



GUÍA ACADÉMICA

“SÉ UN CIENTÍFICO DE DATOS”



TABLA DE CONTENIDOS

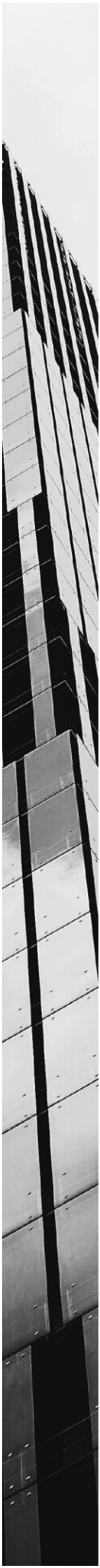
Introducción a Python	6
Fundamentos de Estadística	20
Análisis Exploratorio de Datos	23
Introducción al Machine Learning	27
Visualización Web de Modelos de Machine Learning	38
Aplicaciones de la Ciencia de Datos en tu vida profesional	41

TABLA DE ILUSTRACIONES

Ilustración 1: Analogía de un intérprete	6
Ilustración 2: Operaciones Booleanas	11
Ilustración 3: Operaciones Booleanas de comparación	12
Ilustración 4: Operaciones Booleanas de comparación	12
Ilustración 5: Función	16
Ilustración 6: Diagrama de dependencias de Scikit - Learn	28
Ilustración 7: Flujo de trabajo de Scikit - Learn	30
Ilustración 8: Representación gráfica de tipo de modelos	35
Ilustración 9: Matriz de confusión	36

TABLA DE CÓDIGOS

Código 1: Hola mundo	10
Código 2: Función type().....	11
Código 3: Variables	11
Código 4: Librería Math	13
Código 5: Sentencias condicionales	14
Código 6: Sentencias condicionales	14
Código 7: Sentencia FOR.....	15
Código 8: Sentencia While	16
Código 9: Estructura de una función	17
Código 10: Funciones.....	17
Código 11: Instalación de Scikit - Learn desde consola.....	28
Código 12: Importar Scikit - Learn desde Anaconda	29
Código 13: Importar dataset e imprimir la forma de los datos	31
Código 14: Estadística descriptiva de un dataset	32
Código 15: Normalizar los datos	33
Código 16: Escalar los datos	33
Código 17: Maneras de tratar valores faltantes	34
Código 18: Separar datos en train y test	35



Módulo 1

Introducción a Python

Objetivos

- Desarrollar en el estudiante la lógica de programación.
- Entender la relación entre el desarrollo de software y el análisis y ciencia de datos.
- Entender los fundamentos, sintaxis y buenas prácticas del lenguaje de programación Python.
- Desarrollar una aplicación de Escritorio para descargar videos de YouTube.

1. Introducción a Python

1.1. Bienvenida

El lenguaje de programación que aprenderemos es Python. Python es un gran ejemplo de un lenguaje de alto nivel (High-level language) otros lenguajes de alto nivel son: C, C++, Java.

La lógica dicta que, si existen lenguajes de alto nivel, deberían existir lenguajes de bajo nivel, la respuesta es sí y suelen ser llamados lenguaje de máquina.

Tenemos dos tipos de programas que procesan lenguajes de alto nivel a lenguajes de bajo nivel, debemos tener en cuenta que al final del día nuestros computadores solo entienden 1 o 0.

- Los Compiladores procesan lenguajes de bajo nivel.
- Los Intérpretes procesan lenguajes de alto nivel.

Nos vamos a enfocar en aquellos que procesan lenguajes de alto nivel como lo es Python.

1.1.1. Intérprete

Un intérprete lee un programa de alto nivel y lo ejecuta, es decir el computador hace lo que el programa dice, esto se da en tiempo de ejecución línea por línea. Esto toma un poco de tiempo a diferencia de lo que pasa con los compiladores.



Ilustración 1: Analogía de un intérprete

1.2. Python como lenguaje de programación

Python es considerado un lenguaje interpretado pues sus programas son ejecutados por un intérprete.

Nota: Entendamos que un programa en Python es un fichero con extensión **.py** por ejemplo: my-first-program.py

Desde ahora un fichero que contenga instrucción en un lenguaje de programación será conocido como Script. Que Python sea un lenguaje de programación no hace que deje de ser un programa más al igual que Word, Excel, Google Earth, por ende, debe ser instalado en nuestros computadores: <https://www.python.org/downloads/windows/>.

1.2.1. ¿Qué es un algoritmo?

En su concepción más pura un algoritmo es una secuencia de pasos bien definidos que tienen como objetivo final ejecutar una tarea. Esto no necesariamente se relaciona con computadores, sin embargo, en nuestro caso obviamente se relacionará con computadores.

1.2.2. ¿Qué es un programa?

Un programa contiene un algoritmo y a su vez dicho algoritmos le dará ordenes al computador para ejecutar distintas tareas, tales como:

- Resolver ecuaciones diferenciales.
- Obtener las raíces de un polinomio.
- Buscar y reemplazar textos.

Word contiene un algoritmo que nos ayuda a poner las letras en **negrita**, así como ese Word es un programa que contiene miles de algoritmos distintos.

Independiente del lenguaje de programación un programa podría contener lo siguiente:

- Input: Dato de entrada que es ingresado por teclado o un dispositivo de entrada.
- Output: Se muestra en una terminal, interfaz gráfica o navegador Web.
- Math: Ejecución de operaciones matemáticas básicas como suma, resta o multiplicación.
- Conditional execution: Revisa por ciertas condiciones y ejecuta las adecuadas basándose en "Si esto es verdadero o falso ejecutaré esto o lo otro".
- Repetition: Ejecuta alguna acción en forma de bucle o siguiendo alguna condición.

1.2.3. ¿Qué hace un programador?

Un programador es una persona que tiene la capacidad de crear, mantener y actualizar herramientas informáticas. Es una palabra muy general, diferentes profesionales son programadores además de otras cualidades, por ejemplo:

- Un científico de datos es un programador, pero también es una persona que conoce bien cómo funciona la estadística.
- Un ingeniero de petróleos con enfoque en el Machine Learning es un programador además de una persona con un amplio conocimiento sobre la industria Oil and Gas.

Si empezamos a crear herramientas basadas en código somos programadores.

1.2.4. ¿Por qué es importante aprender a programar?

Sin duda alguna una de las habilidades de un programador es la alta capacidad de resolver problemas. Casi todas las profesiones se complementan de manera casi perfecta con la programación, de esa forma un profesional que tenga esa habilidad puede diversificar su vida profesional de forma increíble.

Los salarios en tecnología cada vez crecen más y más, así como la demanda de estos.

1.2.5. Buenas prácticas de programación.

Como en toda profesión las buenas prácticas son fundamentales para crear herramientas robustas, escalables y eficientes. Cada lenguaje de programación cuenta con documento formal donde se especifica de forma precisa cuales son estas buenas prácticas.

Pero *¿Qué son estas buenas prácticas?* Antes de eso, te quiero mostrar el ZEN de Python o la filosofía del lenguaje de programación:

The Zen of Python, by Tim Peters

“Beautiful is better than ugly.

Explicit is better than implicit. Simple is better than complex.

Complex is better than complicated. Flat is better than nested.

Sparse is better than dense. Readability counts.

Special cases aren't special enough to break the rules. Although practicality beats purity.

Errors should never pass silently. Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

*Although never is often better than *right* now.*

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!”

En esto, notamos claramente como debemos entender a Python y su objetivo: Claro y Simple.

A nivel técnico, la forma correcta de escribir en Python se encuentra en un documento llamado PEP-8 que puede ser revisado es: <https://peps.python.org/pep-0008/>



1.3. Herramientas necesarias y entorno de trabajo.

1.3.1. ¿Dónde escribimos código?

Podemos escribir código en un Bloc de notas e incluso en Word, sin embargo, esto no es una buena idea, es por eso que existen dos formas generales en donde escribir código: Entorno de Desarrollo Integrado y Editores de texto.

1.3.2. Entorno de Desarrollo Integrado

Integrated Development Environment (IDE) o en español Entorno de Desarrollo Integrado. Un IDE es un software bastante robusto que cuenta con todas las herramientas necesarias para desarrollar en un lenguaje de programación, una vez instalado un IDE no tenemos que hacer nada más aparte de empezar a programar.

1.3.3. Editores de texto

Un editor de texto es un software menos robusto, cuyo objetivo principal es recibir texto y estructurarlo en un formato que sea más fácil para el programador leerlo. En un editor de texto simplemente escribimos texto plano y para ejecutarlo debemos hacerlo desde la terminal.

Hoy en día hay una línea muy fina entre editores de texto e IDEs.

Ejemplos de IDEs para Python:

- Pycharm.
- Spyder.

Por otro lado, un editor de texto es de uso general, sin importar el lenguaje de programación.

Ejemplos de Editores de texto:

- Visual Studio Code.
- Atom.
- Notepad ++.

1.3.4. Visual Studio Code

Visual Studio Code es el editor de texto que utilizaremos en este curso, es desarrollado por Microsoft y es el editor de texto más usado, pero no el mejor, pues las herramientas que use un desarrollador dependen de sus necesidades.

Lo podemos instalar en: <https://code.visualstudio.com/download>

1.3.5. Anaconda, Jupyter Notebooks, Ciencia de datos.

Las herramientas que use el programador dependen de sus necesidades. En la ciencia de datos el estándar es utilizar los famosos notebooks que son bloques de código que se ejecutan de forma independiente.

1.3.6. Control de versiones

Es un estándar en la industria de la tecnología manejar varias versiones para un producto. En este curso mencionaremos brevemente qué es Git y GitHub.

1.4. Fundamentos de Python.

Los contenidos de esta sección se pueden seguir con este complemento: <https://colab.research.google.com/drive/19DJpZnjNMyYVoMFxSZEutWTKngmU0vlO?usp=sharing>



Código 1: Hola mundo

Esto es lo primero que harás cuando programes por primera vez en Python, imprimir en pantalla un 'Hola mundo'.

Dentro de Python *print()* es una función nativa para imprimir texto en la terminal. Hablaremos de las funciones en el futuro.

1.4.1. Tipos de datos

Cada lenguaje de programación soporta diferentes tipos de datos, se los conoce como **Built-in Types**. Es importante saber que en Python todo son **Objetos**, este término viene de la Programación Orientada a Objetos, un tema que tocaremos en el futuro.

Por ahora, los diferentes tipos son utilizados dependiendo el tipo de programa que estemos construyendo.

- Boolean: Los tipos de datos Booleanos son valores de verdad: True o False.
- Numeric types - int, float, complex: Existen 3 tipos de datos numéricos en Python.
 - Enteros (int): Hacen referencia a los valores enteros, sin punto decimal. La edad es un tipo de dato entero, mientras que la altura no.
 - Decimales (float): Hacen referencia a los valores decimales. La altura de una persona, su peso, son valores del tipo float o decimales.
 - Complejos: Hacen referencia a los números complejos.
- Str (string): Cadena de caracteres o texto.

En Python podemos observar el tipo de dato con la función *type()*.

A terminal window with a dark background and green, yellow, and red window control buttons at the top left. It contains the following Python code:

```
type(" Feyneur!")
type(123)
type(123.45)
type(True)
```

Código 2: Función type()

1.4.2. Variables y Asignaciones

Una variable es un espacio en memoria en donde podemos guardar un dato. Podemos pensar que una variable es como una caja en donde guardaremos algo.

A terminal window with a dark background and green, yellow, and red window control buttons at the top left. It contains the following Python code:

```
my_variable = 55
print(my_variable)
```

Código 3: Variables

Importante: Python es un lenguaje no tipado, es decir no debemos especificar su tipo, sin embargo, su tipo existe en la asignación.

1.4.3. Operadores y Operandos

En Python tenemos distintas operaciones, en esta parte mencionaremos algunas.

Operaciones Booleanas.

Operation	Result	Notes
<code>x or y</code>	if <code>x</code> is false, then <code>y</code> , else <code>x</code>	(1)
<code>x and y</code>	if <code>x</code> is false, then <code>x</code> , else <code>y</code>	(2)
<code>not x</code>	if <code>x</code> is false, then <code>True</code> , else <code>False</code>	(3)

Ilustración 2: Operaciones Booleanas

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

Ilustración 3: Operaciones Booleanas de comparación

Operaciones numéricas:

Operation	Result
<code>x + y</code>	sum of <code>x</code> and <code>y</code>
<code>x - y</code>	difference of <code>x</code> and <code>y</code>
<code>x * y</code>	product of <code>x</code> and <code>y</code>
<code>x / y</code>	quotient of <code>x</code> and <code>y</code>
<code>x // y</code>	floored quotient of <code>x</code> and <code>y</code>
<code>x % y</code>	remainder of <code>x / y</code>
<code>-x</code>	<code>x</code> negated
<code>+x</code>	<code>x</code> unchanged
<code>abs(x)</code>	absolute value or magnitude of <code>x</code>
<code>int(x)</code>	<code>x</code> converted to integer
<code>float(x)</code>	<code>x</code> converted to floating point
<code>complex(re, im)</code>	a complex number with real part <code>re</code> , imaginary part <code>im</code> . <code>im</code> defaults to zero.
<code>c.conjugate()</code>	conjugate of the complex number <code>c</code>
<code>divmod(x, y)</code>	the pair <code>(x // y, x % y)</code>
<code>pow(x, y)</code>	<code>x</code> to the power <code>y</code>
<code>x ** y</code>	<code>x</code> to the power <code>y</code>


Ilustración 4: Operaciones Booleanas de comparación

1.4.4. Orden de operaciones

Podemos realizar operaciones entre variables del mismo tipo, en ese sentido debemos tener muy en cuenta el orden de las operaciones: Paréntesis, exponentes, multiplicación, división, suma y resta.

En este punto nos adelantaremos un poco y hablaremos de las **Librerías**, las librerías son porciones de código creadas por *Python Software Foundation* o por desarrolladores independientes.

Podemos pensar que es un Kit de herramientas para una tarea, en este caso hay una librería que nos entrega más operaciones matemáticas como lo es **Math**, para obtener este Kit de herramientas usamos:



```
import math #importamos funcionalidades
print('El factorial de 5 es: ', math.factorial(5))
```

Código 4: Librería Math

1.4.5. Sentencias condicionales

Las sentencias condicionales son base en la programación, responden a la lógica” Si esto es verdadero o falso haz esto o esto”. Pensemos en lo siguiente:[1]

Si está lloviendo, no saldré a jugar, caso contrario saldré a jugar.

Vamos a intentar traducir esto a pseudo código.

Si (¿está lloviendo?) entonces:

Salgo a jugar

Si no:

No Salgo a jugar

Nótese que todo depende de la respuesta a *¿Está lloviendo?* “si esta es **Sí** o **Verdadero** salimos a jugar, pero si esta es **No** o **Falso** no salimos a jugar.

En Python esto se ve de la siguiente manera:

```

number_one = 5
number_two = 8

if number_one > number_two:
    print("Esto jamas será verdad!")
else:
    print(f"En efecto {number_one} es menor que {number_two}")

```

Código 5: Sentencias condicionales

1.4.6. Tipos de sentencias condicionales

Realmente las sentencias condicionales son sencillas una vez entiendes el concepto base.

Existen otro tipo de sentencias condicionales para otras exigencias.

Elif

```

# Esta sentencia condicional se usa cuando tengo varias opciones
para una misma pregunta.

user_input = input("Ingrese un numero del 1 al 3. ")

if user_input is 1:
    print("Has ingresado 1.")
elif user_input is 2:
    print("Has ingresado 2.")
elif user_input is 3:
    print("Has ingresado 3.")

# Esto lo podemos hacer con sentencias simples

if user_input == 1:
    print("Has ingresado 1")
else:
    if user_input == 2:
        print("Has ingresado 2")
    else:
        print("Has ingresado 3")

# El uso de uno u otro depende el caso.

```


Código 6: Sentencias condicionales

1.4.7. Sentencias repetitivas

Una sentencia repetitiva, no es más que un bucle generado por computador. Pensemos en un ventilador, un ventilador sigue girando muchas veces hasta que alguien lo apague o en su defecto se vaya la luz.

Tenemos dos sentencias repetitivas **For** y **While**.

FOR

A screenshot of a code editor with a dark background and green border. The code is written in Python and calculates the sum of numbers from 1 to 10. It includes a comment, variable initialization, a for loop with print and sum operations, and a final print statement.

```
# Suma de los numeros del 1 al 10

sum = 0

for i in range(1,11):
    print(i)
    sum = sum + i

print("La suma es: ", sum)
```

Código 7: Sentencia FOR

Entendamos la función `range()`, es una función que entrega una lista de números de la siguiente manera:

- `range(n)`: toma los números desde el 0 hasta el número `n-1`.
- `range(n+1)`: toma los números desde el 0 hasta el `n`.
- `range(1, n+1)`: toma los números desde el 1 hasta el `n`.

WHILE

While es una sentencia que necesita un valor de verdad para detenerse, a diferencia de FOR que se detiene una vez ya no tiene en donde iterar más. While solo para cuando se le pase un valor negativo.

```

# Ejemplo de uso de While
# vamos a imprimir en pantalla los numeros del 1 al 5

sum = 1

while(sum <= 5):
    print(sum)

    sum = sum + 1

```

Código 8: Sentencia While

El uso de While o For depende el algoritmo que estemos haciendo.

1.4.8. Estructura de datos y algoritmos

Las estructuras de datos y algoritmos es un tema muy extenso, existen materias enteras de dos tomos en la universidad para abarcar este tema. En este curso solo mencionaremos brevemente su utilidad.

Las estructuras de datos son diferentes formas de guardar datos, en este punto ya nos estamos acercando un poco más a la ciencia de datos. Diferentes estructuras tienen asociados diferentes algoritmos con diferente eficiencia.

1.4.9. Funciones

Este es uno de los temas más importantes en este curso y en la programación en general.

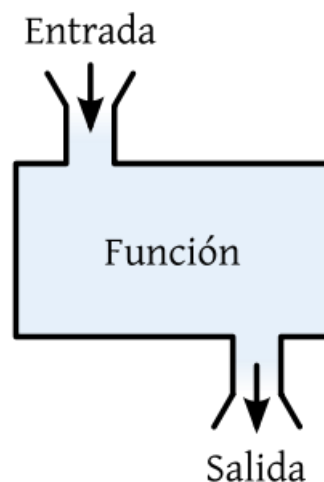



Ilustración 5: Función

Una función es 'algo' que recibe un input, lo transforma y entrega algo diferente. Esto no es diferente de la definición de una función matemática.

Una función tiene diferentes partes:

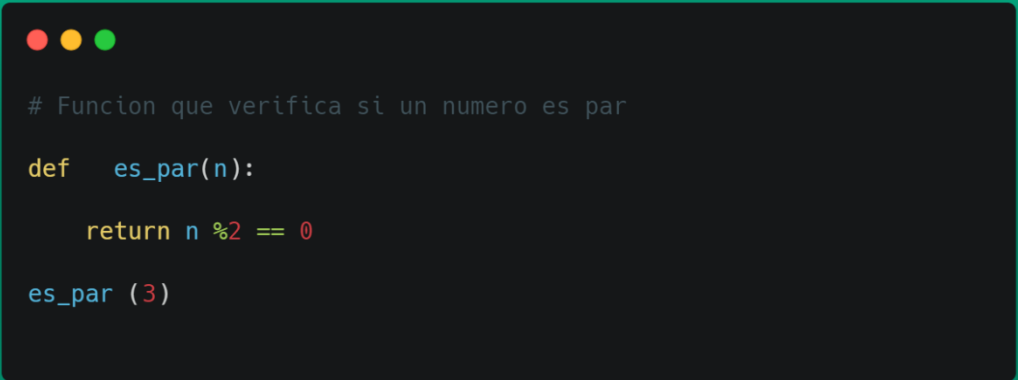


```
def name_of_function(arguments):  
    # body of function  
  
    return item_to_return
```

Código 9: Estructura de una función

Los argumentos son los elementos por transformar y podemos retornar otros elementos que han sido basados en dichos argumentos.

En nuestro notebook complementario tendremos más ejemplos de funciones.



```
# Funcion que verifica si un numero es par  
  
def es_par(n):  
    return n %2 == 0  
  
es_par (3)
```

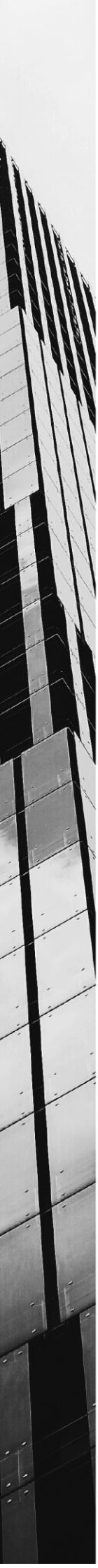
Código 10: Funciones

1.5. Programación Orientada a Objetos

La Programación Orientada a Objetos al igual que las estructuras de datos y algoritmos es un tema sumamente extenso.

Esta es una forma diferente de programar, mucho más estructurada que sigue un conjunto de reglas muy rigurosas.

Debemos entender que aprender programación orientada a objetos no es una elección es casi obligatorio para entender cómo funcionan las librerías, y cómo funcionan distintos algoritmos.



Módulo 2

Fundamentos de Estadística

Objetivos

- Comprender los fundamentos de la estadística descriptiva presentes en los modelos de machine learning.

1. Fundamentos de Estadística

1.1. Introducción

La estadística es una rama de las matemáticas cuyo objetivo se encuentra en coleccionar, organizar, analizar, interpretar y presentar datos. La estadística aborda problemas concretos en los cuales estudia una población y trata de aplicar un modelo en la descripción de alguna característica o fenómeno que pueda observarse en ella. Además, la manera en que usualmente se recopilan los datos es a través de encuestas o experimentos.

Es importante resaltar uno de los conceptos claves en la recolección de datos: el proceso de medición. La medición involucra transformar información de un evento en variables, las cuales grosso modo pueden ser cualitativas/categóricas o cuantitativas/numéricas. Las primeras hacen énfasis en agrupar datos que poseen alguna característica en común; en cambio las variables cuantitativas dan un valor numérico partiendo de una escala local o global.

Los métodos más importantes en el análisis de datos son:

- Estadística Descriptiva: Se encarga de resumir datos de una muestra usando índices estadísticos como la media, desviación estándar o rangos. Además, usualmente está acompañada de esquemas visuales que ayudan a interpretar de manera sencilla la información presentada.
- Estadística Inferencial: Trata de realizar predicciones provenientes de un conjunto de datos. Asume que la muestra proviene de una población más grande y que por lo tanto puede utilizar modelos para realizar inferencias acerca de datos que no necesariamente han sido obtenidos de manera bruta.

1.2. ¿Por qué es importante aplicar la estadística en modelos de aprendizaje automático?

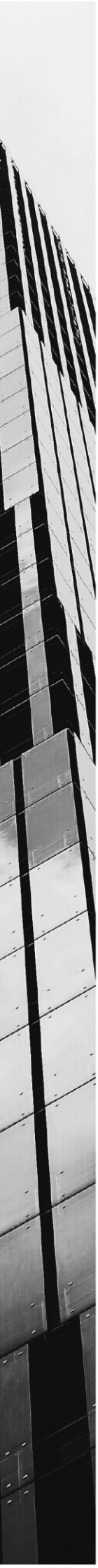
Uno de los paradigmas más importantes del aprendizaje automático es la teoría del aprendizaje estadístico. Este marco de referencia nos permite utilizar la inferencia estadística para crear funciones predictivas basadas en los datos de un problema. Tiene un amplio uso en la industria, por ejemplo, en visión computacional, reconocimiento de caracteres y bioinformática.

En síntesis, se trata de modelar los datos partiendo de minimizar el error proveniente de las funciones matemáticas que los caracterizan, para ello utiliza conceptos probabilísticos asumiendo que existe manera de clasificar los datos y que además esta función podrá ser utilizada en la extrapolación de información a partir de lo ya concebido.

1.3. Partir de la matemática al algoritmo

Las medidas de tendencia central o los modelos inferenciales provenientes de la estadística son procesos que pueden ser obtenidos computacionalmente una vez los datos se hayan recolectado mediante minería. No obstante, el verdadero poder de predicción se encuentra cuando se crean algoritmos que pueden ser sometidos a refinamientos para que su poder de obtener información se maximice.

La estadística tiene procesos formales que parten de definiciones y teoremas de teoría de probabilidad para sentar su uso dentro de la matemática aplicada. Esto permite brindar una base matemática sólida para que los algoritmos de aprendizaje automático den resultados plausibles, replicables y, sobre todo, útiles.



Módulo 3

Análisis Exploratorio de Datos

Objetivos

- Comprender los fundamentos y herramientas para desarrollar un análisis exploratorio de datos, paso previo a construir modelos de machine learning.

1. Análisis Exploratorio de Datos

1.1. Introducción

El análisis exploratorio de datos hace referencia al proceso de descubrir patrones y relaciones con respecto a un conjunto de datos. Su objetivo principal se enfoca en dar sentido a los datos disponibles y familiarizarse con ellos, previo a un análisis o modelado. Incluso, es considerado un paso primordial previo a la construcción de modelos y puede ayudar a formular posibles preguntas en el área que se esté aplicando.

Dependiendo del tipo de análisis o el modelo que se desee construir hay diferentes objetivos que se pueden desear alcanzar. En general algunos de ellos son:

- Descubrir la forma de los datos y determinar cómo está estructurada.
- Inspeccionar y familiarizarse con los datos al resumiéndolos y visualizándolos.
- Detectar valores atípicos, datos faltantes, anomalías en la información y tomar decisiones sobre cómo actuar ante ellos.
- Encontrar nuevas vías para el análisis y futuras ideas de investigación en el área.
- Preparar la información para un modelo o análisis, incluyendo: revisión de supuestos, selección de características y un método apropiado.

Así como los objetivos del análisis exploratorio de datos pueden variar, las técnicas utilizadas para cumplir los objetivos también lo hacen. De forma general, los procesos generalmente utilizan estrategias que caen en alguna de las siguientes categorías.

1.2. Inspección de datos

La inspección de datos es un primer paso clave para cualquier tipo de análisis. Esto puede ayudar a clarificar potenciales problemas o vías para abarcar de mejor manera los problemas. Por ejemplo, el analizar las primeras filas de un conjunto de datos nos brinda información del tipo de datos con los que vamos a trabajar: cualitativos, cuantitativos, faltantes, etc. Dentro de la inspección de datos, detectar aquellos faltantes, determinar cuánta información es y qué hacer con ella es lo que se abarca en esta categoría.

1.3. Resumen numérico

Una vez nuestros datos han sido inspeccionados y realizado algunos pasos de limpieza, resumir la información es una buena forma de condensar la información que tenemos en dimensiones más razonables. Para los datos numéricos, nos permite escalar la información, conocer su dispersión y determinar su tendencia. Para datos categóricos, nos da información sobre el número de categorías con el que vamos a trabajar y su

frecuencia. En código existen funciones, por ejemplo: `describe()` de pandas que tras adaptar nuestro set de datos nos permite visualizar un resumen de nuestra información.

1.4. Visualización de datos

Si bien los resúmenes numéricos son útiles para condensar información, los resúmenes visuales brindan aún más contexto y detallan esa misma información en una pequeña cantidad de espacio. Existen muchos tipos diferentes de visualizaciones que se pueden crear a partir del análisis exploratorio de datos. Por ejemplo, los histogramas nos permiten inspeccionar la distribución de una característica cuantitativa, proporcionándonos información sobre la dispersión, la tendencia central o el sesgo. Otro tipo de visualizaciones como los diagramas de dispersión nos permiten analizar las relaciones entre varios datos; nos podríamos preguntar si existe una relación entre las horas de sueño de una persona y sus horas de estudio.

En síntesis, el análisis exploratorio de datos es un paso crucial previo a cualquier proyecto que involucre datos. Al hacerlo se informa la limpieza de datos, permite construir nuevas preguntas sobre el problema que se está trabajando, permite elegir técnicas de modelado y nos resulta útil para ajustar el modelo seleccionado.

Python de su lado ha sido constantemente ranqueado dentro del top 10 de lenguaje de programación y utilizado ampliamente por expertos en ciencia de datos (Mukhiya & Ahmed, 2020). En nuestro módulo trabajaremos con algunas librerías como pandas, seaborn, numpy y matplotlib, las cuales nos brindan las herramientas para ejecutar todos los conceptos que hemos descrito. Responderemos preguntas del estilo:

1.4.1. ¿Qué es numpy y por qué necesitamos comprender de él?

Numpy es una librería de código abierto de Python enfocado para la computación numérica y científica. Numpy facilita la eficiente implementación y el uso de:

- Creación, copiado y división de arreglos y matrices
- Operaciones con arreglos y matrices multidimensionales
- Indexación
- Funciones algébricas lineales, búsqueda, conteo, etc.

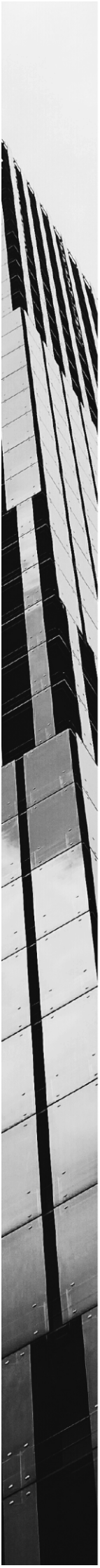
En varias dimensiones implementar estas funciones resulta en tiempo computacional costoso, lo cual Numpy maneja bajo algoritmos vanguardistas.

1.4.2. ¿Qué es pandas y por qué necesitamos comprender de él?

Matplotlib es una librería multiplataforma enfocada en la creación de gráficos 2D a partir de listas o matrices. Necesitamos comprender esta librería pues nos proporciona la habilidad de visualizar nuestros datos mediante gráficos haciendo que la información sea más comprensible. Por ejemplo, diagramas de dispersión, gráficos de barras, diagramas de caja pueden ser desarrollados de manera eficiente con instrucciones claras y concisas.

1.4.3. ¿Qué es matplotlib y por qué necesitamos comprender de él?

Matplotlib es una librería multiplataforma enfocada en la creación de gráficos 2D a partir de listas o matrices. Necesitamos comprender esta librería pues nos proporciona la habilidad de visualizar nuestros datos mediante gráficos haciendo que la información sea más comprensible. Por ejemplo, diagramas de dispersión, gráficos de barras, diagramas de caja pueden ser desarrollados de manera eficiente con instrucciones claras y concisas.



Módulo 4

Introducción al Machine Learning

Objetivos

- Entender para qué funciona el aprendizaje automático y sus aplicaciones en la vida real.
- Comprender los procesos de preprocesamiento de los datos para la creación de modelos de aprendizaje automático exitosos.
- Entender los conceptos básicos de los tipos de aprendizajes y en qué tipo de problemas aplicar cada uno.
- Ser capaz de construir y ejecutar modelos con las librerías de Scikit-learn y Tensorflow en Python.

1. Introducción al Machine Learning

1.1. ¿Qué es el aprendizaje de máquina o automático?

El aprendizaje automático o Machine Learning es un área de la Inteligencia Artificial, que se enfoca en la capacidad de que las máquinas aprendan de manera autónoma a partir de ciertos datos proporcionados, se basa en el reconocimiento de patrones y la capacidad de generalizar comportamientos, son técnicas capaces de detectar automáticamente los patrones en los datos, para mediante procesos de inferencia poder generar resultados que se asemejen a los realizados por las personas, en tareas como clasificación, predicción, agrupamiento, entre otras.

El aprendizaje automático se divide en dos enfoques, supervisado y no supervisado. El aprendizaje supervisado utiliza ejemplos conocidos para obtener las inferencias mientras que el aprendizaje no supervisado no dispone de ejemplos con un objetivo o etiqueta conocido, y debe aprender a generar asociaciones., estos enfoques se los verán a profundidad más adelante.

El aprendizaje automático o aprendizaje de máquina tiene su sustento principalmente en modelos estadísticos, de probabilidad y geometría, la base fundamental de estos es la regresión lineal.

1.1.1. El machine learning como paradigma de resolución de problemas.

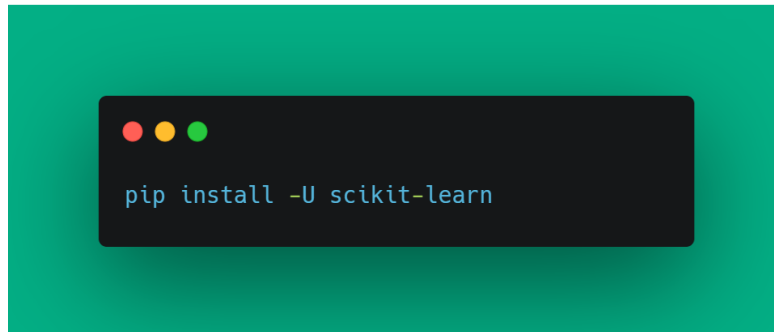
Las técnicas de Aprendizaje Automático están experimentando un auge sin precedentes en diversos ámbitos, tanto en el mundo académico como en el empresarial, principalmente debido a dos factores importantes:

- El gran avance en términos de hardware y software, lo que permite una capacidad computacional de cálculo muchísimo mayor.
- La gran disponibilidad de información que existe actualmente debido al auge del internet y los datos masivos

En base a lo anterior, es importante mencionar como cada vez más el aprendizaje automático ha sido ampliamente empleado en resolver problemas de las siguientes ramas: regresión, clasificación, agrupamiento y predicción.

1.1.2. Introducción a Scikit-Learn

Scikit-learn es una librería/framework de Python para Aprendizaje Automático, la misma es de código abierto y agrupa una grama muy completa de algoritmos de aprendizaje supervisado y no supervisado. En la siguiente URL, se puede acceder a la documentación de la librería: <https://scikit-learn.org/stable/>

A terminal window with a green background and a black command prompt. The command 'pip install -U scikit-learn' is entered in white text. Above the command prompt are three colored dots: red, yellow, and green.

```
pip install -U scikit-learn
```

Código 11: Instalación de Scikit - Learn desde consola

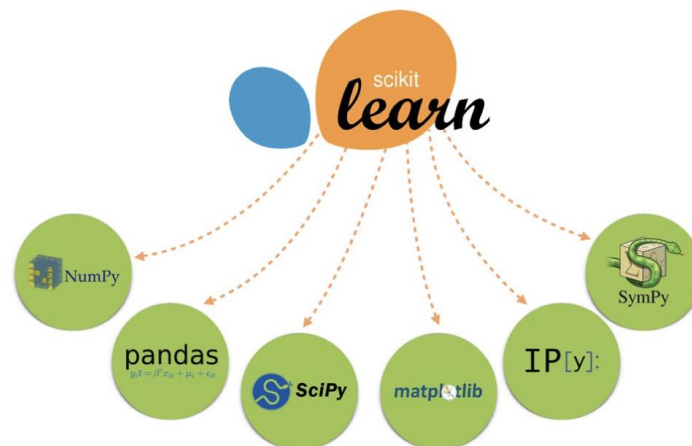


Ilustración 6: Diagrama de dependencias de Scikit - Learn

Scikit-learn está construida sobre la librería Scientific Python, la misma que incluye gran cantidad de dependencias relacionadas al procesamiento numérico, matricial y de visualización, como se puede ver en la Ilustración 6.

Para poder emplear el paquete de Scikit-learn es importante importar estos paquetes ahora mencionados, adicionalmente para poder usar scikit-learn es importante instalarlo en nuestra distribución de Anaconda o en el ambiente global de Python, el proceso de instalación necesario se lo puede ver en los Códigos 11 y 12.

```
conda list scikit-learn # Ver la
version de la libreria
conda list # Revisar todos los paquetes
disponibles
python -c "import sklearn;
sklearn.show_versions()"
```

Código 12: Importar Scikit - Learn desde Anaconda

1.1.3. Flujo de trabajo en Scikit-Learn

Dentro de Scikit-learn tenemos un entorno completo que es capaz de ejecutar diversas funcionalidades, entre estas se puede realizar lo siguiente:

- Algoritmos de aprendizaje supervisados
La gran mayoría de los algoritmos de Machine Learning que entran en la clasificación de aprendizaje supervisado forman parte de scikit-learn, desde modelos, como la regresión lineal, como las máquinas de vectores de soporte (SVM), árboles de decisión y hasta los métodos bayesianos, todos estos serán estudiados más adelante.
- Validación cruzada
Existen varios métodos para verificar la precisión de los modelos supervisados y esta librería cuenta con las instrucciones necesarios para poder implementar estos métodos sin tanto esfuerzo.
- Algoritmos de aprendizaje no supervisados
Igual que los algoritmos de aprendizaje supervisados, esta librería cuenta con una gran variedad de algoritmos disponibles para aprendizaje no supervisado, comenzando por el clustering, el análisis factorial, el análisis de componentes principales y las redes neuronales no supervisadas.
- Varios conjuntos de datos o dataset
Esta librería posee varios dataset que son muy útiles para que puedas practicar tus conocimientos de Machine Learning con Python, implementando esta librería. Algunos de estos dataset los utilizaremos a lo largo de este curso.
- Extracción y selección de características
Esta librería es muy útil para extraer características de imágenes y texto, así como también para identificar atributos significativos a partir de los cuales crear modelos supervisados.

Como vemos la librería nos brinda un marco de trabajo bastante estable y completo para poder tener un flujo de trabajo desde la consecución de los datos hasta la obtención de modelos con buen desempeño, es así como el proceso habitual de trabajo con esta librería sigue los siguientes pasos.

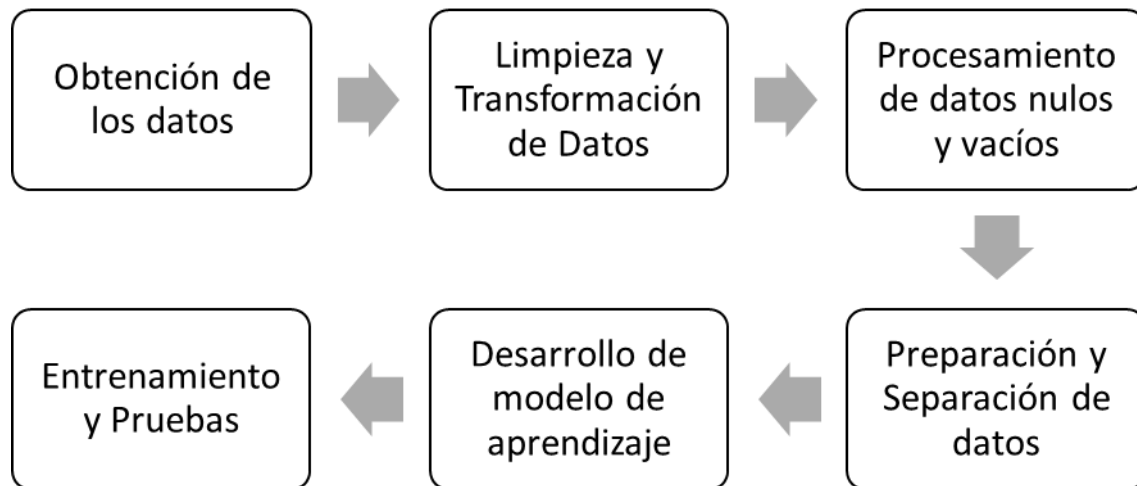


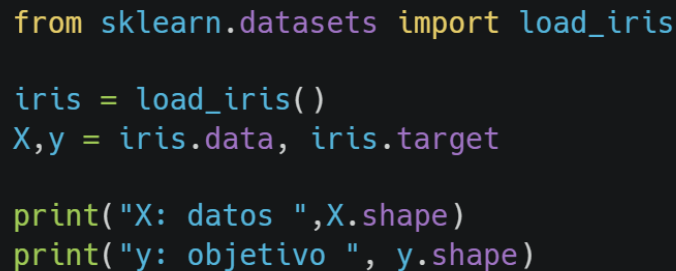
Ilustración 7: Flujo de trabajo de Scikit - Learn

Todos los pasos anteriormente descritos serán explicados a profundidad en las siguientes subsecciones.

1.1.3.1. Obtención de los datos

Para el proceso de obtención de los datos, se pueden emplear cualquier dataset, sin embargo, en este curso emplearemos los datasets provistos por Sklearn. Una fuente muy valiosa para obtener dataset es la web Kaggle, en donde además de dataset etiquetados y con información, podemos encontrar soluciones de código y referencias para la creación y evaluación de modelos de aprendizaje automático.

En Código 13 se presenta como importar un dataset, el mismo que es el 'load_iris', un dataset que funciona para clasificación multi clase.



```
from sklearn.datasets import load_iris

iris = load_iris()
X,y = iris.data, iris.target

print("X: datos ",X.shape)
print("y: objetivo ", y.shape)
```

Código 13: Importar dataset e imprimir la forma de los datos

El proceso de importación de datos al código Python es muy importante, ya que la mayoría de los algoritmos brindados por Scikit-learn trabajan con matrices o vectores, ambos en dataframes o arreglos de NumPy.

1.1.3.2. Limpieza y transformación de datos

Uno de los problemas más habituales que existen al tratar con datasets propios o externos es que muchas veces esto tienen datos que son irrelevantes o que afectan en demasía al desarrollo de modelos de aprendizaje, es por esto por lo que es importante limpiar los datos y transformarlos de ser necesario.

En cuanto al proceso de limpieza, este debe iniciar con un análisis de estadístico de los datos, para poder entender la distribución de las variables, las medidas de estadística tradicional y de que datos se puede prescindir. Considerando esto, se pueden aplicar dos técnicas para poder analizar los datos y limpiarlos, la matriz de correlación y la estadística descriptiva.

- Estadística Descriptiva

Los valores de la estadística descriptiva son muy útiles para entender que variables o datos pueden ser relegados, entre los cálculos más usuales que se pueden ejecutar sobre un dataset tenemos: media, moda, mediana, desviación estándar, percentiles, rangos de datos.

Para ejecutarlos en Python es muy sencillo, necesitamos que los datos están almacenados en un dataframe y luego como lo muestra el Código 14, invocar a la función *describe()*.

- Matriz de Correlación

Una matriz de correlación investiga la dependencia entre múltiples variables al mismo tiempo. Muestra datos al estilo tabla en donde cada fila y columna representan una variable, y el valor correspondiente es el coeficiente de

correlación que denota la fuerza de la relación entre ambas variables. Hay muchos tipos de coeficientes de correlación (coeficiente de Pearson, coeficiente de Kendall, coeficiente de Spearman, etc.).

Las variables con valores de coeficiente de correlación más cercanos a 1 muestran una fuerte correlación positiva, los valores más cercanos a -1 muestran una fuerte correlación negativa y los valores más cercanos a 0 muestran una correlación débil o nula. Una matriz de correlación es muy útil para detectar y eliminar datos que no aporten información significativa para el modelo

```
import pandas as pd

# Convertimos los datos a un datarame en caso de no serlo
df_X = pd.DataFrame(X)

# Generamos la estadística descriptiva
estadistica = df_X.describe()

# Generamos la matriz de correlacion
matriz_correlacion = df_X.corr()
```

Código 14: Estadística descriptiva de un dataset

En cuanto al proceso de transformación, es importante entender el tipo de dato y el rango en el que se está manejando todo, un claro ejemplo es que cuando se trabaja con imágenes los valores de los píxeles son entre 0 y 255, sin embargo, trabajar con rangos tan grandes puede ser problemático y lo óptimo es llevarlo a un rango entre 0 y 1. Considerando esto, el proceso de transformación se lo puede abordar desde dos aristas principales, la normalización y el escalado de datos.

- Normalización de los Datos

La Normalización de los datos busca transformar las observaciones para que estas puedan ser descritas en base a la distribución normal, la misma que es también conocida como "curva de campana", se trata de una distribución estadística específica en la que aproximadamente las mismas observaciones caen por encima y por debajo de la media, la media y la mediana son iguales y hay más observaciones más cercanas a la media. La distribución normal también se conoce como distribución gaussiana.

Como se puede ver en el Código 15, scikit-learn proporciona funciones directas para ejecutar la normalización de datos, en este caso se emplea el mismo dataset anteriormente definido y la variable a normalizar es la que contiene los datos, no la que contiene las etiquetas.


```
from sklearn.preprocessing import Normalizer

# Normalizacion de los datos
normalization = Normalizer()
norm_x = normalization.fit_transform(X)
```

Código 15: Normalizar los datos

- Escalado de los Datos

El escalado de los datos busca transformar las observaciones para que estas se ajusten a una escala o rango específico, como por ejemplo que los datos se encuentren entre 0 y 100 o entre 0 y 1. Al escalar las variables, esto puede ayudar a comparar diferentes variables en igualdad de condiciones, ya que muchas veces los valores que toman las variables de un mismo dataset pueden ser muy diferentes.

Como se puede ver en el Código 16., scikit-learn proporciona funciones directas para ejecutar el escalado de datos, en este caso se emplea el mismo dataset anteriormente definido y la variable a normalizar es la que contiene los datos, no la que contiene las etiquetas.

```
from sklearn.preprocessing import StandardScaler

# Escalar los datos
scalling = StandardScaler()
scall_x = scalling.fit_transform(X)
```

Código 16: Escalar los datos

Adicionalmente al Standard Scaler presentado, se tienen varios métodos adicionales para realizar un escalado de los datos, como lo son el algoritmo Min-Max o el Max-Abs, ambos se encuentran presentes en la librería Scikit-learn. Así mismo para la normalización tendemos más técnicas disponibles en el siguiente enlace:

<https://scikit-learn.org/stable/modules/preprocessing.html#>

- Procesamiento de datos nulos y vacíos

Muchos datasets del mundo real contienen valores vacíos o nulos, a menudo codificados como espacios en blanco, NaN u otros marcadores de posición. Sin embargo, estos valores pueden alterar el comportamiento regular de los modelos de aprendizajes, sesgando de cierta manera los resultados o reduciendo la precisión de estos.

Una estrategia básica para usar conjuntos de datos incompletos es eliminar filas y/o columnas enteras que contengan valores faltantes. Sin embargo, esto tiene el precio de perder datos que pueden ser valiosos (aunque estén incompletos). Una mejor estrategia es imputar los valores faltantes, es decir, inferirlos de la parte conocida de los datos y remplazarlos, algunas maneras de hacerlo son mediante estimaciones como la mediana, la media o a partir de valores cercanos al valor vacío con una interpolación. Las diversas formas de ejecutar este procesamiento de datos nulos o vacíos se pueden ver en el Código 17.

```
import pandas as pd

# Convertimos los datos a un dataframe en caso de no serlo
df_X = pd.DataFrame(X)

# Diversas maneras de tratar datos faltantes

# Eliminamos las filas con valores faltantes, que en este caso
# estaban marcados con NaN
df_no_Nan = df_X.dropna()

# Eliminacion de todas las columnas con un porcentaje faltante (toda
# columna con hasta 20% de datos faltantes)
df_no_Nan_col = df_X.loc[:,df_X.isnull().sum()< 0.2*df_X.shape[0]]

# Reemplazamos valores faltantes con la funcion interpolacion
df_inter = df_X.interpolate(limit_direction = "both")

# Reemplazamos valores faltantes con la media
df_media = df_X.copy()
for col in cols:
    df_media[col].fillna(df_media[col].mean,inplace = True)
```

Código 17: Maneras de tratar valores faltantes

En términos del desarrollo de modelos de aprendizaje, estos conjuntos de datos son incompatibles con los estimadores de scikit-learn que asumen que todos los valores en una matriz son numéricos y que todos tienen un significado.

- Preparación y separación de datos

Una pieza importante en el desarrollo de algoritmos de aprendizaje automático es la creación de datasets de prueba y entrenamiento, esto debido a que el modelo debe aprender sobre un conjunto de datos, pero debe ser probado sobre otro, con el fin de evitar que el modelo no sea capaz de generalizar y únicamente ‘memorice’. A este proceso donde el modelo es incapaz de aprender y responder a datos distintos que a los de entrenamiento se lo conoce como overfitting o sobre entrenamiento, esto se puede ver de manera gráfica en la Ilustración 8.

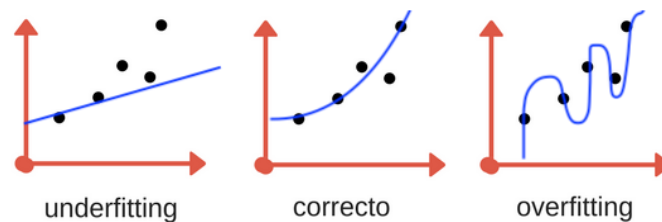


Ilustración 8: Representación gráfica de tipo de modelos

La manera óptima de seleccionar los conjuntos de entrenamiento y pruebas es a partir de elección aleatoria de datos sobre el dataset, esto se lo puede ejecutar en Python con el código como se muestra en el Código 18.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.25, random_state = 0)
```

Código 18: Separar datos en train y test

1.1.3.3. Desarrollo de modelos de aprendizaje

Una vez se tiene los datos limpios, escalados, preprocesados y separados en conjuntos de entrenamiento y prueba, es momento de desarrollar modelos de aprendizaje, para poder abordar de mejor manera este desarrollo se ahondará en los tipos de aprendizaje en los capítulos subsiguientes, sin embargo, en este momento es importante entender el concepto de parámetros e hiperparámetros.

Por un lado, tenemos los parámetros los mismos que son las variables que se estiman durante el proceso de entrenamiento con los conjuntos de datos. Por lo que sus valores no se configuran manualmente, sino que son obtenidos y van cambiando constantemente a través de las iteraciones del algoritmo, son la parte más importante en

los modelos de aprendizaje automático, ya que es la parte de los modelos que se aprende de los datos.

Son necesarios para realizar las predicciones. Además de definir la capacidad del modelo para resolver un problema dado y en su mayoría la manera en la que se los encuentre o varíe es a partir de funciones de optimización.

Por otro lado, tenemos los hiperparámetros, los mismos que no cambian durante la ejecución del modelo, si bien es cierto que deben ser optimizados, esto no son encontrados por el modelo, estos son en su mayoría provistos y designados manualmente al inicio de la ejecución del modelo.

Como se menciona, el valor óptimo de un hiperparámetro no se puede conocer a priori para un problema dado. Por lo que se tiene que utilizar valores genéricos, reglas genéricas, los valores que hayan funcionado anteriormente en problemas similares o buscar la mejor opción mediante prueba y error.

1.1.3.4. Entrenamiento y pruebas

Una vez se tiene el modelo con hiperparámetros configurados, es importante entrenarlo y medir su rendimiento. Para el proceso de entrenamiento se emplea el dataset de train que se había separado con anterioridad, la manera de entrenar el modelo es pasarlo con entrada de datos al modelo que se construye, se verán ejecuciones más detalladas en los capítulos venideros.

Con respecto al aspecto de pruebas y medir el rendimiento, existen muchas métricas que se pueden emplear, como el Accuracy, Matriz de confusión, Curvas ROC, entre otras.

- Matriz de Confusión

Es una métrica para medir el desempeño de nuestro algoritmo de aprendizaje, en donde cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real., o sea en términos prácticos nos permite ver qué tipos de aciertos y errores está teniendo nuestro modelo a la hora de pasar por el proceso de aprendizaje con los datos.

Como se puede ver en la Figura 12., la matriz de confusión brinda varios valores a partir de los cuales es posible calcular métricas como: precisión, exactitud, sensibilidad y especificidad.

VALORES PREDICCIÓN	VALORES REALES	
	Verdaderos positivos	Falsos Positivos
Falsos Negativos		
Verdaderos Negativos		

Ilustración 9: Matriz de confusión



Módulo 5

Visualización Web de Modelos de Machine Learning

Objetivos

- Desarrollar aplicaciones web para compartir modelos de machine learning.
- Comprender todas las herramientas disponibles en Streamlit para publicar aplicaciones web.

1. Visualización Web de Modelos de Machine Learning

Después de elaborar nuestro modelo de análisis de datos e inteligencia artificial, es el momento de presentarlo al público. Dentro de este contexto, Python permite diseñar páginas web mediante la librería **Streamlit**.

1.1. ¿Qué es Streamlit?

Es una herramienta de uso libre, que permite a los “científicos de datos” construir de manera interactiva herramientas web sin ninguna experiencia en desarrollo front-end.

1.2. ¿Por qué añadir Streamlit a tus librerías en Python?

- Uso libre.
- Simple de programar.
- Flexible, puede utilizarse para herramientas de análisis de datos hasta inteligencia artificial.

1.3. ¿Cómo prepararse para crear tu primer herramienta web?

- Es necesario que se posea tanto las bases de datos a utilizar como los modelos de machine learning previamente entrenados.
- Se necesita un editor de código que permita escribir las líneas de código necesarias para el desarrollo de la aplicación. Los editores de código que pueden utilizarse son: Sublime Text o Visual Studio Code.

1.4. Flujo de desarrollo.

Cada vez que se desee actualizar la aplicación, se debe guardar el archivo fuente. Cuando se hace esto, Streamlit detecta que existe un cambio y pregunta si deseas volver a correr la aplicación.

Selecciona “Always rerun” para que se actualice automáticamente cada vez que el código fuente cambie.

1.5. Formas de mostrar diferentes tipos de datos.

Existen algunas maneras de mostrar en Streamlit. Las siguientes son algunas de las estructuras de datos que pueden incluirse:

- Texto.
- Tablas de datos.
- Diagramas de barras, líneas, etc.

- Mapas

Además de poder presentar de diferentes formas los datos, Streamlit permite configurar la página web con diferentes objetos o “widgets”. Entre los diferentes elementos que se pueden añadir están:

- Botones.
- Barras de desplazamientos.
- Cajas de selección.
- Cajas de chequeo.



Módulo 6

Aplicaciones de la Ciencia de Datos en tu vida profesional

Objetivos

- Conocer la aplicación de la ciencia de datos en el fútbol.
- Conocer los conceptos del análisis de opinión pública.
- Comprender el uso de herramientas tecnológicas para analizar los datos.

1. Aplicaciones de la Ciencia de Datos en tu vida profesional

1.1. Ciencia de datos en el análisis futbolístico.

El fútbol como deporte es un juego que involucra un montón de acciones para determinar un resultado, estas previstas por 11 jugadores con facultades técnicas y tácticas entrenadas a lo largo de la semana. Aún con todo lo planificado, lo único seguro que rodea a este deporte es lo incierto de su esencia y que, dentro de cada jugada, siempre decidirán los futbolistas con todo lo impredecible que eso significa. La naturaleza de ser un juego espontáneo jamás cambiará, pero la ciencia de datos es una herramienta utilizada en la élite del fútbol que al menos aporta a reducir el margen de error de los partidos y poder controlar todo lo que sucede en los 90 minutos. Los deportes también se miden por patrones.

Actualmente el fútbol es ciencia, también matemáticas o al menos una forma didáctica de aprenderlas. La parábola que toma el balón luego de un disparo, la triangulación que se genera en una posesión de balón para atacar: es fútbol y también matemáticas.

Los números cada vez juegan un papel más importante en el fútbol: goles, asistencias, porcentaje de posesión y pases, frecuencia de intercepciones o acciones defensivas, son algunas de las estadísticas con las que se evalúan a los equipos y jugadores dentro de un informe para preparar un partido.

Las clásicas pizarras de entrenamiento mutaron para convertirse en ordenadores, con los detalles del ángulo con el que se cobra un tiro de esquina, las líneas de pase, los mapas de calor, entre otros. Todas estas son cuestiones de Big Data y sistemas en red.

La gran cantidad de información que se recibe luego de un partido es lo que se denomina Big Data en fútbol, y esta permite obtener a datos a los cuales se puede profundizar en el conocimiento del juego. Así se mejora en la eficacia de los deportistas en los entrenamientos, como también en la salud y la alimentación.

Durante un partido de fútbol se generan aproximadamente 8 millones de datos, en los que aparte de los jugadores, también se toma en cuenta al arbitraje y al balón. Esta información que no se puede percibir a simple vista porque el ojo humano solo puede capturar el 30% de lo que sucede en el juego, se almacena para más tarde ser trabajada por el Big Data y así sacar conclusiones de lo que pasó en la cancha.

Estas son algunas de las áreas principales en las que interviene el Big Data en el fútbol:

Decisiones técnicas basadas en los datos de los deportistas en plena competencia, evaluados con herramientas de análisis que permiten identificar los errores técnicos y tácticos cometidos por el jugador en tiempo real, y así poder corregir.

Análisis predictivo que analiza la gran cantidad de fuentes de datos existentes (biometría, árbitro, equipo técnico o rendimiento de nuevos jugadores). Este modelo de machine learning permite identificar a los jugadores más aptos, reducir el riesgo de lesiones y desarrollar predicciones de los jugadores propios y de los rivales.

Marketing de eventos para armar campañas personalizadas y predecir que publicidad o actividad de comunicación atraerá a más aficionados.

Eficiencia de recursos con instalaciones inteligentes en los estadios para controlar con mayor precisión el rendimiento de todos los recursos.

El Big Data es fundamental en el deporte actual. En Inglaterra ya es obligatorio que los clubes tengan en su plantilla a matemáticos de datos y algunos equipos de elite utilizan modelos estadísticos para fichar jugadores. Ese es el caso del Sevilla que es una de las instituciones que mejor fichan en el mundo, comprando jugadores a precios cómodos y luego vendiéndolos a cifras exorbitantes, de esta manera también contribuyen a competir en La Liga de España, dentro de esos modelos estudian la prevención de lesiones o incluso el valor de mercado de los jugadores.

Hay varias aplicaciones en el fútbol que sirven para recoger el mayor número de datos y así beneficiar a los jugadores y cuerpos técnicos. Estas son algunas de las variables analizadas por partido y equipo según la herramienta Instat:

Indicadores de éxito: goles, oportunidades de gol, oportunidades de gol exitosas, tiros, tiros a puerta, eficacia en tiros a puerta, tiros fuera, tiros bloqueados, pases de gol y eficacia en pases de gol.

A nivel global: tiempo efectivo de juego, porcentaje de posesión y duración media de las posesiones.

A nivel ofensivo: pases, pases exitosos, tiros, centros, eficacia en los centros, duelos, duelos ganados, duelos aéreos, regates, regates realizados con éxito, pérdidas de balón, ataques posicionales, ataques posicionales con tiro, contraataques, contraataques con tiro, acciones a balón parado, tiros libre directos, corners, corners con tiro, eficacia en el corners, ataques por el carril izquierdo, ataques por el carril central y ataques por el carril derecho, etcétera.

A nivel defensivo: faltas realizadas, tarjetas amarillas, duelos defensivos, duelos defensivos ganados, entradas, entradas realizadas con éxito, interceptaciones de balón, recuperaciones de balón y recuperaciones de balón en campo rival.

Con los valores obtenidos de los algoritmos, se puede conocer el valor del jugador en el mercado, goles, asistencias, movimientos, su historial de lesiones y hasta su comportamiento en la cancha.

Los datos y entender la forma de organizarlos y utilizarlos, componen el mayor porcentaje de éxito en un equipo.

1.2. Ciencia de datos en la opinión pública.

Las sociedades, desde sus primeros rasgos de civilización, se han evidenciado como conjuntos complejos conformados por individuos diversos que realizan múltiples acciones (salir a caminar, decidir dónde dormir, comer, trabajar, recrearse, etc), muchas veces pensando en la adaptación y motivados por los incentivos de su entorno. A pesar de la autonomía en cuestión, los seres humanos hemos decidido asignar responsabilidades a ciertos grupos para que manejen los recursos, el orden y la planificación del espacio en el que nos desarrollamos, mediante las reglas de lo que denominamos un sistema democrático (en su mayoría de preferencia occidental).

Sí bien los gobiernos (denominación que hemos empleado para el grupo de autoridades) toman las decisiones dentro de esa administración bajo los criterios, ideologías e incentivos políticos que tienen, el análisis de la opinión pública respaldado con la ciencia de datos representa una herramienta increíble para obtener una mayor eficiencia y sostenibilidad política en sus resultados a lo largo del tiempo. De acuerdo al informe del BID, solo en el 2012 se han producido y recopilado más datos que en toda la historia previa hasta dicha fecha. Hablamos de una cantidad de 2.5 EB (Exabytes al día), lo que equivale a 2.5 quintillones de bytes, provenientes de la red y del territorio físico.

Con ese framing y viviendo en sistemas democráticos, la opinión pública es muy importante porque define el éxito o el fracaso de un proyecto político, básicamente “el poder es del pueblo”. Sí, mucho está en juego, desde la victoria (no implica necesariamente quedar primero) que podemos tener en las elecciones como candidatos o como partido, hasta la conclusión en buenos resultados del plan en política pública que queramos implementar, al ser ya autoridades al servicio público.

Ahora, sí bien existe la utilidad monetaria y política del análisis de datos para las autoridades, la gran cantidad de información generada y que sigue apareciendo a nivel público, se transforma en una oportunidad invaluable para mejorar la vida de las personas a través de obras y acciones que se convierten en intangibles, para una ciudad o un país.

Estos son algunos de los conceptos para la preparación y la interpretación de los resultados del análisis de datos en la Opinión Pública:

Análisis Cuantitativo: Utiliza la recolección de datos para probar supuestos o hipótesis planteadas previamente con base en la medición numérica y el análisis estadístico de variables.

Capital político: Es el crédito reputacional que posee un político, el cual proviene tanto del entorno político como de los resultados que este va alcanzando en el desarrollo de su carrera.

Gobernabilidad: La capacidad que tiene el gobierno para implementar políticas públicas y leyes a través de la legitimidad social, con el fin de que sus resultados no se vean viciados por otros agentes de forma intencional.

Tejido social: Es el grado de pertenencia, solidaridad y cohesión existente en un grupo de individuos, el cual es fundamental entender para garantizar el bienestar de los habitantes en una ciudad.

Opinión Pública: La opinión pública es la constante manifestación, variación y evolución de las opiniones y posteriores acciones de los individuos sobre la gestión y el manejo de los recursos públicos por parte de las administraciones gubernamentales.

Gestión Pública: Es un conjunto de acciones sistemáticas que realizan las autoridades de un gobierno con los recursos de una nación.