APRENDIZAJE AUTOMÁTICO

PRÁCTICAS - PRÁCTICA 0

Pedro Gallego López

Doble Grado de Ingeniería Informática y Matemáticas

Universidad de Granada
Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

8 de marzo de 2021

Práctica 0

Introducción

La práctica en cuestion se propone para reforzar los conocimientos adquiridos en las primeras clases de rácticas, en donde se han introducido los principios de *Python*, *Matplotlib*, *Numpy* y *Scikit-learn*.

Ejercicio 1

Leer la base de datos de iris que hay en scikit-learn. Obtener las características (datos de entrada X) y la clase (y). Quedarse con las características 1 y 3 (primera y tercera columna de X). Visualizar con un color diferente (con naranja, negro y verde), e indicando con una leyenda la clase a la que corresponde cada color

La base de datos **iris** contiene 3 tipos de flores iris: Setosa, Versicolor y Virginica, donde se tienen sus cuatro características: ancho y alto del sépalo además del ancho y largo del pétalo. Esto guardado en un `numpy.ndarray` de 150×4.

Para poder cargar la base de datos iris hacemos uso de los datasets de `sklearn`: utilizamos concretamente `datasets.load_iris()`. Lo guardamos en una variable denominada `iris`, esta variable contiene dos atributos que nos darán precisamente las características y la clase, estos atributos son `iris.data` y `iris.target` respectivamente. Para quedarnos con las características 1 y 3, que son el largo tanto del sépalo como del pétalo, hacemos uso del acceso a los elementos del array que proporciona numpy: `array[:,:3:2]` cogiendo las filas pares menores menores que 3, así serían la 0 y la 2, que justamente son las características que nos piden.

- `X = iris.data[:,:3:2]`
- `y = iris.target`

Para los colores he cogido los siguientes en formato hexadecimal:

- Naranja: #FF9300
- Negro: #000000
- Verde: #31BF00

La asignación de estos colores la hacemos dentro de la función `plt.scatter` donde al parámetro `cmap` le asignamos `colors.ListedColormap(colores)` siendo `colores` un vector que contiene los colores nombrados anteriormente. Al parámetro `edgecolor` le asignamos el valor `'k'` para que tengan borde los puntos y podamos diferenciarlos. Asignamos también nombre a los ejes.

Para la leyenda debemos de coger la salida que nos da `plt.scatter` y coger como parámetro `handle` de la función `plt.legend`: `scatter.legend_elements()[0]` para saber a qué punto asignar cada etiqueta. Las etiquetas las cogemos directamente de nuestra variable `iris`

El resultado sería el siguiente:

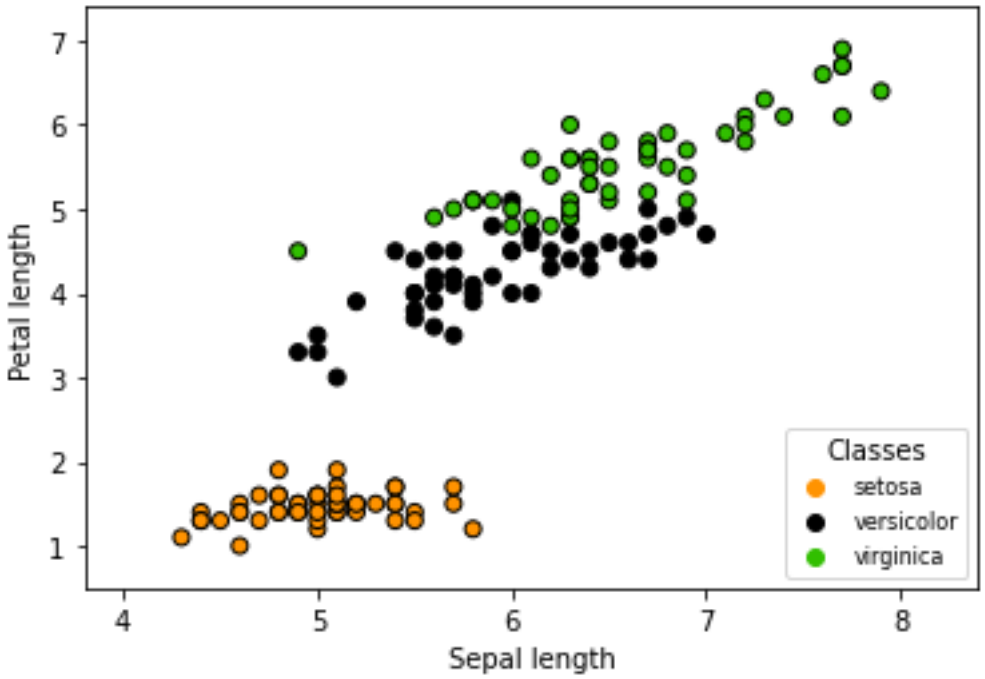

Figura 1.1

Ejercicio 2

Separar en training (75 % de los datos) y test (25 %) aleatoriamente conservando la proporción de elementos en cada clase tanto en training como en test. Con esto se pretende evitar que haya clases infra-representadas en entrenamiento o test.

`train_test_split` de `sklearn.model_selection` va a ser la función que nos va a solucionar el ejercicio. Cargamos los datos en `(X,y)` y ejecutamos la línea:

- `X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, shuffle=True, stratify=y)`

Donde hemos definido en `train_size=0.75` que el training tenga el 75 % de los datos y el test el 25 %. Con `shuffle=True` damos aleatoriedad y con `stratify=y` conseguimos mantener la proporción que nos pide el enunciado. Para comprobar que de verdad se cumple esto he hecho comprobaciones con print, la salida está en la tabla 1.1.

Vemos como en el caso del TEST no es exactamente 1/3 del total y dista un poco, esto se debe a que el número $0{,}25 \times 150$ que es el número de test que hay, no es un número entero, esto hará que el reparto no sea exactamente igual, pero si lo más aproximado posible. Es más, ejecutando varias veces el código se observa como las proporciones son exactamente las mismas permutándose entre ellas.

Cantidad del train: 112 que supone un 0.7466666666666667 % del total
Cantidad del test: 38 que supone un 0.2533333333333335 % del total
En el TOTAL están las proporciones
Clase 1: 0.3333333333333333 %
Clase 2: 0.3333333333333333 %
Clase 3: 0.3333333333333333 %
En el TRAIN están ahora las proporciones
Clase 1: 0.33035714285714285 %
Clase 2: 0.33035714285714285 %
Clase 3: 0.3392857142857143 %
En el TEST están ahora las proporciones
Clase 1: 0.34210526315789475 %
Clase 2: 0.34210526315789475 %
Clase 3: 0.3157894736842105 %

Tabla 1.1: Salida de comprobación de resultados

Ejercicio 3

Obtener 100 valores equiespaciados entre 0 y 4π. Obtener el valor de sin(x), cos(x) y tanh(sin(x)+cos(x)) para los 100 valores anteriormente calculados. Visualizar las tres curvas simultáneamente en el mismo plot (con líneas discontinuas en verde, negro y rojo).

Para obtener los 100 valores equiespaciados entre 0 y 4π usamos

- `np.linspace(0, 4*np.pi, (100))`

Obtenemos los valores de las funciones con `y1=np.sin(x)`, `y2=np.cos(x)` y `y3=np.tanh(y1+y2)` aprovechando los valores ya calculados con `y1, y2`. Creamos los plots con `plt.subplots()` y utilizamos los parámetros `'g--','k--','r--'` para conseguir las líneas discontinuas en verde, negro y rojo: las letras marcan el color y el `--` marca que sea discontinua. El resultado sería el siguiente:

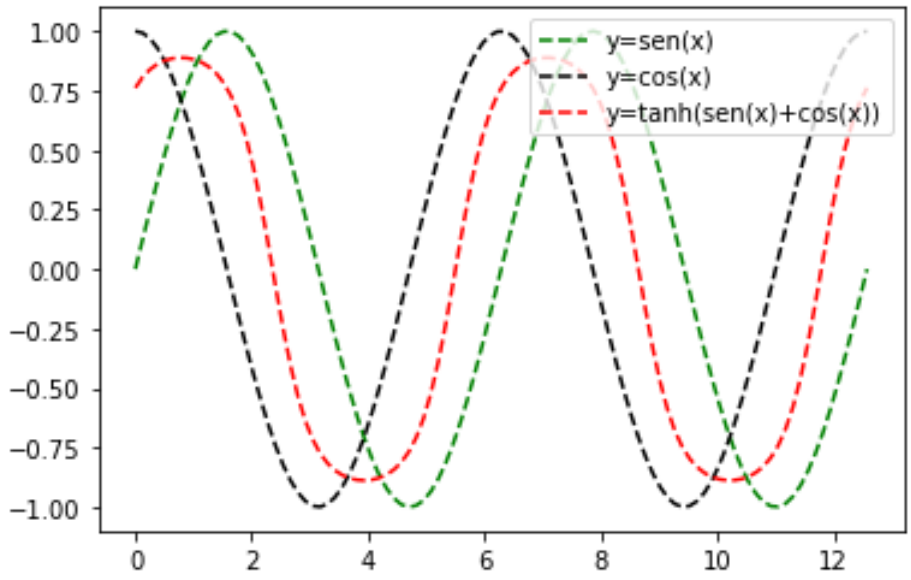

Figura 1.2: Gráfica del ejercicio 3.