UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

UNIDAD DIDÁCTICA I

MACHINE LEARNING

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo I – Bases y Herramientas

Unidad I – Instalaciones y conceptos previos.

Presentación:

Machine Learning es la actividad de adquirir datos y descubrir patrones de eventos que nos permitan generar modelos destinados a permitirle a las computadoras predecir comportamientos y en consecuencia tomar decisiones. Los modelos en mi opinión por más complejos que se tornen, nunca podrán sustituir la intuición y la creatividad humana, sin embargo dada la gran facilidad de acceder a la información que hoy día poseemos, es posible predecir comportamientos en áreas muy diferentes, que incluso incluyen la vida privada de las personas. Tenemos que tener en cuenta que nos encontramos en una era en la cual uno de los pilares es la conectividad total, es decir que se pretende para bien o para mal lograr que cada ser vivo o elemento de control posea un IP desde la cual poder controlar comportamientos o estados de funcionamiento. El análisis de estos datos para una mente humana puede ser prácticamente inaccesible, pero si combinamos el potencial de análisis de datos y las herramientas disponible por un lenguaje como python con un uso coherente de habilidades matemáticas como la probabilidad y la estadística, podemos llegar a la implementación de modelos aplicables, a caso concretos que puedan ser ajustados con el transcurso del tiempo y la experiencia generada en base a los datos recolectados.

En esta unidad comenzaremos a instalar y a utilizar las herramientas básicas que nos permitan formar las bases de nuestros análisis.

Objetivos:

Que los participantes:

Realicen la instalación de los programas con los que vamos a trabajar.

Se introduzcan en los conceptos básicos del aprendizaje automático.

Bloques temáticos:

1.- Introducción.

2.- Instalaciones.

3.- Matrices y Arrays.

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Introducción

El aprendizaje realizado por las máquinas no puede sustituir actualmente el conocimiento aplicado que pueden desarrollar un grupo de expertos dentro de una determinada área, pero puede potenciar los logros alcanzados, por ejemplo permitiendo:

- Procesar volúmenes de datos inmanejables por una persona.
- Encontrar patrones dentro de los datos.
- Tomar decisiones en fracciones de segundos.

Las máquinas pueden tornarse de esta forma en una herramienta que nos puede incluso ayudar a salvar vidas en áreas muy diversas, como pueden ser por ejemplo:

- **La salud** al utilizar robots que permitan a un cirujano prever situaciones en medio de una operación.
- **El ordenamiento vial** al poner en circulación autos inteligentes que puedan prevenir un accidente en el caso de que una persona se duerma o sufra un ataque cardiaco perdiendo de esta forma el control del auto.

Las áreas de aplicación son muchas, desde fines altruistas a fines bélicos, pero todas tienen detrás la misma metodología, utilizando modelos de aprendizaje automático basados en estadísticas, teoría de probabilidad, álgebra lineal y la optimización continua del modelo mediante construcciones matemáticas.

Realizar todo desde cero en cada trabajo sería una tarea muy ardua, afortunadamente hoy en día con la ayuda de python contamos con una gran cantidad de herramientas que realizan parte del trabajo por nosotros.

Tipos de tareas.

Todo modelo debe contar con una entrada de datos, los cuales pueden ser de diferente tipo, como datos de texto, números, imágenes e incluso audio o video. Y contar con una salida que represente por ejemplo la variación de una temperatura dada por un valor flotante o la asignación de un valor asociado al reconocimiento de una imagen que pueda retornar las huellas digitales de una persona con determinadas características biométricas.

El objetivo principal del aprendizaje automático es analizar y construir algoritmos que puedan en base a datos históricos, aprender y realizar predicciones sobre nuevos datos de entrada y que sean capaces de definir que tan bien el modelo realizado está aprendiendo.

En base al tipo de datos de entrada podríamos realizar una clasificación del tipo de tarea de aprendizaje en:

- **Aprendizaje no supervisado**: Utilizado para detectar por ejemplo patrones de comportamiento en base a datos que no traen asociados una descripción adjunta.

- **Aprendizaje supervisado**: Utilizado para, por ejemplo, crear una plataforma para realizar encuestas que clasifican los datos y preferencias de las personas de forma tal de poder encontrar patrones que permitan generar una regla que asigne a una determinada preferencia una salida específica. En este caso se pueden utilizar técnicas de regresión lineal o clasificación para encontrar tendencias.

- **Aprendizaje de retroalimentación**: En este caso los datos obtenidos del modelo proporcionan una retroalimentación dotando al sistema con la capacidad de adaptarse de forma dinámica de forma de obtener un objetivo final. En base a la retroalimentación obtenida, el sistema evalúa su desempeño y reacciona en consecuencia.

Núcleo del aprendizaje automático.

Lo bueno de los datos es que hay muchos en el mundo. Lo malo es que es difícil procesar estos datos. Los desafíos provienen de la diversidad y el ruido de los datos. Nosotros, los humanos, usualmente procesamos datos que llegan a nuestros oídos y ojos. Estas entradas se transforman en señales eléctricas o químicas. En un nivel muy básico, las computadoras y los robots también funcionan con señales eléctricas. Estas señales eléctricas se traducen entonces en unos y ceros. Sin embargo, nosotros vamos a programar en Python y normalmente representamos los datos como números, imágenes o textos. En realidad, las imágenes y el texto no son muy convenientes, por lo que necesitamos transformar las imágenes y el texto en valores numéricos.

Especialmente en el contexto del aprendizaje supervisado, tenemos un escenario similar al de estudiar para un examen. Tenemos un conjunto de preguntas de práctica y los

exámenes reales. Deberíamos poder responder las preguntas del examen sin saber las respuestas a ellas. Esto se llama **generalización** aprendemos algo de nuestras preguntas de práctica y, con suerte, podemos aplicar el conocimiento a otras preguntas similares. En el aprendizaje automático, estas preguntas de práctica se denominan **conjuntos de entrenamiento** o **muestras de entrenamiento**. Son de donde los modelos derivan patrones. Y los exámenes reales son **conjuntos de prueba** o **muestras de prueba**. Son donde se aplican los modelos y se extrae cuán compatibles son. A veces, entre las preguntas de práctica y los exámenes reales, tenemos exámenes simulados para evaluar qué tan bien vamos a hacer en los reales y para ayudar a la revisión. Estos exámenes simulados se denominan **conjuntos de validación** o **muestras de validación** en aprendizaje automático. Nos ayudan a verificar qué tan bien se comportan los modelos en una configuración simulada, luego ajustamos los modelos en consecuencia para lograr mayores éxitos.

Flujo de trabajo

La creación de aplicaciones de aprendizaje automático, aunque es similar en muchos aspectos al paradigma de la ingeniería estándar, difiere en un aspecto crucial: la necesidad de trabajar con datos como materia prima. El éxito de un proyecto dependerá, en gran parte, de la calidad de los datos que se adquiera, así como del manejo de dichos datos. Y cómo trabajar con datos cae en el dominio de la ciencia de datos, es útil comprender el flujo de trabajo de la ciencia de datos, el cual está compuesto por siete pasos:

1. Comprender el negocio
2. Adquisición
3. Inspección
4. Preparación
5. Modelado
6. Evaluación
7. Despliegue o puesta en producción

Realicemos un pequeño comentario sobre cada punto:

1 Comprender el negocio

Es necesario realizar previamente a iniciar el análisis, un estudio del negocio o emprendimiento al cual nos vamos a abocar y cuáles pueden ser las fuentes de datos

existentes o a desarrollar para retroalimentar con información precisa los modelos desarrollados.

2 - Adquisición

Los datos para aplicaciones de aprendizaje automático pueden provenir de varios tipos de fuentes, como un correo electrónico, una encuesta online un captcha o un sin número de opciones más, y en varios formatos, que pueden ser números, texto, imágenes, audio, etc. Independientemente de la fuente y del formato es crucial comprender bien cual es el contenido de los datos.

3 - Inspección

Una vez adquiridos los datos, el siguiente paso es inspeccionarlos. El objetivo principal en esta etapa es verificar la integridad de los datos, y la mejor manera de lograrlo es buscar cosas que sean imposibles o muy poco probables. Como ejemplo, si los datos tienen un identificador único, verifique que exista uno solo; si los datos se basan en el precio, verifique que siempre sea positivo; y cualquiera que sea el tipo de datos, verifique los casos más extremos. Es una buena práctica realizar estadísticas simples en los datos y visualizarlos. Este paso tiene un impacto directo sobre la eficacia del modelo realizado.

4 - Preparación

Luego de inspeccionar los datos, debemos estar seguros de que se encuentran ordenados de forma de colocar en un formato que resulte fácil de modelar. En esta etapa es imprescindible comprender bien cuál es el tipo de datos con el cual estamos trabajando, así como las bibliotecas y algoritmos a utilizar.

5 - Modelado

Una vez que se completa la preparación de los datos, la siguiente fase es el modelado. Aquí, seleccionaremos un algoritmo apropiado y utilizaremos los datos para entrenar un modelo.

6 - Evaluación

En esta etapa debemos evaluar cómo responde el modelo, la forma de evaluación depende en gran medida del tipo de datos que estamos utilizando y del tipo de modelo

seleccionado, pero en líneas generales estamos buscando conocer que tan bien responde a las predicciones realizadas.

7 - Despliegue

Al final nos resta implementar el modelo en la práctica, el cual puede ubicarse dentro de una aplicación ya creada, una página web u otro tipo de forma de obtener datos del medio.

2. Instalaciones

En esta sección realizaremos la instalación de los programas básicos que veremos a lo largo de este curso con una pequeña reseña de su finalidad de manera que en las unidades posteriores nos podamos poner directamente a utilizarlos.

Instalación de anaconda

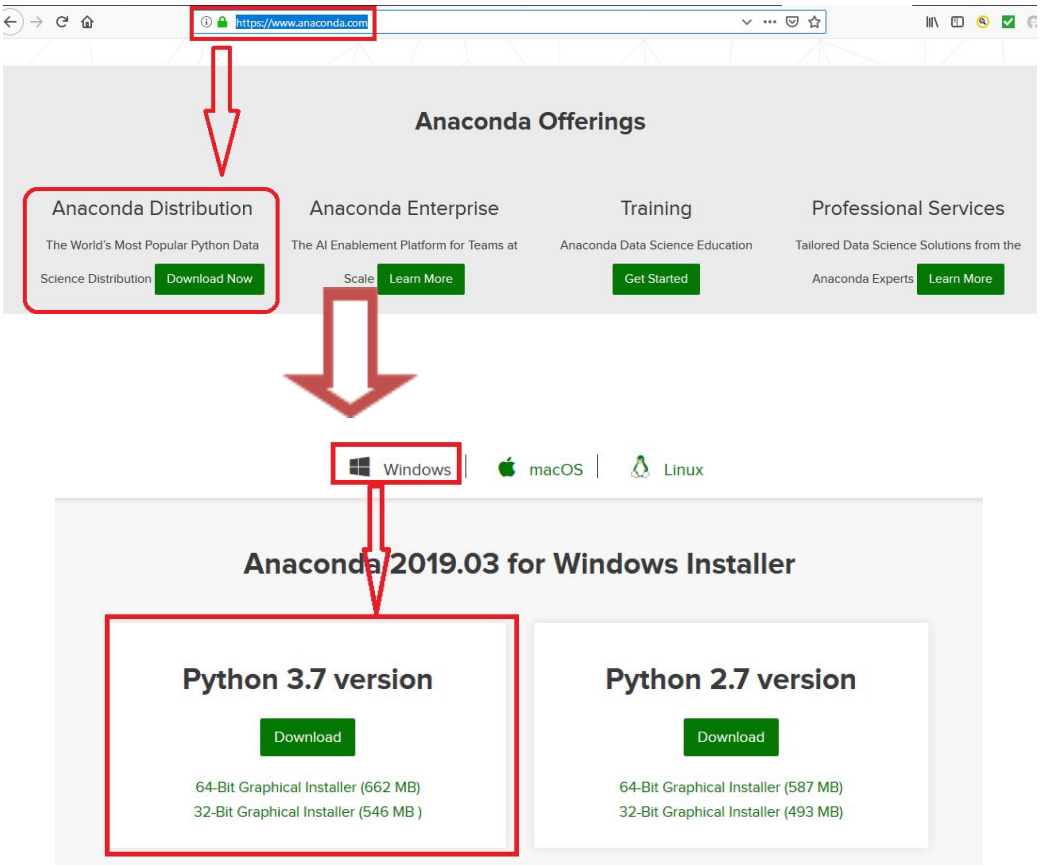
Anaconda es para nosotros, una herramienta indispensable, ya que nos va a permitir trabajar no solo con python, sino que además podremos interactuar con programas realizados en R, Julia y Octave. Estos lenguajes son un complemento indispensable al profundizar en el tema que nos ocupa.

- R es un lenguaje de programación para computación estadística y gráficos.
- Julia es utilizado en computación numérica y está diseñada para programar en la nube.
- Octave es para computación numérica y trabajos matemáticos.

Paso 1

Accedemos a: ***https://www.anaconda.com/***

E instalamos la distribución de anaconda



Al finalizar la instalación podemos ejecutar en el prompt de Anaconda el siguiente comando para obtener información sobre la instalación que acabamos de realizar:

conda info

Este comando nos muestra entre otras cosas la ubicación de la instalación, la cual por defecto es: C:\ProgramData\Anaconda3

Hay que tener la precaución de saber que el directorio “ProgramData” es un directorio oculto por defecto en Windows.

```
(base) C:\Users\juan>conda info

     active environment : base
    active env location : C:\ProgramData\Anaconda3
            shell level : 1
       user config file : C:\Users\juan\.condarc
 populated config files :
          conda version : 4.6.11
    conda-build version : 3.17.8
         python version : 3.7.3.final.0
       base environment : C:\ProgramData\Anaconda3  (read only)
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/win-64
                          https://repo.anaconda.com/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/r/win-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/msys2/win-64
                          https://repo.anaconda.com/pkgs/msys2/noarch
          package cache : C:\ProgramData\Anaconda3\pkgs
                          C:\Users\juan\.conda\pkgs
                          C:\Users\juan\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\juan\.conda\envs
                          C:\ProgramData\Anaconda3\envs
                          C:\Users\juan\AppData\Local\conda\conda\envs
               platform : win-64
             user-agent : conda/4.6.11 requests/2.21.0 CPython/3.7.3 Windows/10 Windows/10.0.17134
          administrator : False
             netrc file : None
           offline mode : False
```

Jupyter

Jupyter es una de las herramientas más útiles en el análisis de datos con python la cual viene dentro de las aplicaciones que instalamos junto con Anaconda. Cuando utilizamos jupyter, podemos cargar datos, transformarlos y modelarlos dentro de una única ventana, permitiendo probar el código y explorar ideas de forma rápida y fácil. Sumado a esto Jupyter nos permite documentar el código, agregando texto con formato de forma de incluso poder generar un reporte de lo realizado.

Básicamente Jupyter permite crear ventanas web que se ejecutan localmente y contienen código que puede ser editado en el momento tanto en python como en R,

Los portátiles Jupyter son aplicaciones web que se ejecutan localmente y contienen códigos en vivo, ecuaciones, figuras, aplicaciones interactivas y texto de Markdown. El lenguaje estándar es Python, y eso es lo que usaremos para este libro; sin embargo, tenga en cuenta que una variedad de alternativas son compatibles. Esto incluye el otro lenguaje de ciencia de datos dominante, R:

Para ver cómo funciona crearemos nuestra primera ventana siguiendo una serie de pasos:

Paso 1 – Creamos un directorio de trabajo

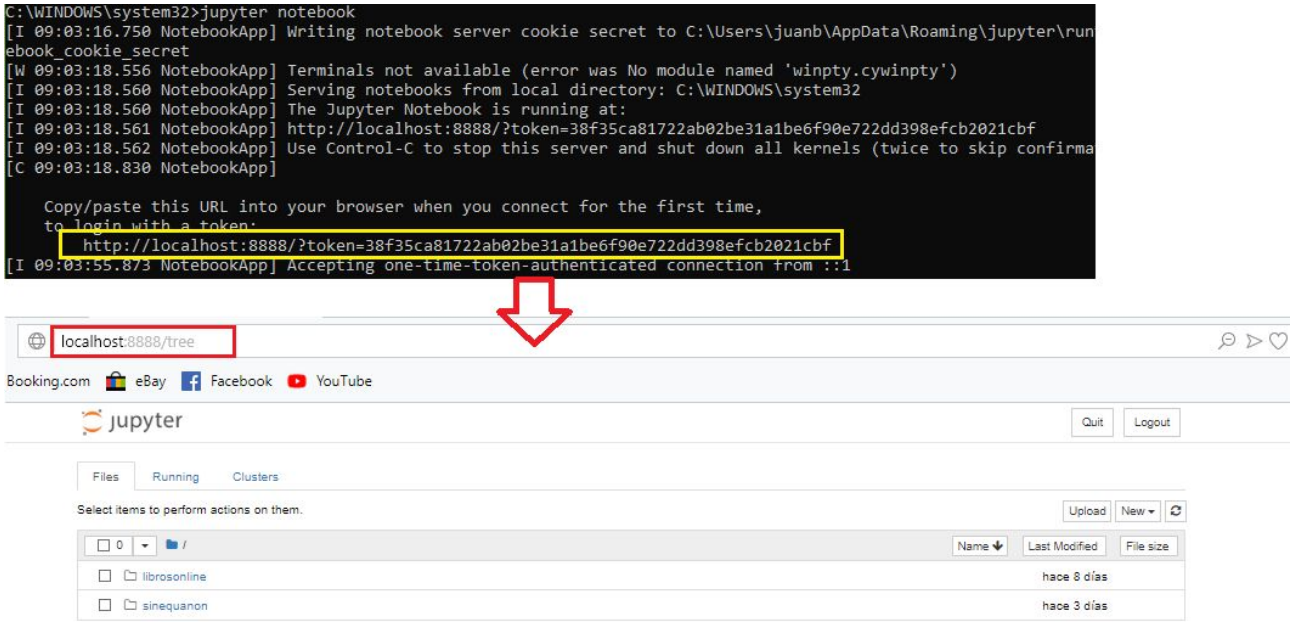
En mi caso he creado un directorio que voy a llamar “unidad1-jupyter

Paso 2 – Accedemos al directorio

Abrimos el prompt de Anaconda e ingresamos al directorio de trabajo desde la terminal, una vez dentro ejecutamos el siguiente comando.

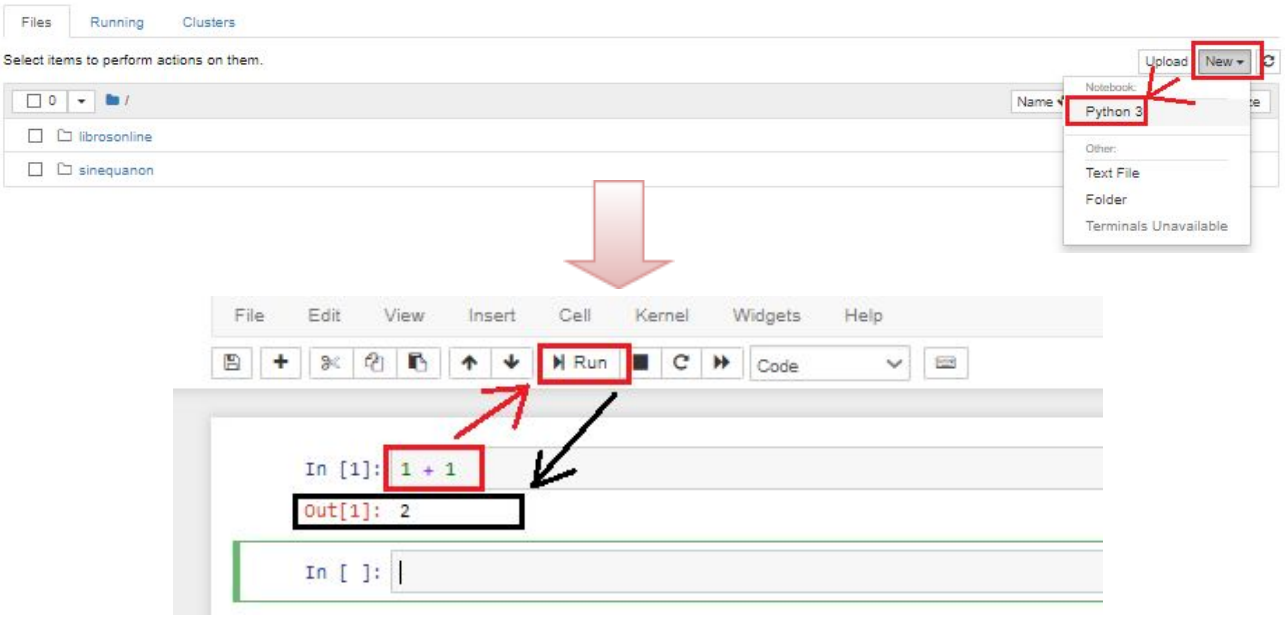
```
jupyter notebook
```

Observamos como en el cmd nos aparece la indicación de que se está ejecutando el servidor local en el puerto 8888 e inmediatamente se abre el explorador por defecto mostrando el tablero de trabajo.

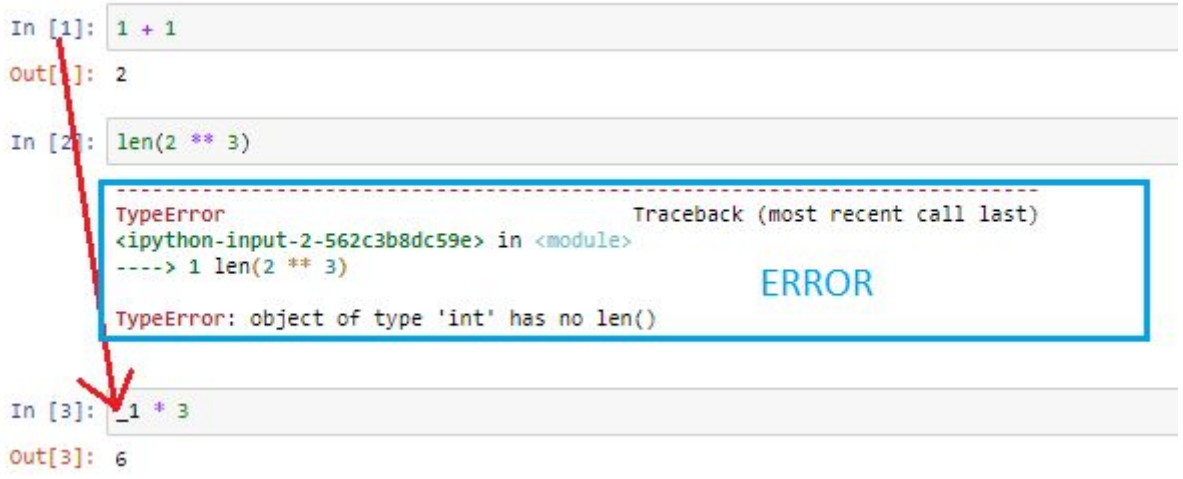


Paso 3 – Creamos un nuevo nootebok

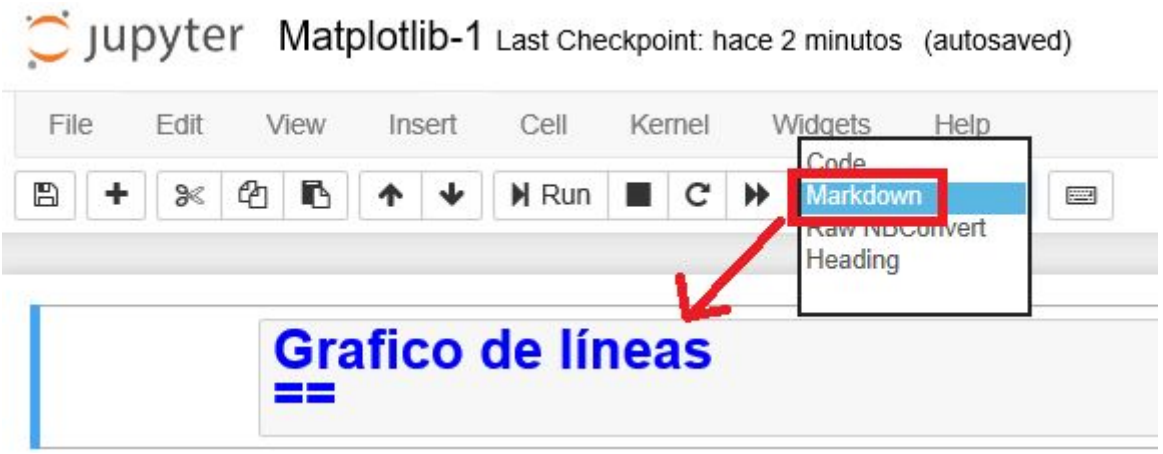
Creamos un nuevo notebook de Ipython presionando en new, los notebooks se organizan en celdas, y si escribimos en una celda y ejecutamos nos retorna el resultado agregando una nueva celda.



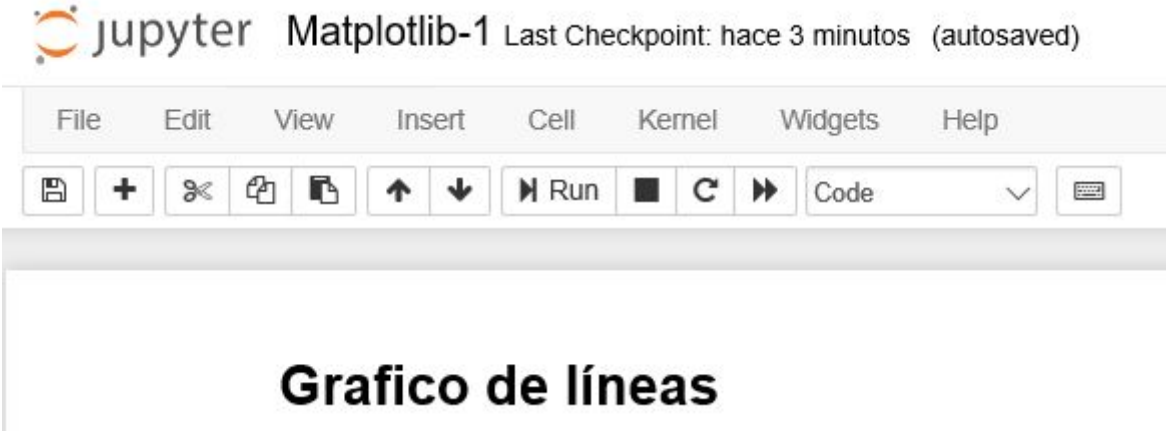
Es interesante utilizar jupyter en lugar de un editor por la gran cantidad de funcionalidades adicionales que podemos utilizar, por ejemplo un guión bajo, nos permite recuperar el resultado de una celda escrita anteriormente, mientras que si cometemos un error nos sale un mensaje preciso de la causa del error. En la siguiente imagen podemos ver como la línea 2 nos da un error, y como podemos referenciar a la línea 1 dentro de una línea de código ejecutable para utilizar un resultado previo.



También podemos agregar texto, como títulos y comentarios si cambiamos la entrada de tipo código (code) a Markdown y adicionamos algunos caracteres extra.



Luego de ejecutar la línea nos da:



Otros tipos de texto pueden ser:

Codigo agregado	
#Título	Título
Texto	Texto
__negrita__	Negrita

-Primera viñeta *Segunda viñeta *Tercera viñeta	Viñetas
> Comentario especial	Comentario especial

Luego de ejecutar nos queda:

Título

Contenido de texto

negrita

-Primera viñeta

*Segunda viñeta

*Tercera viñeta

> Comentario especial

numpy

NumPy es una de las piedras fundamentales que posee python en el área de análisis numérico que suele ser utilizado por matemáticos, científicos e ingenieros, así como por diversas plataformas en el área de inteligencia artificial y machine learning. Numpy permite trabajar con vectores y matrices de forma eficiente. Los vectores y matrices no son comparables a las listas de python aún cuando así lo parezcan a simple vista.

Aunque las listas de Python son muy fáciles de crear y manipular, no admiten operaciones vectorizadas, a diferencia de las matrices creadas con numpy. NumPy no solo es más eficiente en operaciones de matrices multidimensionales en comparación con las listas de

Python, sino que también proporciona muchos métodos matemáticos que pueden ser utilizados con solo importar la librería.

Pandas

Pandas es una librería de python que nos permite obtener datos de distinta procedencia como html, csv, json, pickle, sql. En el transcurso de las unidades iremos viendo como implementar **pandas**. Un ejemplo simple de cómo cargar un archivo csv podría ser:

```
import pandas as pd

dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1")

dat_csv
```

Matplotlib

Matplotlib es un paquete de Python para la visualización de datos, que nos permite trabajar con diferentes formatos de representación. Matplotlib nos brinda una interfaz simple orientada a objetos en la cual podemos trazar desde gráficos simples a una amplia variedad de gráficos e incluso aplicarlo en el desarrollo web. Podemos ver a continuación un ejemplo simple en donde a partir de las listas de datos x e y, creamos el gráfico con la ayuda de plot() y show.

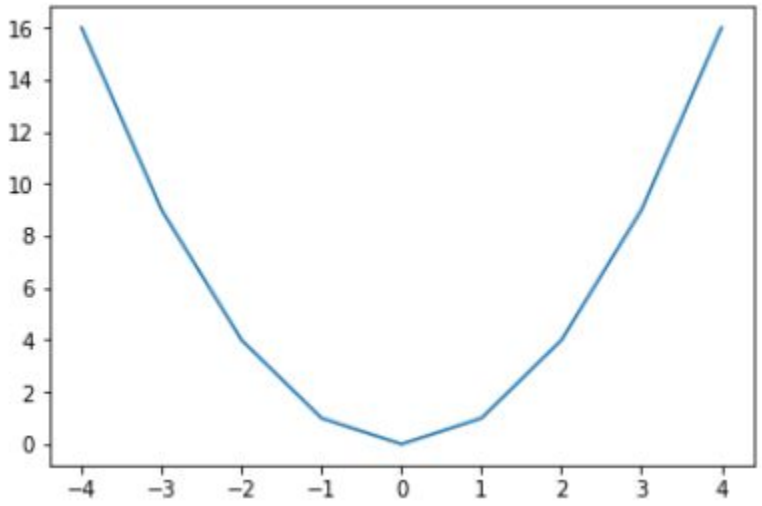
```
import matplotlib.pyplot as plt

x = list(range(-4 , 5))

y = [i**2 for i in x]

plt.plot(x,y)

plt.show
```



En la unidad 3 desarrollaremos cómo utilizar matplotlib para representar datos complejos.

scikit

Scikit-learn es un software gratuito y de código abierto que nos ayuda en el trabajo del aprendizaje automático, está construido completamente en Python y utiliza las librerías de NumPy y SciPy. Scikit se ha vuelto muy popular, ya que permite implementar con bastante rapidez la mayoría de los algoritmos utilizados en este tema.

3. Matrices y Arrays

Al trabajar con algebra lineal utilizamos el concepto de matriz, entendiéndose a una matriz como un arreglo bidimensional de números. Las matrices son designadas por letras mayúsculas y poseen dos subíndices los cuales hacen referencia a la cantidad de elementos de cada fila y columna.

$$A_{m\,x\,n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

La matriz anterior "A" se dice que tiene una dimensión de "m" elementos por "n" elementos y también puede ser expresada utilizando dos subíndices (i, j) como:

$$A = (a_{ij})_{m\,x\,n}.$$

En donde **i varía entre 1 y m** y **j varía entre 1 y n**.

El proceso de acceso a los elementos de una matriz con conjuntos de enteros se denomina indexación.

En nuestro trabajo las matrices son un elemento fundamental, y al trabajar con software utilizamos el nombre de **array** como una abstracción de una matriz. Python no posee un objeto tipo array, por lo cual nos vamos a valer en principio de la librería **numpy** para poder trabajar con arrays. Hay que tener en cuenta que un array no es una lista, aun cundo a simple vista puedan existir similitudes.

En numpy expresaremos un array como A [i, j], esta librería admite todas las operaciones de matrices analizadas en álgebra lineal.

Creamos un array con numpy

Para crear un array podemos importamos la librería de numpy y utilizamos el método "array()" de forma de pasar todos los elementos del array.

float/main.py

```
1  import numpy as np
2  A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9,10, 11, 12]])
```

```
3  print(A)
4  print(type(A))
```

Como podemos ver jupyter nos retorna el resultado al imprimir en pantalla el array A y lo representa como una matriz.

```
In [4]: import numpy as np
        A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9,10, 11, 12]])
        print(A)
        print(type(A))

        [[ 1  2  3  4]
         [ 5  6  7  8]
         [ 9 10 11 12]]
        <class 'numpy.ndarray'>
```

Suma de matrices

Para poder realizar la suma de dos matrices, estas deben poseer el mismo tamaño, ya que la suma de matrices de **m** filas y **n** columnas se debe realizar sumando los elementos que ocupan la misma posición dentro de la matriz según se puede ver a continuación.

$$C_{m\,x\,n} = A_{m\,x\,n} + B_{m\,x\,n} = \left(a_{ij} + b_{ij}\right)$$

En donde $1 \leq i \leq m$ y $1 \leq j \leq n$

$$C_{m\,x\,n} = \begin{bmatrix} A_{11} + B_{11} & \cdots & A_{1n} + B_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} + B_{m1} & \cdots & A_{mn} + B_{mn} \end{bmatrix}$$

Nota: La suma de matrices es conmutativa.

Producto de matriz por escalar

El producto de una matriz **A** por un escalar **α** se obtiene multiplicando cada elemento de la matriz por el escalar.

$$\propto . A = \propto . \left(a_{ij}\right)$$

Nota: Esta operación también es conmutativa.

Matriz transpuesta

La matriz transpuesta de una matriz A se define intercambiando filas por columnas y se indica como A^T . Veamos un ejemplo para comprenderlo mejor:

$$A = (1\ 2\ 3\ 4\ 5\ 6\ )$$

$$A^T = \ (1\ 3\ 5\ 2\ 4\ 6\ )$$

Multiplicación de matrices

Dada dos matrices $A_{m\,x\,n}$ y $B_{n\,x\,p}$ se define el producto $A.B$ como la matriz de dimensiones **m x p** en donde el elemento de la posición de la fila **i** y columna **j** es resultado del producto de los vectores fila **i** de **A** y columna **j** de **B**.

Veamos un ejemplo:

Dadas las matrices A y B

$$A = (1\ 2\ 3\ 4\ 5\ 6\ ) \qquad B = \ (1\ 3\ 5\ 2\ 4\ 6\ )$$

$$A_{3\,x\,2}.B_{2\,x\,3} = \begin{pmatrix} 1*1+2*2 & 1*3+2*4 & 1*5+2*6 \\ 3*1+4*2 & 3*3+4*4 & 3*5+4*6 \\ 5*1+6*2 & 5*3+6*4 & 5*5+6*6 \end{pmatrix} = \begin{pmatrix} 5 & 11 & 17 \\ 11 & 25 & 39 \\ 13 & 39 & 61 \end{pmatrix} = C_{3\,x\,3}$$

Propiedades del producto de matrices

Algunas propiedades de matrices son:

- El producto de matrices no es necesariamente conmutativo.
- El producto de matrices es asociativo.
- El producto de matrices es distributivo.
- El producto de matrices tiene elemento neutro.

Vector

Un vector es una matriz de una dimensión, el cual puede ser representado por $A_{m\,x\,1}$ o $A_{1\,x\,n}$ en el caso de que se trate de un vector columna o fila respectivamente.

Además, hay algunas matrices especiales importantes que veremos a continuación:

Matriz cero

La matriz cero o matriz nula posee todos sus componentes iguales a cero, en una matriz 0, es opcional agregar subíndices:

$$0_{m\,x\,n} = \begin{bmatrix} 0_{11} & \cdots & 0_{1n} \\ \vdots & \ddots & \vdots \\ 0_{m1} & \cdots & 0_{mn} \end{bmatrix}$$

Matriz identidad

La matriz identidad posee unos en su diagonal, mientras que todos los demás son ceros:

$$I_{m\,x\,n} = \begin{bmatrix} 1_{11} & \cdots & 0_{1n} \\ \vdots & \ddots & \vdots \\ 0_{m1} & \cdots & 1_{mn} \end{bmatrix}$$

Cuando multiplicamos una matriz A por la matriz de identidad, el resultado será igual a A:

$$A \, x \, I = A$$

Una matriz de identidad es muy útil para calcular el inverso de una matriz. Cuando multiplicas cualquier matriz dada con su inverso, el resultado será una matriz de identidad:

$$A^{-1} \, x \, A = I$$

Bibliografía utilizada y sugerida

Libros

Mark Lutz, O'Reilly (2013, 02). Programming Python 5th Edition

By Yuxi (Hayden) Liu, (2019, 02). Python Machine Learning By Example, Second Edition

Alexander Combs, Michael Roman, (2019, 02). Python Machine Learning Blueprints, Second Edition

Manual online

(2019, 01). Obtenido 01, 2019, de http://www.numpy.org/
(2019, 01). Obtenido 01, 2019, de http://es.python-requests.org/es/latest/
(2019, 01). Obtenido 01, 2019, de http://ipython.org/
(2019, 01). Obtenido 01, 2019, de https://pandas.pydata.org/
(2019, 01). Obtenido 01, 2019, de https://matplotlib.org/
(2019, 01). Obtenido 01, 2019, de https://seaborn.pydata.org/
(2019, 01). Obtenido 01, 2019, de https://scikit-learn.org/stable/
(2019, 01). Obtenido 01, 2019, de https://www.tensorflow.org/learn

Lo que vimos

En esta unidad hemos visto algunos conceptos básicos del aprendizaje automático y realizamos la instalación de los programas que vamos a utilizar a lo largo del curso.

Lo que viene:

En la siguiente unidad analizaremos cómo tomar datos de diferentes procedencias con pandas.