)

B = np.random.randn(4,3)
print('An example of a random matrix is:')
print(B)
```

```
An example of vector is:
[1 2 3]
An example of matrix is:
[[1 2]
 [3 4]]
An example of all zero matrix is:
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
An example of all one matrix is:
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
An example of an identity matrix is:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
An example of a random matrix is:
[[ 1.62880415 -0.57376857 -0.59805418]
 [ 2.13365345 -1.11635029 -2.38086695]
 [ 0.33302098  1.87509429  0.7108517 ]
 [ 0.40250957  1.07906447 -1.58486728]]
```

Numpy

- Matrix addition and subtraction:
  –

$$A+3=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}+3=\begin{bmatrix} a_{11}+3 & a_{12}+3 \\ a_{21}+3 & a_{22}+3 \end{bmatrix}$$

[6…

```
A = np.identity(2)
B = np.random.randn(2,2)
print('A=', A)
print('B=', B)
print('A+B=', A+B)
```

```
A= [[1. 0.]
 [0. 1.]]
B= [[-2.07207485  1.29934274]
 [-0.32812115 -0.01156101]]
A+B= [[-1.07207485  1.29934274]
 [-0.32812115  0.98843899]]
```

Numpy

- Matrix multiplication:
  -

$$A*3=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}*3=\begin{bmatrix} 3a_{11} & 3a_{12} \\ 3a_{21} & 3a_{22} \end{bmatrix}$$

```
A = np.identity(2)
B = np.random.randn(2,2)
print('A=', A)
print('B=', B)
print('A*B=', A*B)
```

```
A= [[1. 0.]
 [0. 1.]]
B= [[-0.48317533  1.91022493]
 [-1.10575629 -1.71650468]]
A*B= [[-0.48317533  0.        ]
 [-0.         -1.71650468]]
```

Numpy

- Matrix multiplication:

$$A*3=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}_{2,2} * \begin{bmatrix} c_{11} \\ c_{21} \end{bmatrix}_{2,1} = \begin{bmatrix} a_{11}c_{11}+a_{12}c_{21} \\ a_{21}c_{11}+a_{22}c_{21} \end{bmatrix}_{2,1}$$

```python
A = np.random.randn(2,2)
B = np.random.randn(2,1)
print('A=', A)
print('B=', B)
print('A.B=', A.dot(B))
# print('A.B=', np.dot(A, B))

print('B.A=', B.dot(A))
```

$$A*3=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} *3 = \begin{bmatrix} 3a_{11} & 3a_{12} \\ 3a_{21} & 3a_{22} \end{bmatrix}$$

```
A= [[-1.24919346  0.14974871]
 [ 0.32670595  1.5773978 ]]
B= [[ 1.86124703]
 [-0.03904939]]
A.B= [[-2.3309052 ]
 [ 0.54648405]]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
<ipython-input-13-bf615d4be707> in <module>()
      6 # print('A.B=', np.dot(A, B))
      7
----> 8 print('B.A=', B.dot(A))

ValueError: shapes (2,1) and (2,2) not aligned: 1 (dim 1) != 2 (dim 0)
```

Scikit-Learn Package

- robust machine learning library
- has great integration with the Python numerical and scientific libraries NumPy and SciPy
- list of all requirements for scikit-learn
  - **SciPy**: Fundamental library for scientific computing
  - **NumPy**: Base n-dimensional array package
  - **Pandas**: Data structures and analysis
  - **Matplotlib**: Comprehensive 2D/3D plotting
  - **IPython/Jupyter**: Enhanced interactive console
  - **Sympy**: Symbolic mathematics
- Install using, (*pip install -U scikit-learn*) or (*conda install scikit-learn*)

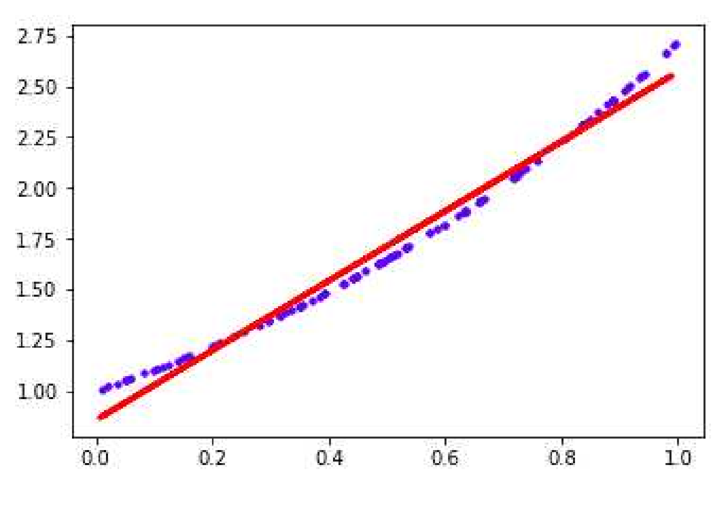
Dataset in Scikit-Learn

- a dataset is a dictionary-like object that holds all the data and some metadata about the data
- data is stored in the **.data** member, which is a *n_samples, n_features* array
- comes with a few standard datasets, i.e., the **iris** and **digits** datasets for classification and the **boston house prices** dataset for regression
- for supervised problem, one or more response variables are stored in the **.target** member

```python
from sklearn import datasets
digits = datasets.load_digits()
print('digits.data:', digits.data.shape)
print('digits.target:', digits.target)
```

```
digits.data: (1797, 64)
digits.target: [0 1 2 ... 8 9 8]
```

Learning models using the scikit-learn library




```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# create new linear model object
linear_model = LinearRegression()

# generate 100 random training data between 0-1 (train input)
X = np.random.rand(100, 1)
# create an exponential function as y (train output)
Y = np.exp(X)

# train the model
linear_model.fit(X, Y)

# generate test data (test input)
X_test = np.random.rand(300, 1)

# predict test label (test output)
Y_test = linear_model.predict(X_test)

plt.plot(X, Y, '.b')
plt.plot(X_test, Y_test, color='red', linewidth=3)
plt.show()
```