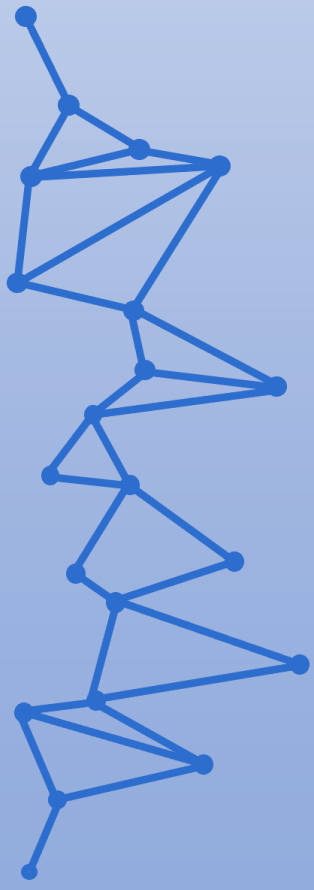
Curso de Especialización de
Inteligencia Artificial y Big Data (IABD)



Programación de Inteligencia Artificial

UD04. Librerías de programación de Aprendizaje Automático con Python. Resumen.

JUAN ANTONIO GARCIA MUELAS

Llamamos librería, dentro de un framework o lenguaje de programación, al conjunto de clases y métodos o funciones escritas en ese lenguaje de programación, y que permiten realizar determinadas tareas con menos líneas de código.

PANDAS.

La librería Pandas fue **creada en 2008, y se liberó en 2009** como proyecto de código abierto. Más tarde, en 2015, pasó a estar gestionado por NumFOCUS.

El principal **logro**, es poder **generar un objeto** de la **clase DataFrame,** muy versátil y con gran capacidad de manipulación. Facilita enormemente el trabajo de los científicos de datos.

Introduce la parte de código que falta en la siguiente instrucción para cargar datos con la librería Pandas en un proyecto de ciencia de datos:

```
import pandas as pd
data = pd.read_csv('datos.csv')
```

La función `shape` de Pandas permite ver las dimensiones principales de un array.

La función `head` de Pandas muestra por defecto 5 registros.

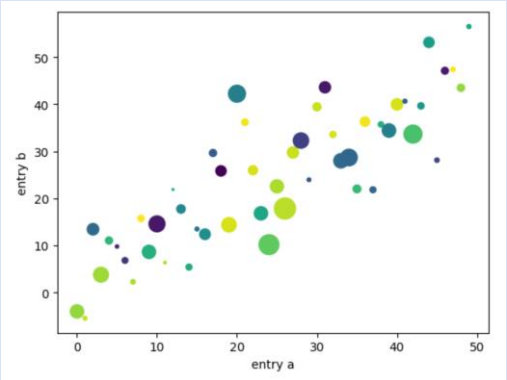
La función `describe` de Pandas nos muestra los principales valores estadísticos de un dataset.

La función `plot` de Pandas se utiliza para representaciones sencillas.

MATPLOLIB

Matplotlib es la **librería para Python** para hacer **representaciones gráficas y de imágenes en procesos de ciencia de datos.** Permite, incluso, hacer gráficas interactivas, con pocas líneas de código. Está **basada en** la **gestión de arrays de NumPy,** y creada como **parche para Python en 2002 por John Hunter**.

Introduce el código necesario en el espacio en blanco para poder hacer una representación de tipo dispersión de puntos como la de esta imagen




```
data = {'a': np.arange(50),'c': np.random.randint(0, 50, 50),'d':
np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

La función de `Pyplot` que sirve para crear gráficas de dispersión de puntos es `scatter`.

SCIKIT LEARN

Scikit learn es una de las librerías de aprendizaje automático más utilizadas en todo el mundo. Surgió como un proyecto de **Google Summer of Code** de la mano de **David Cournapeau.** Su concepto inicial era ser una extensión auxiliar a Scipy. Contiene **algoritmos en Cython** para **cálculos pesados**.

Es **muy utilizada en regresión lineal, regresión logística, random forest y máquinas de vector soporte. Permite** crear modelos de **redes neuronales**, **pero no se suele utilizar** para esa técnica **tras la aparición de Tensorflow**.

Está especialmente enfocada en la generación y entrenamiento de modelos sencillos de aprendizaje automático.

La clase para generar un modelo de K-Nearest Neighbors con Scikit-learn es KNeighborsClassifier()

Rellena los espacios en blanco con el código necesario para crear y entrenar un modelo de regresión logística utilizando la librería Scikit learn

```
from sklearn.linear_model import LogisticRegression
data = pd.read_csv('data.csv')
y = data['result']
X = data.drop('result', axis=1)
model = LogisticRegression()
model.fit(X, y)
```

TENSORFLOW Y KERAS

Tensorflow, desarrollado originalmente por **Google Brain y publicado** con licencia de código abierto **Apache en 2015**. Fue rápidamente absorbido por la comunidad en torno al aprendizaje automático, que ya contaba con otros incentivos, como las competiciones de Kaggle, y las nuevas opciones de hardware.

En **2019** sale la **versión alfa de TensorFlow 2, que incluye Keras (un subpaquete integrado)**, desarrollada **por François Chollet**, con módulos de alto nivel que simplifican la programación de **redes neuronales profundas**.

Actualmente incluye también, **Tensorboard** para **visualizar entrenamientos y resultados**, o **Tensorflow Lite**, para **integrar los modelos en dispositivos móviles** en modo de cómputo en el perímetro (**edge computing**).

Rellena el hueco en blanco para que el código a continuación genere un modelo de tipo Sequential (a capas) con dos capas, una de 128 neuronas y otra de 10 neuronas.

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(128, activation='relu')
model.add(layers.Dense(10, activation='softmax')
```

Sequential es la clase de la librería Keras que nos permite construir un modelo de deep learning con varias capas.

La expresión layers.Dense() añade una capa de tipo red neuronal cuando estamos trabajando con Keras.

PYTORCH

Pytorch es una **librería de código abierto** para aprendizaje automático **basada en Torch** para Python por el **equipo de investigación de IA de Facebook**. Optimizada para **cálculo de tensores,** para modelos de **redes neuronales profundas**. Se puede implementar **incluso en producción**.

Cabe desatacar la librería fastai, que simplifica el desarrollo de inteligencia artificial, con **modelos pre-entrenados que ya vienen cargados**.

Dado el siguiente código que define la clase a utilizar a la hora de generar un modelo de red neuronal profunda con Pytorch, ¿Cuál es la llamada correcta para crear el modelo?

```
class Net(nn.Module):
  def __init__(self, input_size, hidden_size, num_classes):
    super(Net,self).__init__()
    self.fc1 = nn.Linear(input_size, hidden_size)
    self.relu = nn.ReLU()
    self.fc2 = nn.Linear(hidden_size, num_classes)

  def forward(self,x):
    out = self.fc1(x)
    out = self.relu(out)
    out = self.fc2(out)
    return out
```

respuesta:

```
net = Net(input_size, hidden_size, num_classes)
```

La clase para generar un modelo de Maquina de Vectores Soporte o Support Vector Machine es SVC().