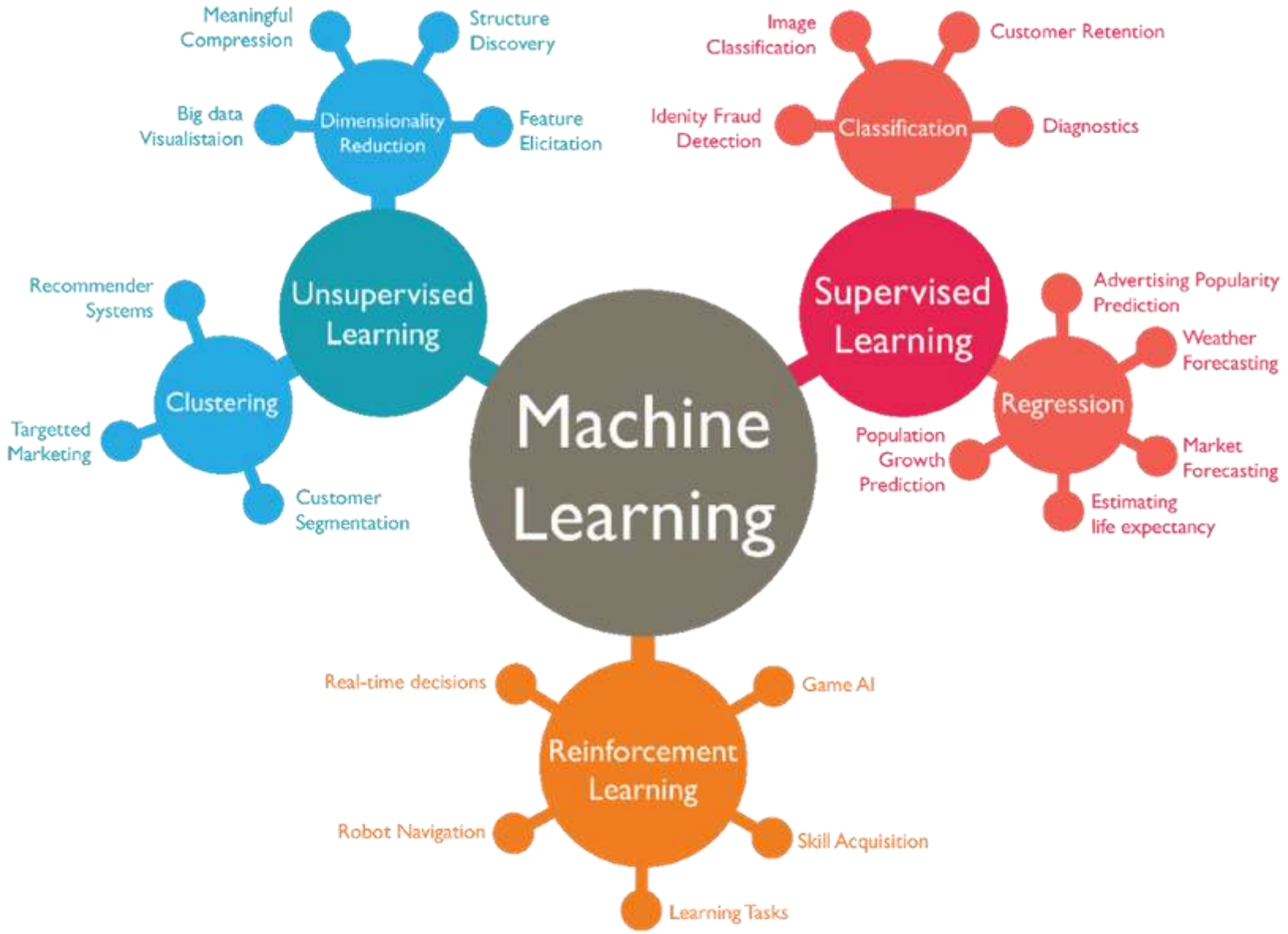
CIS

Conceptos en Python y Librerías de apoyo

APRENDIZAJE DE MÁQUINA
FERNANDO TESHIMA

UNIVERSIDAD DE LIMA

Repasando


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Supervisado

- Regresión Lineal
- Regresión Polinomial
- Regresión Logística
- Redes Neuronales
- Máquinas de Soporte Vectorial (SVM)
- Naive Bayes

No Supervisado

- Clustering Jerárquico
- K-means
- Análisis de Componente Principal (PCA)
- Expectation-Maximization

Representación de Data

Mayormente se trabajará con arreglos de n-dimensiones

x1	y	x2	x3	x4
sqft	price	City	bedrooms	baths
3392	339000	Dublin	3	2.1
4100	899900	pleasanton	4	3
3200	448641	Clayton	5	4
1436	239999	Moraga	4	3
1944	377500	Antioch	3	2
1500	299900	Danville	3	2.5
1700	265000	El Dorado Hills	4	3
2507	449000	Shingle Springs	4	3
1580	439950	McKinleyville	3	2
1500	699888	Marina	4	2
2705	1250000	Roseville	3	2
1715	439000	Rocklin	4	3

Vectores

- Suma
- Multiplicación escalar
- Norma
- Vector Unitario
- Dot product

Python

Variables

```
x = 2.345 # Un Número
saludos = "Hola" # Un string
a = [1 , 3 , 4 , 5 , 7 , 8] # Una Lista
```

Pueden cambiar de valor

```
x = 2.345 # valor inicial
x = -2 # x es ahora -2
x = 4 + 5 # x es ahora 9
```

A diferencia de C/C++ y Java, las variables pueden cambiar de tipo

No es necesario especificar el tipo de variable cuando se declara.

```
En Java : double x = 4.235;
String str = "Hola" ;

En Python : x = 4.235 # no se necesita(;)
str = "Hola"
```

Python mantiene internamente el tipo de variable

- `x = 4.235 # x es de tipo ‘double’`
- `x = [1 ,2 ,3 ,4 ,5] # x es ahora una lista`
- `x = ‘hola’ # x es un string`

Operaciones

- `2 + 5 # Suma`
- `2.2 * 3 # Multiplicación`
- `'Lo' * 4 # Genera ' LoLoLoLo '`
- `'hola, ' + 'mundo!' # Genera 'hola, mundo!'`

Shorthand

Shorthand combina asignación y suma/multiplicación:

```
x = 4
x -= 3 # x = x -3 , x es 3
y = 'hola'
y *= 2 # ahora y es 'holahola'
```

Otros operadores shorthand son `/=, +=, %=`

Comparar

- `x == 5 # revisa igualdad`
- `y != ’ hello ’ # revisa desigualdad`
- `4 > 6 # Falso`
- `4 >= 2 # Verdadero`
- `2.5 < z < 6.6 # Desigualdades encadenadas`

Operadores Booleanos

```
x = 2
y = 'Hola'
x == 2 or y == 'Hola' # Verdadero
x < 0 and y != 'Hola' # Falso
not y # Negación
```

Control de Flujo

- IF

```
if x:
    print('hola')
if (( x and y ) or not z ):
    print('hola')
else:
    print('hi!')
```

- FOR-LOOP

```
arr = [1 ,2 ,3 ,4]
for number in arr :
    print(number * 2) # muestra ’2 ,4 ,6 ,8’
```

Funciones

```
def suma(x , y):
     print(x + y)
suma(1 , 2) # muestra 3
```

Pueden retornar valores

```
def find_max (x , y):
     if x >= y :
          return x
     return y

print( find_max (4 , 5)) # muestra 5
```

Indentación

- Python usa la indentación para agrupar código
- Cada bloque de código tiene el mismo nivel de indentación. (ej. For-loop, funciones, if, etc.)
- Python va a indicar si no se tiene la indentación adecuada
- 2 o 4 espacios es la indentación estándar en Python

```
a = 1
b = 3
if a > b :
    tmp = a
        a = b # error!
    b = tmp
```

¿Qué está mal con estas funciones?

```
def f1 (x , y , z ):
if x < y :
    return z
return 0

def f2 (x , y , z ):
    if x < y :
    return z
    return 0
```

Respuesta

```
def f1 (x , y , z ):
    if x < y :          # indentado
        return z        # indentado
    return 0            # indentado

def f2 (x , y , z ):
    if x < y :
        return z        # indentado
    return 0
```

Estructura de Datos

- Listas
- String
- Tuplas
- Diccionarios

Veremos solo Listas y String

En la web de Python está la información sobre las otras estructuras

https://docs.python.org/es/3/tutorial/datastructures.html

Listas

- Las listas en Python soportan indexado. Empiezan en 0

```
seq = [2, 4, 6 ,8]
print(seq [0])    # muestra '2'
print(seq [3])    # muestra '8'
print(seq [-1])   # muestra '8'
print(seq [4])    # error
seq.append(10)    # agrega 10 al final de 'seq '
print(seq [4])    # muestra '10'
```

```
l1 = [2, 4, 6]
l2 = [8, 10]
l3 = l1 + l2          # l3 es [2 ,4 ,6 ,8 ,10]
l3.pop()              # quita 10 de l3

print(l3[0:3])        # muestra [2 ,4 ,6]
print(l3[3:])         # muestra [8]
l3[1:3] = [7 ,7]      # l3 es ahora [2 ,7 ,7 ,8]

# funciones nativas
len(l3)               # 4, número de elementos
max(l3)               # 8, máximo
min(l3)               # 2, mínimo
```

- El For-Loop se puede usar para iterar y buscar elementos en la lista

```
# No necesitan ser del mismo tipo
list = [2, 4, 'naranjas', 6.0 , 'azul']
for elemento in list :
    print(elemento)

if 'azul' in list and 'rojo' not in list:
    print('hola') # esto se va a mostrar

# muestra junto el elemento y el índice
for indx, val in enumerate(list):
    print(indx, val)
```

Strings

- Las propiedades son similares a las listas

```
str = 'Hello,World!’

print(str[0])          # muestra 'H'
print(str[ -1])        # muestra '!’, index negativo
print(str[6:11])       # muestra 'World ‘

str += '!!'            # str es 'Hello,World!!!’

len(str)               # 14, largo del string
str = str.lower()      # 'hello,world!!!'
```

Import

```
import math

# el codigo esta en squares.py
import squares

# importar un a libreria con otro nombre
import numpy as np
import matplotlib.pyplot as plt
```

NumPy

- Es una extensión de Python que añade soporte para arreglos y matrices de gran tamaño y de múltiples dimensiones.
- NumPy trata a los objetos como ndarray (arreglo de n dimensiones de data homogénea)

Funciones Básicas

```
import numpy as np

# crear un ndarray
a = np.array ([1 ,2 ,3])

a.shape                          # muestra “(3 ,)”
print(a[0],a[1],a[2])            # muestra “1 2 3”

a [0] = 5
print(a)                    # muestra [5 ,2 ,3]
```

Index y Slicing

```
# crear un array 2D de forma (3 ,4)
a = np . array ([[1 ,2 ,3 ,4],
                 [5 ,6 ,7 ,8],
                 [9 ,10 ,11 ,12]])

print(a[0,1])                # muestra 2

# slicing para obtener un sub array [[2 ,3] ,[6 ,7]]
b = a[:2 ,1:3]               #desde inicio a 1, desde 1 a 2

# cambiar valores en b
b [0,0] = 77
print(a[0 ,1])               # muestra 77
```

Un slice es una vista del array original. Si se cambia, también se modifica en el array original

Broadcast

```
arr1 = np . array ([1 ,2 ,3 ,4 ,5])


# multiplicar cada elemento de arr1 por 5
print ( arr1 * 5) # muestra  '[5 ,10 ,15 ,20 ,25]'
```

De igual forma funciona cuando se pasa el array a una función

```
def masuno ( x ):
     return x + 1

masuno(1)              # Muestra 2
masuno(arr1)           # Muestra [2 ,3 ,4 ,5 ,6]
```

Vectores

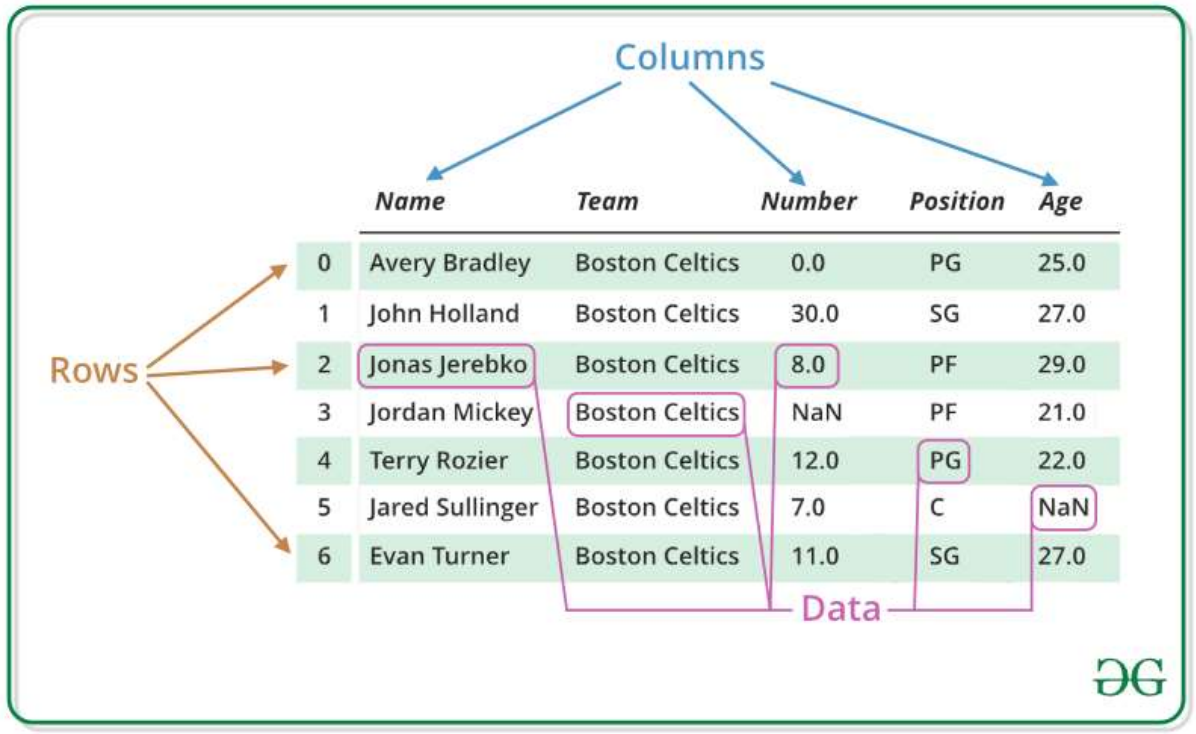
```
u = np.array ([1 ,2 ,3])
v = np.array ([4 ,5 ,6])

# Suma de Vectores
u + v
# Dot product
np.dot (u , v )
```

Hay que imaginarlo como una hoja de cálculo

Dataframe es un objeto 2D con indices y columnas

Series es un objeto 1D similar a las columnas

Provee de muchas funciones similares a las hojas de calculo (tablas dinámicas, max, min, isnan, …)



pandas

Uso general

seaborn

Cuando se quiere aspectos de estadística

Herramientas de Visualización