Taller Aprendiendo Python I

JUAN HERNÁNDEZ

Python - NumPy

- **ndarray.ndim** : the number of axes (dimensions) of the array.
- **ndarray.shape** : the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.
- **ndarray.size** : the total number of elements of the array. This is equal to the product of the elements of shape.
- **ndarray.dtype** : an object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.
- **ndarray.itemsize** : the size in bytes of each element of the array. For example, an array of elements of type float64 has itemsize 8 (=64/8), while one of type complex32 has itemsize 4 (=32/8). It is equivalent to ndarray.dtype.itemsize.
- **ndarray.data** : the buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

Python - NumPy

```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

Python - NumPy

```
import numpy as np
a = np.zeros((2,2))              # Create an array of all zeros
print(a)                         # Prints "[[ 0.  0.]
                                 #          [ 0.  0.]]"
b = np.ones((1,2))    # Create an array of all ones
print(b)                  # Prints "[[ 1.  1.]]"
c = np.full((2,2), 7)  # Create a constant array
print(c)                      # Prints "[[ 7.  7.]
                              #          [ 7.  7.]]"
d = np.eye(2)         # Create a 2x2 identity matrix
print(d)                # Prints "[[ 1.  0.]
                        #          [ 0.  1.]]"
e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]
                             #              [ 0.68744134  0.87236687]]"
```

Python - NumPy

```
import numpy as np
# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"
b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])   # Prints "77"
```

Matplolib

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
import numpy as np
# Prepare the data
x = np.linspace(0, 10, 100)
# Plot the data
plt.plot(x, x, label='linear')
# Add a legend
plt.legend()
# Show the plot
plt.show()
```

MatPlolib

```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
import numpy as np
# Create a Figure
fig = plt.figure()
# Set up Axes
ax = fig.add_subplot(111)
# Scatter the data
ax.scatter(np.linspace(0, 1, 5), np.linspace(0, 5, 5))
# Show the plot
plt.show()
```

Pandas

```
In [3]: s = pd.Series([1, 3, 5, np.nan, 6, 8])

In [4]: s
Out[4]:
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Pandas

```
In [5]: dates = pd.date_range('20130101', periods=6)

In [6]: dates

Out[6]:

DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',

               '2013-01-05', '2013-01-06'],

              dtype='datetime64[ns]', freq='D')

In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

In [8]: df

Out[8]:

                   A         B         C         D

2013-01-01  0.469112 -0.282863 -1.509059 -1.135632

2013-01-02  1.212112 -0.173215  0.119209 -1.044236

2013-01-03 -0.861849 -2.104569 -0.494929  1.071804

2013-01-04  0.721555 -0.706771 -1.039575  0.271860

2013-01-05 -0.424972  0.567020  0.276232 -1.087401

2013-01-06 -0.673690  0.113648 -1.478427  0.52498
```

Pandas

```
In [9]: df2 = pd.DataFrame({'A': 1.,
   ...:                     'B': pd.Timestamp('20130102'),
   ...:                     'C': pd.Series(1, index=list(range(4)), dtype='float32'),
   ...:                     'D': np.array([3] * 4, dtype='int32'),
   ...:                     'E': pd.Categorical(["test", "train", "test", "train"]),
   ...:                     'F': 'foo'})
   ...:
In [10]: df2
Out[10]:
     A          B    C  D      E    F
0  1.0 2013-01-02  1.0  3   test  foo
1  1.0 2013-01-02  1.0  3  train  foo
2  1.0 2013-01-02  1.0  3   test  foo
3  1.0 2013-01-02  1.0  3  train  foo
```

Pandas

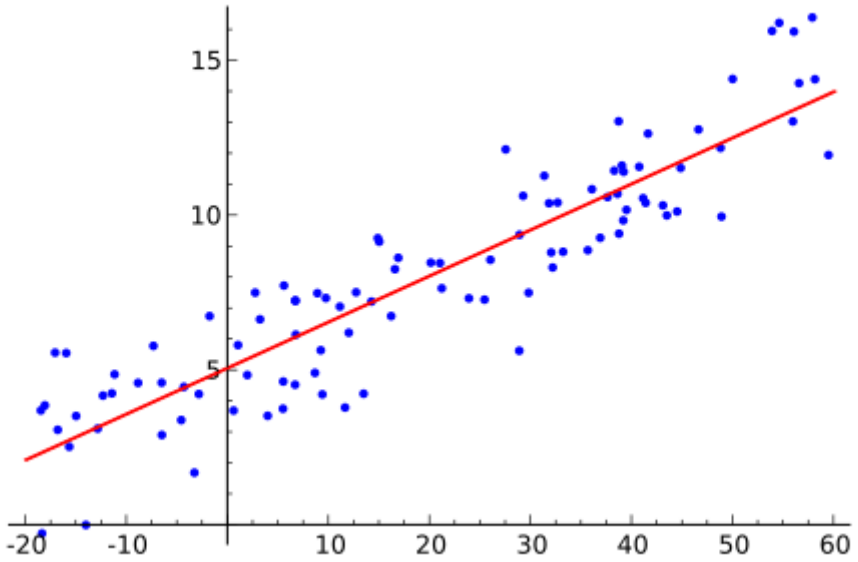
```
In [11]: df2.dtypes
Out[11]:
A          float64
B    datetime64[ns]
C          float32
D            int32
E         category
F           object
dtype: object
```

Machine Learning- Supervisado

La primera modalidad de aprendizaje que tiene el machine learning es la de aprendizaje supervisado. Usándola, se entrena al algoritmo otorgándole las preguntas, denominadas características, y las respuestas, denominadas etiquetas. Esto se hace con la finalidad de que el algoritmo las combine y pueda hacer predicciones.

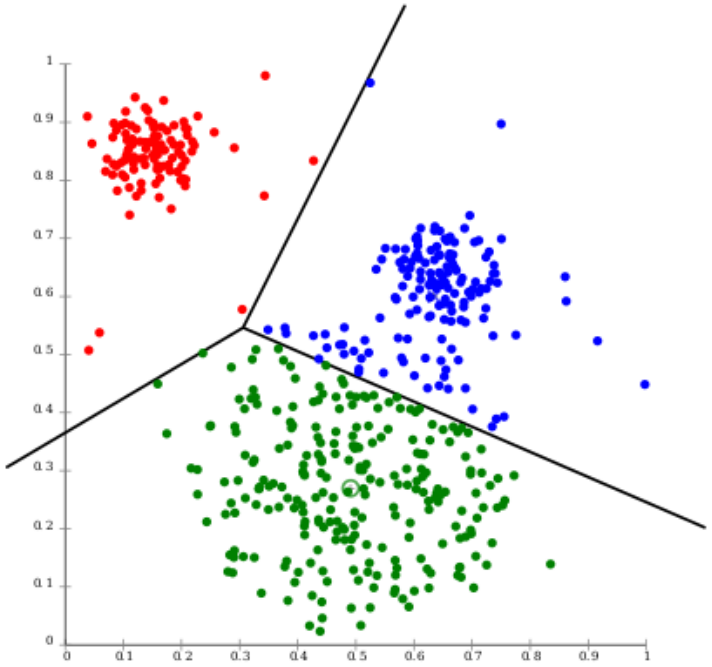
Regresión Lineal

Tiene como resultado un número específico. Si las etiquetas suelen ser un valor numérico, mediante las variables de las características, se pueden obtener dígitos como dato resultante.



Clasificación

en este tipo, el algoritmo encuentra diferentes patrones y tiene por objetivo clasificar los elementos en diferentes grupos



El algoritmo no está en capacidad de determinar a qué grupo pertenece un valor o cuál es el resultado de una operación. Solamente logra relacionar características con etiquetas y así obtener un resultado.