Capítulo 1

Introducción

Comenzamos nuestro curso de Machine learning con Python realizando una introducción del proceso paso a paso que se debe de utilizar en el mundo del modelado predictivo. Veremos como el lenguaje Python es un potente lenguaje de programación que nos permitirá analizar y tomar decisiones sobre cualquier conjunto de datos de manera muy sencilla con los conocimientos ya adquiridos de programación que ya deberemos de poseer.

En este sentido en esta primera unidad veremos:

- **Machine learning**. Explicaremos los conceptos básicos de Machine Learning y dónde se enfoca este curso.
- **Python para Machine Learning**. Python es un ecosistema muy grande entre librerías y recursos que pone a nuestra disposición, los cuales vamos a ver en detalle los que utilizaremos en este curso.
- **Curso rápido de Python**. Haremos un recorrido de la sintaxis de Python, conociendo los conceptos básicos como asignaciones, estructuras de datos, estructuras de control, funciones y la importancia de los paquetes.

Ver Vídeo de este apartado

```
1    1.1. Descripción de los contenidos.
```

Entorno de desarrollo Jupyter Notebook. Machine Learning funcionamiento y el proceso para llegar a partir de la base de datos al "data analytic"->"treatment"->"model"->"Forecasting"...

1.1. Machine learning

Normalmente, cuando escuchamos la palabra Data Analytics o machine learning, automáticamente la asociamos a modelos matemáticos complejos y programación en diferentes lenguajes. Esta primera impresión nos lleva a tener que el aprendizaje de esta temática si no disponemos del perfil. Pues bien, en este curso vamos a romper este mito y nos daremos cuenta que no hay que ser bueno con las matemáticas ni tampoco ser un gran programador para poder aprender y utilizar machine learning en nuestras necesidades.

En este contexto, nos daremos cuenta que en este curso no vamos a profundizar en la teoría (base matemática) ni tampoco en la gran cantidad de parámetros que existen dentro de los algoritmos de machine learning, simplemente vamos a aprender de la importancia de este y utilizarlo de manera muy sencilla entendiendo la gran cantidad de información que nos provee.

1.1.1. Machine learning con Python

Antes de nada, vamos a enfocar el siguiente curso dentro de la temática de Machine learning. Dentro de este campo el curso "Machine learning con Python" se encuentra dentro del modelado predictivo del machine learning muy utilizado en el ámbito académico y profesional. Por tanto, veremos como este lenguaje destaca de entre todas las opciones por facilitar enormemente nuestro trabajo con un amplio abanico de opciones dentro de la minería de datos.

Si hacemos una diferencia con la estadística tradicional puede decirse que este tipo de estadística tiene la funcionalidad principal de llegar a comprender los datos mientras que el modelado predictivo se refiere a la técnica que tiene por objeto descubrir patrones de comportamiento de nuestros datos, en este caso tabulares como hojas de cálculo, para tener predicciones más precisas.

Capítulos

Una vez que sepa cómo completar una tarea discreta utilizando la plataforma y obtener un resultado de manera confiable, puede hacerlo una y otra vez en proyecto tras proyecto. Comencemos con una visión general de las tareas comunes en un proyecto de machine learning. Un proyecto de machine learning de modelado predictivo se puede dividir en 6 tareas de nivel superior:

1. **Introducción**: Comenzamos nuestro curso de Machine Learning con Python realizando una introducción del proceso paso a paso que se debe de utilizar en el mundo del modelado predictivo con el lenguaje Python. Veremos como Python es un potente lenguaje de programación que nos permitirá analizar y tomar decisiones sobre cualquier conjunto de datos de manera muy sencilla con los conocimientos ya adquiridos de programación que ya deberemos de poseer.

2. **Análisis de datos**:En esta tercera unidad, se estudiarán una de las partes más importantes para un científico de datos, que es el análisis y tratamiento de los datos. Para ello, comenzaremos analizando la estadística descriptiva para así poder realizar técnicas de procesamiento de los atributos, así como las diferentes transformaciones de nuestros datos para poder obtener un mejor rendimiento de nuestros algoritmos.

3. **Tratamiento de datos**: En esta unidad comparamos los diferentes resultados que tienen los algoritmos, diferenciando si nuestro problema es un problema de regresión (valor numérico a predecir) o de clasificación (valor de clase a predecir). Finalmente, aprenderemos a comparar los diferentes resultados de los algoritmos para así crear un modelo más robusto en la fase de predicción.

4. **Fase de modelado**: Una vez realizado un preprocesamiento de los datos debemos comprenderlos antes de modelarlos. Para poder comprenderlos existen técnicas como feature selection nos permitirán seleccionar las variables más significativas en los resultados. Posteriormente,

aprenderemos sobre los algoritmos más importantes de machine learning ya que tenemos que comparar el rendimiento de los algoritmos para buscar al mejor candidato.

5. **Fase de optimización y forecasting**: Antes de trabajar con proyectos reales en machine learning, es importante trabajar la fase de optimización, para obtener mejores resultados de los algoritmos de machine learning, y forecasting, para trabajar las predicciones de nuestros datos no etiquetados. En este sentido las secciones principales de trabajo en esta sección serán.

6. **Proyectos de machine learning**: Una vez visto todos los conceptos teórico/prácticos, trabajamos un proyecto completo de machine learning, es decir, de inicio a fin, evaluando cada una de las partes intermedias a tener en cuenta.

Estas lecciones están diseñadas para leerse de principio a fin en orden, y le muestran exactamente cómo completar cada tarea en un proyecto de machine learning de modelado predictivo. Por supuesto, puedes dedicarte a lecciones específicas más tarde para refrescarte. Algunas lecciones son demostrativas y muestran cómo utilizar técnicas específicas para una tarea de machine learning común (por ejemplo, carga de datos y preprocesamiento de datos). Otros están en un formato de tutorial, que se desarrolla a lo largo de la lección para culminar en un resultado final (por ejemplo, ajuste de algoritmos y métodos de conjunto).

1.1.2. Resultados del curso

La realización del curso "Machine Learning con Python" le abrirá un amplio abanico de posibilidades a la hora de tomar decisiones basadas en el aprendizaje, es decir, de nuestro conjunto de datos. En este sentido, pasaremos de ser interesados por el machine learning a creadores de nuevo conocimiento basándonos en las técnicas de machine learning. Para ello, este curso le asignará los siguientes resultados específicos:

- Trabajar con conjuntos de datos de tamaño pequeño/mediano.
- Aprender las técnicas de modelado predictivo con Python.
- Comprender que es la minería de datos y aplicarla a un conjunto de datos específico.
- Analizar e interpretar un conjunto de datos y la correlación entre ellos para mejorar las predicciones.
- Comprender y analizar la fase del análisis de datos previos al modelado algorítmico en machine learning.
- Realizar modelos algorítmicos robustos con una optimización de sus hiperparámetros para la fase de predicción.
- Desarrollar y analizar proyectos de machine learning como regresión, clasificación y multiclase.

1.1.3. Proceso de aprendizaje

Lo más importante cuando estamos aprendiendo a trabajar con machine learning es cómo entregar nuestro de resultado de manera confiable y óptima. Esto quiere decir, que dado un problema de machine learning, deberemos aprender cómo trabajar a través de él y entregar un conjunto de predicciones o cómo entregar un modelo robusto que pueda generar predicciones de manera confiable, es decir, no solo predicciones, sino predicciones precisas de manera sólida y confiable. Por ello, siempre encontrará los siguientes pasos:

1. Definición del problema.
2. Preparación de los datos.
3. Evaluación de un conjunto de algoritmos (de diversa taxonomía).
4. Mejorar los resultados en la fase de optimización (tuning).
5. Finalizar el modelo y presentar los resultados.

1.1.4. Por qué utilizar Python

Esto es lo que NO debes hacer cuando comienzas a estudiar aprendizaje automático en Python.

1. Sea realmente bueno en la programación de Python y la sintaxis de Python.

2. Estudie profundamente la teoría y los parámetros subyacentes para los algoritmos de aprendizaje automático en `scikit-learn`.

3. Evite o toque ligeramente todas las demás tareas necesarias para completar un proyecto real.

Creo que este enfoque puede funcionar para algunas personas, pero es una forma realmente lenta y indirecta de alcanzar su objetivo. Le enseña que necesita pasar todo su tiempo aprendiendo cómo usar algoritmos individuales de aprendizaje automático. Tampoco le enseña el proceso de construcción de modelos predictivos de aprendizaje automático en Python que realmente puede usar para hacer predicciones. Lamentablemente, este es el enfoque utilizado para enseñar el aprendizaje automático que veo en casi todos los libros y cursos en línea sobre el tema.

1.1.5. Sobre el curso Machine Learning con Python

Este curso está enfocado a profesionales que desean saber cómo construir modelos de aprendizaje automático confiables y precisos en Python.

- **Este no es un curso de aprendizaje automático**. No entraremos en la teoría básica del aprendizaje automático (por ejemplo, inducción, compensación de sesgo-varianza, etc.). Se espera que esté familiarizado con los conceptos básicos de aprendizaje automático o que pueda aprenderlos usted mismo.

- **Este no es un curso de algoritmos**. No trabajaremos en los detalles de cómo funcionan los algoritmos específicos de aprendizaje automático (por

ejemplo, bosques aleatorios). Se espera que tenga algunos conocimientos básicos de algoritmos de aprendizaje automático o cómo aprenderlo usted mismo.

- **Este no es un curso de programación de Python**. No pasaremos mucho tiempo en la sintaxis y programación de Python (por ejemplo, tareas básicas de programación en Python). Se espera que sea un desarrollador que pueda aprender un nuevo lenguaje tipo C con relativa rapidez.

Todavía puede sacar mucho provecho de este curso si es débil en una o dos de estas áreas, pero puede tener dificultades para aprender el idioma o requerir más explicaciones de las técnicas.

Ver Vídeo de este apartado

```
1    1.2. Descripción del curso.
```

1.2. Python para modelado predictivo

El ecosistema de Python está creciendo y puede convertirse en la plataforma dominante para el aprendizaje automático. La razón principal para adoptar Python para el aprendizaje automático es porque es un lenguaje de programación de propósito general que puede usar tanto para I+D como para producción. En esta sección descubrirá el ecosistema de Python para el aprendizaje automático. Después de completar esta lección sabrás:

1. Python y su creciente uso para el aprendizaje automático.
2. SciPy y la funcionalidad que proporciona con NumPy, Matplotlib y Pandas.
3. scikit-learn que proporciona todos los algoritmos de aprendizaje automático.
4. Cómo configurar su ecosistema Python para el aprendizaje automático y qué versiones usar

1.2.1. Qué es Python

Python es un lenguaje de programación interpretado de propósito general. Es fácil de aprender y usar principalmente porque el lenguaje se enfoca en la legibilidad. La filosofía de Python se captura en el Zen de Python que incluye frases como:

- Hermoso es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Escaso es mejor que denso.
- La legibilidad cuenta.

Es un lenguaje popular en general, que aparece constantemente en los 10 principales lenguajes de programación en encuestas en StackOverflow[1]. Es un lenguaje dinámico y muy adecuado para el desarrollo interactivo y la creación rápida de prototipos con el poder de soportar el desarrollo de grandes aplicaciones. También se usa ampliamente para el aprendizaje automático y la ciencia de datos debido al excelente soporte de la biblioteca y porque es un lenguaje de programación de propósito general (a diferencia de R o Matlab). Por ejemplo, podemos encontrar numerosos estudios y encuenstas como el de IEEE Spectrum[2] o KDnuggets[3] o este informe extraído de una encuesta de Kaggle[4], en los que ponen a Python como el lenguaje de programación recomendable

[1] https://insights.stackoverflow.com/survey/2019
[2] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018
[3] https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html
[4] https://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/

si eres (o vas a ser) Científico de Datos. Estas páginas mencionadas son muy importantes para un Científico de Datos, tenlas siempre presentes.

Esta es una consideración simple y muy importante. Significa que puede realizar su investigación y desarrollo (averiguar qué modelos usar) en el mismo lenguaje de programación que usa para sus sistemas de producción. Simplificando enormemente la transición del desarrollo a la producción.

1.2.2. Librerías importantes en Python

Librería `SciPy`

`SciPy` es un ecosistema de bibliotecas de Python para matemáticas, ciencias e ingeniería. Es un complemento de Python que necesitará para el aprendizaje automático. El ecosistema `SciPy` se compone de los siguientes módulos principales relevantes para el aprendizaje automático:

- `NumPy`: una base para `SciPy` que le permite trabajar eficientemente con datos en matrices.
- `Matplotlib`: le permite crear gráficos y gráficos 2D a partir de datos.
- `Pandas`: herramientas y estructuras de datos para organizar y analizar sus datos.

Para ser efectivo en el aprendizaje automático en Python, debe instalar y familiarizarse con `SciPy`. Específicamente:

- Preparará sus datos como matrices `NumPy` para modelar en algoritmos de aprendizaje automático.
- Utilizará `Matplotlib` (y envoltorios de Matplotlib en otros marcos) para crear gráficos de sus datos.
- Utilizará `Pandas` para cargar, explorar y comprender mejor sus datos.

Librería `scikit-learn`

La biblioteca `scikit-learn` es cómo puede desarrollar y practicar el aprendizaje automático en Python. Está construido sobre y requiere el ecosistema `SciPy`. El nombre *scikit* sugiere que es un complemento o kit de herramientas de `SciPy`. El enfoque de la biblioteca son los algoritmos de aprendizaje automático para clasificación, regresión, agrupamiento y más. También proporciona herramientas para tareas relacionadas, como evaluar modelos, ajustar parámetros y preprocesar datos.

Al igual que Python y `SciPy`, `scikit-learn` es de código abierto y se puede utilizar comercialmente bajo la licencia BSD. Esto significa que puede aprender sobre el aprendizaje automático, desarrollar modelos y ponerlos en funcionamiento, todo con el mismo ecosistema y código. Una razón poderosa para usar `scikit-learn`.

1.2.3. Instalación del ecosistema Python

Python tiene un ecosistema muy grande de librerías y componentes ya que puede utilizarse para múltiples propósitos, en nuestro caso se utilizará como lenguaje de programación para Ciencia de Datos. En este sentido, y por fines prácticos del curso, se instalará el software Anaconda[5] ya que nos trae todo el ecosistema para Ciencia de Datos y utilizaremos Jupyter Notebook[6] como entorno de programación.

Anaconda

> "Anaconda es un distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue

[5] https://www.anaconda.com/distribution/
[6] https://jupyter.org/

> y administración de los paquetes de software. Las diferentes versiones de los paquetes se administran mediante el sistema de gestión de paquetes conda, el cual lo hace bastante sencillo de instalar, correr, y actualizar software de ciencia de datos y aprendizaje automático como ser Scikit-team, TensorFlow y SciPy."
>
> *Wikipedia, 2019.*

Por tanto, es open source, e incluye Python, una serie de paquetes orientados al stack científico y un gestor de paquetes y entornos virtuales llamado `conda`. Entre otras ventajas:

- Facilita la instalación de Python y paquetes, sobre todo en Windows
- Entornos virtuales centralizados, y referencias por nombres
- Interfaz mejorada sobre `pip`, mostrando información de todas las dependencias, tamaño de paquetes, etc.
- Realmente todos trabajan sobre Python, `pip` y `venv` o similares
- ...

Jupyter Notebook

> "Jupyter Notebook es un entorno computacional interactivo basado en la web para crear documentos Jupyter notebook. El término "cuaderno" puede hacer referencia coloquialmente a muchas entidades diferentes, principalmente la aplicación web Jupyter, el servidor web Jupyter Python o el formato de documento Jupyter según el contexto. Un documento de Jupyter Notebook es un documento JSON, que sigue un esquema versionado y que contiene una lista ordenada de celdas de entrada/salida que pueden contener código, texto (usando Markdown), matemáticas, tramas y medios enriquecidos, que generalmente terminan con "`.ipynb`" extensión."
>
> *Wikipedia, 2019.*

En este sentido podemos extraer las siguientes características:

- Es un shell interactivo de Python, con soporte para gráficos e interfaces gráficas de usuario
- Jupyter Notebook es popularmente conocido como iPython notebook (nació a partir de ese proyecto)
- JuPyteR: Julia, Python y R (los tres lenguajes para los que inicialmente fue diseñado). Ahora soporta más de 40
- Lleva iPython un paso más allá: proporciona una web interactiva sobre la que utilizar Python.
- Se despliega como un servidor web y se accede por el navegador.
- Es una herramienta muy cómoda para el prototipado de software, visualización de resultados, análisis exploratorio de datos, etc.
- Para el desarrollo de software más complejo, librerías, etc. existen IDEs más potentes, como PyCharm[7].

Ver Vídeo de este apartado

```
1    1.3. Herramientas de trabajo.
```

1.2.4. Paquetes

Los paquetes o librerías son la forma en que se distribuye el código Python de terceros. Estos paquetes pueden descargarse e instalar en nuestro entorno de Jupyter Notebook con la función '`!pip install package`'. Veamos une ejemplo con el paquete más importante en el desarrollo de este curso, este es el paquete `scikit-learn`, que es el paquete que nos permitirá utilizar tanto algoritmos como funciones relacionadas al machine learning, ver Código 1.1.

[7] https://www.jetbrains.com/pycharm/

```
> # Instalar un paquete
> !pip install scikit-learn
```

CÓDIGO 1.1: Instalar el paquete scikit-learn

Llegados a este punto, realice la instalación en Jupyter Notebook de los paquetes `pandas`, `numpy`, `scikit-learn` y `matplotlib`.

Para poder utilizar las funciones que nos pone a nuestra disposición el paquete deberemos cargarlo en nuestro programa mediante '`import package`' o '`from package import function`'. Por ejemplo, puede cargar el paquete `Pandas` de la siguiente manera, ver Código 1.2.

```
> # Cargar un paquete
> import pandas
> from pandas import read_csv
```

CÓDIGO 1.2: Cargar un paquete

Así mismo puede poner acrónimos con la palabra reservada '`as`', ver Código 1.3. En este caso, solamente se recompienda utiliza las palabras por convecionalismo, estas son: `np` para `numpy`, `pd` para `pandas` y `ptl` para `matplolib`.

```
> # Poner acronimo
> import pandas as pd
```

CÓDIGO 1.3: Poner acrónimo a una librería

Así mismo, se recomienda utilizar siempre `import` ya que hay veces que las funciones con mismo nombre se encuentran en diferentes librerías y sin indicar la librería puede llevar a problemas en el código, ver Código 1.4.

```
> # ejemplo de uso de libreria
> import pandas as pd
# load dataset
> url = 'https://archive.ics.uci.edu/ml/machine-learning-databases
/abalone/abalone.data'
```

```
> dataframe = pd.read_csv(url, header=None)
> print(dataframe)
```

CÓDIGO 1.4: Ejemplo de uso de librería

Abrir Jupyter Notebook

```
    Extra - Tutorial de Jupyter.
```

Ver Vídeo de este apartado

```
    1.4. Extra - Tutorial de Jupyter.
```

Abrir Jupyter Notebook

```
    1.2. Python para MD.
```

Ver Vídeo de este apartado

```
    1.5. Python para Modelado predictivo.
```

1.3. Curso rápido en Python y `SciPy`

No necesita ser un desarrollador de Python para comenzar a usar el ecosistema de Python para el aprendizaje automático. Como desarrollador que ya sabe cómo programar en uno o más lenguajes de programación, puede aprender un nuevo lenguaje como Python muy rápidamente. Solo necesita conocer algunas propiedades del idioma para transferir lo que ya sabe al nuevo lenguage. Después de completar esta sección sabrás:

- Cómo navegar por la sintaxis del lenguaje Python.
- Conocimientos suficientes NumPy, Matplotlib y Pandas para leer y escribir scripts Python de aprendizaje automático.
- Una base desde la cual construir una comprensión más profunda de las tareas de aprendizaje automático en Python.

Abrir Jupyter Notebook

```
1.3. Curso rápido de Python.
```

1.3.1. Sintaxis en Python

Al comenzar en Python, necesita conocer algunos detalles clave sobre la sintaxis del lenguaje para poder leer y comprender el código de Python. Esto incluye:

1. Asignación.
2. Control de flujo.
3. Estructuras de datos.
4. Funciones.

Cubriremos cada uno de estos temas a su vez con pequeños ejemplos independientes que puede escribir y ejecutar. Recuerde, el espacio en blanco tiene significado en Python.

1. Asignación

Como programador, la asignación y los tipos no deberían sorprenderte.

Strings

Observe cómo puede acceder a los caracteres en la cadena utilizando la sintaxis de la matriz, ver el Código 1.5.

```
> # Strings
> data = 'hello world'
> print(data[0])
> print(len(data))
> print(data)

```

```
h
11
hello world
```

CÓDIGO 1.5: Ejemplo de String

Números

Los números se manejan de manera muy sencilla, asignando cualquier tipo, ver el Código 1.6.

```
> # Numbers
> value = 123.1
> print(value)
> value = 10
> print(value)

123.1
10
```

CÓDIGO 1.6: Ejemplo de Numbers

Boolean

Los Boolean pueden tomar dos valores `True` o `False`, ver el Código 1.7.

```
> # Multiple Assignment
> a, b, c = 1, 2, 3
> print(a, b, c)

1 2 3
```

CÓDIGO 1.7: Ejemplo de Boolean

Asignación Múltiple

Esto también puede ser muy útil para desempacar datos en estructuras de datos simples, ver el Código 1.8.

```
> # Boolean
> a = True
> b = False
> print(a, b)

True False
```

CÓDIGO 1.8: Ejemplo de Asignación Múltiple

Valor `None`

Así mismo podemos asignar un valor tipo `None`, ver el Código 1.9.

```
> # No value
> a = None
> print(a)

None
```

CÓDIGO 1.9: Ejemplo de Asignación None

Ver Vídeo de este apartado

```
    1.6. Python - Asignaciones.
```

2. Control de flujo

Hay tres tipos principales de control de flujo que debe aprender: condiciones *If-Then-Else*, *For-Loops* y *While-Loops*.

Condicional *If-Then-Else*

Observe los dos puntos (:) al final de la condición y la intención de la pestaña significativa para el bloque de código bajo la condición, ver el Código 1.10.

```
> value = 99
> if value == 99:
>     print('That is fast')
> elif value > 200:
```

```
>     print('That is too fast')
> else:
>     print('That is safe')

That is fast
```

CÓDIGO 1.10: Ejemplo de If-Then-Else

Bucle *For*

`for` hace un blucle que se repite hasta cumplir la condición, ver el Código 1.11.

```
> # For-Loop
> for i in range(10):
>     print(i)

0
1
2
3
4
5
6
7
8
9
```

CÓDIGO 1.11: Ejemplo del bucle For

Bucle *While*

`while` hace un blucle que se repite hasta cumplir la condición, ver Código 1.12.

```
> # While-Loop
> i = 0
> while i < 10:
>     print(i)
```

```
>     i += 1

0
1
2
3
4
5
6
7
8
9
```

CÓDIGO 1.12: Ejemplo del bucle While

Ver Vídeo de este apartado

```
    1.7. Python - Control de flujo.
```

3. Estructuras de datos

Hay tres estructuras de datos en Python que encontrará más utilizadas y útiles. Son tuplas, listas y diccionarios.

Tupla

Las tuplas son colecciones de artículos de solo lectura, ver Código 1.13.

```
> a = (1, 2, 3)
> print(a)

(1, 2, 3)
```

CÓDIGO 1.13: Ejemplo de Tupla

Lista

Las listas usan la notación de corchetes y pueden indexarse usando la notación de matriz. Tenga en cuenta que estamos utilizando algunas funciones

simples similares a printf para combinar cadenas y variables al imprimir, ver Código 1.14.

```
> mylist = [1, 2, 3]
> print(f"Zeroth Value: {mylist[0]}")
> mylist.append(4)
> print(f"List Length: {len(mylist)}")
> for value in mylist:
>     print(value)

Zeroth Value: 1
List Length: 4
1
2
3
4
```

CÓDIGO 1.14: Ejemplo de Lista

Diccionario

Los diccionarios son asignaciones de nombres a valores, como pares *clave-valor*. Tenga en cuenta el uso del corchete y las anotaciones de dos puntos al definir el diccionario, ver Código 1.15.

```
> mydict = {'a':1, 'b':2, 'c':3}
> print(f"A value: {mydict['a']}")
> mydict['a'] = 11
> print(f"A value: {mydict['a']}")
> print(f"Keys: {mydict.keys()}")
> print(f"Values: {mydict.values()}")
> for key in mydict.keys():
>     print(mydict[key])

A value: 1
A value: 11
Keys: dict_keys(['a', 'b', 'c'])
Values: dict_values([11, 2, 3])
11
```

```
2
3
```

CÓDIGO 1.15: Ejemplo de Diccionario

Funciones

El mayor problema con Python es el espacio en blanco. Asegúrese de tener una nueva línea vacía después del código sangrado. El siguiente ejemplo define una nueva función para calcular la suma de dos valores y llama a la función con dos argumentos, ver Código 1.16.

```
> # Sum function
> def mysum(x, y):
>     return x + y
> # Test sum function
> result = mysum(1, 3)
> print(result)

4
```

CÓDIGO 1.16: Ejemplo de Función

Existe muchos recursos en internet para mejorar la programación en Python. Aquí hay algunos de ellos:

- Google Python Class[8]
- Python HOWTOs[9], invaluable for learning idioms and such (Python 3.6).
- Python Standard Library[10] Reference.

Ver Vídeo de este apartado

```
    1.8. Python - Estructuras de datos.
```

[8] https://developers.google.com/edu/python/
[9] https://docs.python.org/3.6/howto/index.html
[10] https://docs.python.org/3.6/library/index.html

1.3.2. Curso de `NumPy`

`NumPy` proporciona las estructuras y operaciones de datos básicos para SciPy. Estas son matrices (`ndarrays`) que son eficientes para definir y manipular.

Crear Array

Observe cómo convertimos fácilmente una lista de Python en una matriz `NumPy`, ver Código 1.17.

```
> # define an array
> import numpy
> mylist = [1, 2, 3]
> myarray = numpy.array(mylist)
> print(myarray)
> print(myarray.shape)

[1 2 3]
(3,)
```

CÓDIGO 1.17: Ejemplo de crear un array

Acceso a datos

La notación de matriz y los rangos se pueden usar para acceder de manera eficiente a los datos en una matriz `NumPy`., ver Código 1.18.

```
> # access values
> import numpy
> mylist = [[1, 2, 3], [3, 4, 5]]
> myarray = numpy.array(mylist)
> print(myarray)
> print(myarray.shape)
> print(f"First row: {myarray[0]}")
> print(f"Last row: {myarray[-1]}")
> print(f"Specific row and col: {myarray[0, 2]}")
> print(f"Whole col: {myarray[:, 2]}")

```

```
[[1 2 3]
 [3 4 5]]
(2, 3)
First row: [1 2 3]
Last row: [3 4 5]
Specific row and col: 3
Whole col: [3 5]
```

CÓDIGO 1.18: Ejemplo de acceso a elementos de un array

Operadores aritméticos

Las matrices `NumPy` se pueden usar directamente en aritmética, ver Código 1.19.

```
> # arithmetic
> import numpy
> myarray1 = numpy.array([2, 2, 2])
> myarray2 = numpy.array([3, 3, 3])
> print(f"Addition: {(myarray1 + myarray2)}")
> print(f"Multiplication: {(myarray1 * myarray2)}")

Addition: [5 5 5]
Multiplication: [6 6 6]
```

CÓDIGO 1.19: Ejemplo de operadores aritméticos de un array

Hay mucho más en las matrices `NumPy`, pero estos ejemplos le dan una idea de las eficiencias que proporcionan al trabajar con muchos datos numéricos. Para más información de esta librería:

- Documentación de NumPy[11]
- Documentación de SciPy[12]

Ver Vídeo de este apartado

```
    Vídeo 1.9. Python - Curso de NumPy.
```

[11] https://docs.scipy.org/doc/numpy/user/
[12] http://scipy-lectures.org/

1.3.3. Curso rápido de `Matplotlib`

`Matplotlib` se puede usar para crear diagramas y gráficos. La biblioteca se usa generalmente de la siguiente manera:

- Llame a una función de gráfica con algunos datos (por ejemplo `.plot()`).
- Llama a muchas funciones para configurar las propiedades de los gráficos (por ejemplo, etiquetas y colores).
- Haga visible la trama (por ejemplo, `.show()`).

Gráficos lineales

El siguiente ejemplo crea un gráfico lineal simple a partir de datos unidimensionales, ver Código 1.20 y Figura 1.1. El comando '`%matplotlib inline`' es un `iMagic` que hace representar en línea de comandos en Jupyter Notebook el gráfico.

```
%matplotlib inline
> # basic line plot
> import matplotlib.pyplot as plt
> import numpy
> myarray = numpy.array([1, 2, 3])
> plt.plot(myarray)
> plt.xlabel('some x axis')
> plt.ylabel('some y axis')
> plt.show()
```

CÓDIGO 1.20: Ejemplo de gráfico lineal

Gráficos de dispersión

A continuación se muestra un ejemplo simple de crear un diagrama de dispersión a partir de datos bidimensionales, ver Código 1.21 y Figura 1.2.

```
> %matplotlib inline
> # basic scatter plot
> import matplotlib.pyplot as plt
```

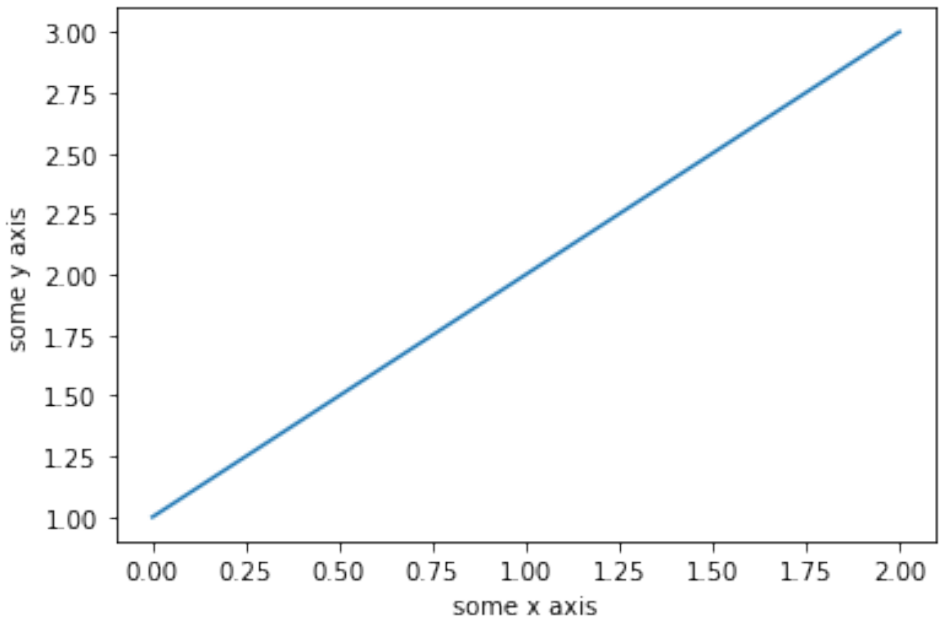

FIGURA 1.1: Gráfico lineal.

```
> import numpy
> x = numpy.array([1, 2, 3])
> y = numpy.array([2, 4, 6])
> plt.scatter(x,y)
> plt.xlabel('some x axis')
> plt.ylabel('some y axis')
> plt.show()
```

CÓDIGO 1.21: Ejemplo de gráfico de dispersión

Hay muchos más tipos de gráficos y muchas más propiedades que se pueden establecer en un gráfico para configurarlo. Seguro hasta el punto de que puede copiar y pegar una receta para trazar datos en un abrir y cerrar de ojos. Es una habilidad invaluable. Creo que será mejor que veas muchos ejemplos de `Matplotlib` para diferentes tipos de trama y prepares recetas para usar después. Hacer lindas tramas es un tema completamente diferente y para eso recomendaría estudiar la API cuidadosamente. Veo las gráficas como herramientas desechables para aprender más sobre un problema.:

- Galería `Matplotlib` o tipos de gráficos con código simple[13]

[13] https://matplotlib.org/gallery.html

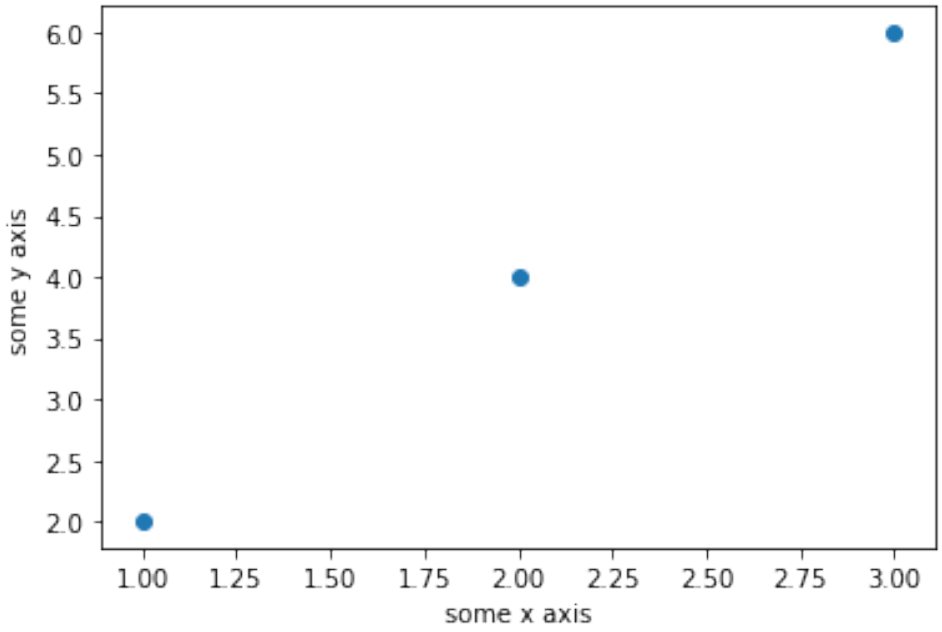

FIGURA 1.2: Gráfico de dispersión.

- Guía de inicio de `Matplolib`[14].
- Referencia `Matplolib` API[15].

Ver Vídeo de este apartado

```
1    1.10. Python - Curso de Matplotlib.
```

1.3.4. Curso rápido de `Pandas`

Pandas proporciona estructuras de datos y funcionalidad para manipular y analizar datos rápidamente. La clave para comprender Pandas para el aprendizaje automático es comprender las estructuras de datos `Series` y `Dataframe`.

[14] https://matplotlib.org/users/beginner.html
[15] https://matplotlib.org/api/index.html

Series

Una serie es una array unidimensional donde se pueden etiquetar las filas y columnas, ver Código 1.22. Puede acceder a los datos en una serie como una array `NumPy` y como un diccionario, ver Código 1.23.

```
> # series
> import numpy as np
> import pandas as pd
> myarray = np.array([1, 2, 3])
> rownames = ['a', 'b', 'c']
> myseries = pd.Series(myarray, index=rownames)
> print(myseries)

a    1
b    2
c    3
dtype: int64
```

CÓDIGO 1.22: Ejemplo de creación una Serie

```
> # series
> print(myseries[0])
> print(myseries['a'])

1
1
```

CÓDIGO 1.23: Ejemplo de acceso a una Serie

Dataframe

Un `Dataframe` es un array multidimensional donde se pueden etiquetar las filas y las columnas, ver Código 1.24. Los datos pueden indexarse usando nombres de columna, ver Código 1.25.

```
> # dataframe
> import numpy as np
```

```
> import pandas as pd
> myarray = np.array([[1, 2, 3], [4, 5, 6]])
> rownames = ['a', 'b']
> colnames = ['one', 'two', 'three']
> mydataframe = pd.DataFrame(myarray, index=rownames,
                             columns=colnames)
> print(mydataframe)

   one  two  three
a    1    2      3
b    4    5      6
```

CÓDIGO 1.24: Ejemplo de creación de un Dataframe

```
> # dataframe
> print("method 1:")
> print(f"one column: {mydataframe['one']}")
> print("method 2:")
> print(f"one column: {mydataframe.one}")

method 1:
one column: a    1
b    4
Name: one, dtype: int64
method 2:
one column: a    1
b    4
Name: one, dtype: int64
```

CÓDIGO 1.25: Ejemplo de acceso a una Serie

Hay mucha documentación para `Pandas` porque es una herramienta muy flexible. En general, los ejemplos de libros de cocina serán más útiles para usted aquí, ya que le darán ideas sobre diferentes formas de cortar y cortar sus datos:

- Guía de usuario de documentación de `Pandas`[16].

[16] https://pandas.pydata.org/pandas-docs/stable/

- Guía rápida[17] de `Pandas` que proporciona ejemplos cortos y documentados.
- Referencia `Pandas` API[18].

Ver Vídeo de este apartado

```
1    1.11. Python - Curso de Pandas.
```

1.3.5. Scikit-learn

En este caso vamos a ver esta librería poco a poco a o largo del curso aunque hay una excelente documentación en el sitio web `scikit-learn`. Le recomiendo pasar tiempo leyendo la API para cada clase y función que utilice para aprovechar al máximo:

- Guía de usuario de documentación de `scikit-learn`[19].
- Guía rápida[20] de `scikit-learn` que proporciona ejemplos cortos y documentados.
- Referencia de `scikit-learn` API[21].

1.4. Conclusiones

En este primer capítulo hemos podido observar algunos puntos muy introductorios sobre el curso. Específicamente, hemos estudiado los diferentes temas que tiene el curso haciendo bastante hincapié en la importancia del modelado predictivo dentro de la estadística, esto es, machine learning.

Por otro lado, hemos visto descrito qué es el ecosistema Anaconda para Ciencia de Datos y el por qué de la importancia de utilizar el lenguaje de

[17] https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html
[18] https://pandas.pydata.org/pandas-docs/stable/reference/index.html
[19] https://scikit-learn.org/stable/user_guide.html
[20] https://scikit-learn.org/stable/auto_examples/index.htmll
[21] https://scikit-learn.org/stable/modules/classes.html

programación Python para el modelado predictivo. Respecto a la plataforma Anaconda, como hemos visto, que es un software libre y gratuito que puede ser descargado e instalado de manera muy sencilla. Además, hemos descrito la importancia que tienen las diferentes librerías que tienen Python y su impacto que tienen en los resultados que tengamos.

Ahora bien, procedemos a conocer en profundidad este software y así ver el gran espectro de posibilidades que nos proporciona tanto para analizar y tratar un conjunto de datos, como para mejorar los resultados de machine learning.

Así mismo, a lo largo del curso veremos muchas librería y ejemplos que seguramente no se encuentre en estos contenidos vistos aquí, como, por ejemplo, `Seaborn`. En Python hay muchas librerías y muchos métodos en estas librerías que son imposibles ver de antemano sino que se tienen que ir aprendiendo conforme va avanzando el curso. En este sentido, en esta sección lo que se quiere dejar escrito de antemano las herramientas básicas que deberá conocer para que vaya familiarizándose con el ecosistema Python para Ciencia de Datos.