Argentina programa 4.0

Ministerio de Desarrollo Productivo Argentina

Secretaría de Economía del Conocimiento

PROGRAMADOR JUNIOR EN MACHINE LEARNING I

UNIVERSIDAD NACIONAL DE MISIONES

FACULTAD DE CIENCIAS EXACTAS QUÍMICAS Y NATURALES

CURSO 1: Fundamentos de Machine Learning (ML) Python - Nivel 1 -

FUNDAMENTACIÓN

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes.

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Python puede utilizarse en proyectos de inteligencia artificial, para crear sitios web escalables, realizar cálculos estructurales complejos con elementos finitos, y diseñar videojuegos, entre otras muchas aplicaciones

Objetivos

- Familiarizar al estudiante con el entorno y las herramientas de programación de Python.
- Desarrollar las capacidades de plasmar una propuesta de resolución a un problema sencillo planteado utilizando las herramientas planteadas.
- Identificar las ventajas del trabajo colaborativo y las herramientas disponibles.
- Promover la indagación sobre los temas planteados.

Tabla de contenidos

Introducción a librerías de Python

Entornos virtuales

Un entorno virtual en Python es una forma de aislar las dependencias de un proyecto de las del sistema operativo o de otros proyectos. Esto permite tener diferentes versiones de Python y de los paquetes instalados en cada entorno, sin que haya conflictos entre ellos. Además, facilita el despliegue del proyecto con sus propios módulos dependientes.

Para crear un entorno virtual Python en Windows, se puede utilizar el módulo venv que viene incluido con Python 3.3 o superior. El comando para crear un entorno virtual es:

```
python -m venv venv
```

Esto crea una carpeta en la ruta especificada con el nombre del entorno (por ejemplo, `.venv`). Esta carpeta contiene los ejecutables de Python y los directorios para instalar los paquetes. Para activar el entorno virtual, se debe ejecutar el siguiente comando desde la consola:

```
venv/Scripts/activate
```

Esto cambia el intérprete de Python predeterminado al del entorno virtual y muestra el nombre del entorno a la izquierda de la línea de comandos. Para desactivar el entorno virtual, se debe ejecutar el comando:

```
deactivate
```

Para instalar paquetes en el entorno virtual, se puede utilizar el gestor de paquetes pip, que se instala automáticamente con venv. El comando para instalar un paquete es:

```
pip install nombre-del-paquete
```

También se puede crear un archivo requirements.txt con la lista de los paquetes y sus versiones que se requieren para el proyecto. Para instalar todos los paquetes del archivo, se debe ejecutar el comando:

```
pip install -r requirements.txt
```

¿Qué es una librería?

Python es un lenguaje de programación muy popular y versátil que se utiliza para diversos fines, como análisis de datos, desarrollo web, inteligencia artificial y más. Una de las ventajas de Python es que cuenta con una gran cantidad de librerías o módulos que amplían sus funcionalidades y facilitan la realización de tareas complejas. Una librería de Python es un conjunto de código que se puede importar y utilizar en un programa, sin tener que escribirlo desde cero. Algunas de las librerías más conocidas y utilizadas de Python son: numpy, pandas, matplotlib, scikit-learn, tensorflow, etc.

Instalación de librerías

Para instalar librerías en Python, existen diferentes métodos según el sistema operativo y el entorno de desarrollo que se utilice. Una forma común y sencilla de instalar librerías en Python es mediante el uso de pip, que es un gestor de paquetes que permite descargar e instalar librerías desde el repositorio oficial PyPI (Python Package Index). Para usar pip, se necesita tener instalado Python y acceder a la línea de comandos del sistema. Luego, se puede ejecutar el siguiente comando para instalar una librería:

```
pip install nombre_de_la_librería
```

Por ejemplo, para instalar la librería numpy, se puede ejecutar:

```
pip install numpy
```

Si se quiere instalar una versión específica de una librería, se puede indicar después del nombre con el signo ==. Por ejemplo, para instalar la versión 1.18.5 de numpy, se puede ejecutar:

```
pip install numpy==1.18.5
```

Para actualizar una librería a la última versión disponible, se puede usar el parámetro --upgrade. Por ejemplo, para actualizar numpy, se puede ejecutar:

```
pip upgrade numpy
```

Para desinstalar una librería, se puede usar el comando uninstall seguido del nombre de la librería. Por ejemplo, para desinstalar numpy, se puede ejecutar:

```
pip uninstall numpy
```

Estos son algunos de los comandos básicos para instalar librerías en Python con pip. Para más información y opciones, se puede consultar la documentación oficial de pip: https://pip.pypa.io/en/stable/

Manejo de librerías

Para importar una librería, se utiliza la palabra clave `import`.

```
import numpy
```

Al elemento importado se le puede asignar un alias. Un alias es un nombre alternativo que se le puede asignar a un módulo al importarlo en Python. Esto se hace para facilitar el uso del módulo, especialmente si tiene un nombre largo o complejo. Por ejemplo, si queremos importar el módulo numpy, que es una librería para trabajar con matrices y operaciones matemáticas, podemos usar el alias np:

```
import numpy as np
```

De esta forma, podemos acceder a las funciones y atributos del módulo numpy usando el prefijo np en lugar de numpy. Por ejemplo:

```
a = np.array([1, 2, 3]) # crea un arreglo de numpy
b = np.sum(a) # calcula la suma de los elementos del arreglo
```

Un módulo es un archivo de Python que contiene definiciones y declaraciones de variables, funciones, clases u otros objetos. Un módulo puede ser parte de una librería o paquete, que es un conjunto de módulos relacionados entre sí. Para importar un módulo específico de una librería, se usa la sintaxis:

```
from libreria import modulo
```

Por ejemplo, si queremos importar el módulo pyplot, que es parte de la librería matplotlib y sirve para crear gráficos, podemos usar:

```
from matplotlib import pyplot
```

De esta forma, podemos acceder a las funciones y atributos del módulo pyplot usando el prefijo pyplot. Por ejemplo:

```
pyplot.plot([1, 2, 3], [4, 5, 6]) # crea un gráfico de líneas
pyplot.show() # muestra el gráfico en pantalla
```

La librería Matplotlib

¿Para qué sirve Matplotlib?

Matplotlib es una librería de Python que permite crear gráficos de alta calidad de forma sencilla y rápida. Matplotlib ofrece una gran variedad de tipos de gráficos, como líneas, barras, histogramas, pastel, dispersión, etc. Además, permite personalizar muchos aspectos de los gráficos, como los colores, las etiquetas, las leyendas, los ejes, etc.

Para usar Matplotlib, lo primero que hay que hacer es importar la librería con el siguiente comando:

```
import matplotlib.pyplot as plt
```

Este comando importa el módulo `pyplot` de Matplotlib y le asigna el alias `plt`, que es el más usado por la convención. El módulo pyplot proporciona una interfaz sencilla para crear y manipular gráficos.

Uso de Matplotlib

Para crear un gráfico con Matplotlib, se siguen los siguientes pasos:

1. Crear los datos que se quieren representar. Por ejemplo, se pueden usar listas, arrays de NumPy o series de Pandas.
2. Llamar a la función correspondiente al tipo de gráfico que se quiere crear. Por ejemplo, plt.plot() para un gráfico de líneas, plt.bar() para un gráfico de barras o plt.scatter() para un gráfico de dispersión. Estas funciones reciben como argumentos los datos que se quieren representar y otros parámetros opcionales para personalizar el gráfico.
3. Añadir los elementos adicionales que se quieran al gráfico, como el título, las etiquetas de los ejes, la leyenda, etc. Para ello se usan funciones como plt.title(), plt.xlabel(), plt.ylabel(), plt.legend(), etc.
4. Mostrar el gráfico en pantalla o guardarlo en un archivo. Para mostrarlo se usa la función plt.show(), que abre una ventana con el gráfico. Para guardarlo se usa la función plt.savefig(), que recibe como argumento el nombre del archivo donde se quiere guardar el gráfico.

A continuación se muestra un ejemplo de cómo crear un gráfico de líneas con Matplotlib usando datos aleatorios:

```
# Importar la librería
import matplotlib.pyplot as plt
# Importar NumPy para generar datos aleatorios
import numpy as np
# Crear los datos
x = np.linspace(0, 10, 100) # 100 puntos equiespaciados entre 0 y 10
y = np.sin(x) # Función seno aplicada a cada punto
# Crear el gráfico
plt.plot(x, y) # Gráfico de líneas con x e y
```

```
plt.title("Gráfico de seno") # Título del gráfico
plt.xlabel("x") # Etiqueta del eje x
plt.ylabel("y") # Etiqueta del eje y
# Mostrar el gráfico
plt.show()
```

Ejemplos de código de varios tipos de gráficos con matplotlib

Matplotlib es una librería de Python que permite crear gráficos de datos de forma sencilla y versátil. En este texto se presentan algunos ejemplos de código de varios tipos de gráficos con matplotlib sin utilizar csv, es decir, usando datos generados o definidos en el mismo código.

Un tipo de gráfico muy común es el de líneas, que sirve para representar la evolución de una o más variables en función de otra. Por ejemplo, el siguiente código muestra cómo graficar el seno y el coseno de un ángulo en radianes:

```
import matplotlib.pyplot as plt
import numpy as np
# Generar un array de 100 valores entre 0 y 2*pi
x = np.linspace(0, 2*np.pi, 100)
# Calcular el seno y el coseno de x
y1 = np.sin(x)
y2 = np.cos(x)
# Crear una figura y un eje
fig, ax = plt.subplots()
# Graficar las líneas con diferentes colores y estilos
ax.plot(x, y1, color="red", linestyle="-", label="Seno")
ax.plot(x, y2, color="blue", linestyle="--", label="Coseno")
# Añadir una leyenda
ax.legend()
# Añadir títulos a los ejes y a la figura
ax.set_xlabel("Ángulo (rad)")
ax.set_ylabel("Valor")
ax.set_title("Gráfico de líneas con matplotlib")
# Mostrar el gráfico
plt.show()
```

Otro tipo de gráfico muy útil es el de barras, que sirve para comparar el valor de una variable categórica entre diferentes grupos. Por ejemplo, el siguiente código muestra cómo graficar el número de estudiantes por carrera en una universidad:

```
import matplotlib.pyplot as plt
# Definir los datos
carreras = ["Ingeniería", "Matemáticas", "Física", "Química",
"Biología"]
estudiantes = [120, 80, 60, 40, 30]
```

```
# Crear una figura y un eje
fig, ax = plt.subplots()
# Graficar las barras con los datos
ax.bar(carreras, estudiantes)
# Añadir títulos a los ejes y a la figura
ax.set_xlabel("Carrera")
ax.set_ylabel("Número de estudiantes")
ax.set_title("Gráfico de barras con matplotlib")
# Mostrar el gráfico
plt.show()
```

Un tercer tipo de gráfico que se puede hacer con matplotlib es el de dispersión o scatter plot, que sirve para explorar la relación entre dos variables numéricas. Por ejemplo, el siguiente código muestra cómo graficar la altura y el peso de 50 personas:

```
import matplotlib.pyplot as plt
import numpy as np
# Generar datos aleatorios con una distribución normal
np.random.seed(42) # Fijar una semilla para reproducibilidad
altura = np.random.normal(170, 10, 50) # Media 170, desviación
estándar 10, tamaño 50
peso = np.random.normal(70, 15, 50) # Media 70, desviación estándar
15, tamaño 50
# Crear una figura y un eje
fig, ax = plt.subplots()
# Graficar los puntos con los datos
ax.scatter(altura, peso)
# Añadir títulos a los ejes y a la figura
ax.set_xlabel("Altura (cm)")
ax.set_ylabel("Peso (kg)")
ax.set_title("Gráfico de dispersión con matplotlib")
# Mostrar el gráfico
plt.show()
```

Cuestionario guia de lectura del material

1. ¿Qué es matplotlib y para qué sirve?
2. ¿Qué función se usa para crear un diagrama de dispersión con matplotlib?
3. ¿Qué módulo de matplotlib ofrece una interfaz cómoda y sencilla para añadir elementos a los gráficos?
4. ¿Cómo importar una librería al código Python?
5. ¿Cómo se puede guardar un gráfico creado con matplotlib en un archivo de imagen?

Bibliografía:

1. Bagnato, J. i., (2020). Aprende Machine Learning en Español: Teoría + Práctica Python. Editorial Leanpub.
2. Britos, P. V., & García Martínez, R. (2009). Propuesta de Procesos de Explotación de Información. In *XV Congreso Argentino de Ciencias de la Computación*.
3. Chazallet, S. (2016). Python 3: los fundamentos del lenguaje. Ediciones ENI.
4. Geron, A., (2020). Aprende Machine Learning con Scikit-Learn, Keras y Tensor Flow: Conceptos, herramientas y técnicas para construir sistemas inteligentes. Editorial O'Reilly y Anaya
5. Hilera, J. R. y Martinez, V. J. (2000) Redes Neuronales Artificiales. Fundamentos. modelos y aplicaciones. Alfaomega Ed
6. JIMÉNEZ, R. O., (2021). Python a fondo. Editorial Marcombo
7. Kuna, H. D., Caballero, S., Rambo, A., Meinl, E., Steinhilber, A., Pautsch, J., ... & Villatoro, F. (2010). Avance en procedimientos de la explotación de información para la identificación de datos faltantes, con ruido e inconsistentes. In *XII Workshop de Investigadores en Ciencias de la Computación*.
8. Kuna, H., Pautsch, G., Rey, M., Cuba, C., Rambo, A., ... & Villatoro, F. (2012). Obtenido de COMPARACIÓN DE LA EFECTIVIDAD DE PROCEDIMIENTOS DE LA EXPLOTACIÓN DE INFORMACIÓN PARA LA IDENTIFICACIÓN DE OUTLIERS EN BASES DE DATOS:
9. Matthes, E. (2021) Curso intensivo de Python, 2ª edición: Introducción práctica a la programación basada en proyectos. Editorial Anaya Multimedia
10. Ochoa, M. A. (2004). Herramientas inteligentes para explotación de información. Trabajo Final Especialidad en Ingeniería de Sistemas Expertos url: https://ri.itba.edu.ar/server/api/core/bitstreams/a848d640-0277-459d-9104-b37017309d31/content