Economía y Ciencia de los Datos

Introducción a Python

Carlos Alvarado & Pablo González

8 de agosto de 2022

Bases necesarias y motivación

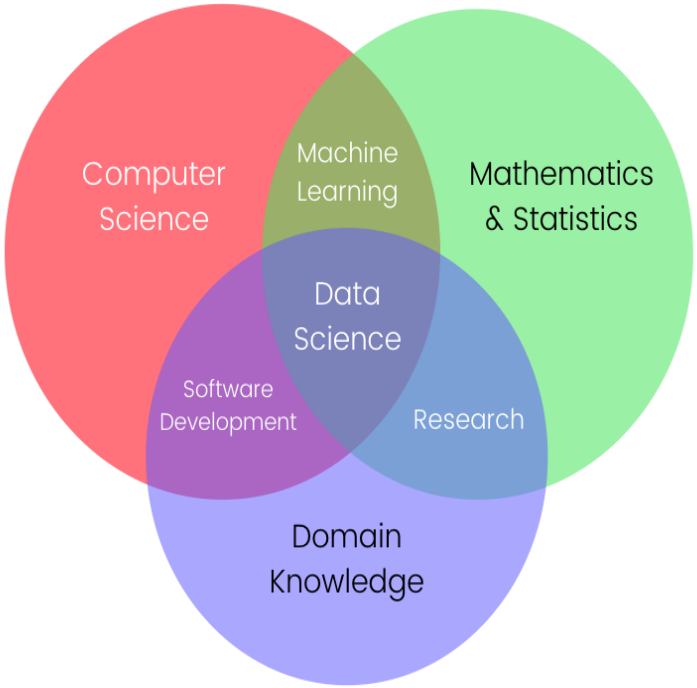
Requisitos

Mínimos:

- Algebra Lineal, probabilidad, estadística y econometría (A nivel de pregrado).
- Experiencia en algún lenguaje de programación.

Deseables:

- Proactividad y disposición a aprender: Muchas veces la respuesta está en internet. Ser autodidacta también es relevante.
- Interés en un área: Aprenderemos herramientas. Ustedes les darán valor.
- Disciplina: Algunos modelos tardan en ser entrenados. Procrastinar es costoso.



¿Por qué tomar este curso?

Mediante este curso no se volverán expertos en un tema, pero les brindará conocimiento básico sobre una variedad de herramientes útiles en la academia e industria.

Entre otras herramientas, veremos:

- Bases de Python para Data Science.
- Uso de APIs y Web-scraping para extraer datos.
- Uso de SQL (lenguaje para bases de datos estructuradas)
- Algoritmos predictivos supervisados y no supervisados. (Conceptos básicos y aplicación)

En resumen: La idea es que interactúen con nuevas tecnologías, y fuentes de información distintas a las empleadas en clases pasadas, además de obtener mayor experiencia analizando datos.

¿Cómo me podría preparar?

Las primeras semanas pueden ser complejas. El curso busca ser auto-contenido, pero requiere atención y práctica para que aprendan a implementar sus soluciones.

Acciones Recomendadas

- Explorar GitHub.
- Repasar Python y Algoritmos: Leetcode / HackerRank / CodeWars, etc.
- Motivarse con papers/proyectos en Paperswithcode, arXiv o Kaggle.

Ante la duda $\rightarrow$ **preguntar**.

Repaso de Programación / Introducción a Python

Códigos

Ir a Notebook

Un ejercicio intermedio

¿Qué hace el siguiente código? ¿Cómo podría mejorar?

```
import numpy as np
def f1(input_variable):
    for n in range(2,input_variable + 1):
        boolean_variable = 1
        for i in range(2, max(2,n-1) ):
            if np.floor(n/i) == n/i:
                boolean_variable = 0
        if boolean_variable == 1: print(n)
    return None
```

Si bien la eficiencia no es el foco del curso, cabe destacar:

- Importancia del cuidado al detalle.
- No olvidar las condiciones de salida.
- "Pequeños" cambios pueden afectar fuertemente a su código cuando n es grande.

Aclaración: Complejidad vs Tiempo de ejecución

Estas funciones:

- Cumplen la misma tarea.
- Tienen complejidad de O(n).
- No tardan lo mismo en ejecutar.

```
def contar_elementos(elementos):
    contador = {}
    for i in elementos:
        if i not in contador: contador[i] = 1
        else: contador[i] += 1
    return contador
counter1 = contar_elementos(items)

from collections import Counter
counter2 = Counter(items)
```