

## **Trabajo de Laboratorio N°1:**

### ***Regresión Lineal***

- **Materia:** Inteligencia Artificial.
- **Carrera:** Ingeniería en Informática
- **Docente:** Ing. Federico Gabriel D'Angiolo
- **Integrantes:**
  - Cabot, Lucas.
  - Calonge, Federico.
  - Lew, Imanol.

# **Índice**

<b>1- Objetivo.</b>	<b>(Pág. 3)</b>
<b>2-Introducción y Conceptos Previos.</b>	<b>(Pág. 4)</b>
<b>2.1-Conceptos de estadística.</b>	<b>(Pág. 4)</b>
<b>2.2-Regresión Lineal: Modelo Matemático.</b>	<b>(Pág. 6)</b>
<b>2.3-Data Sets e importancia del Análisis de Datos.</b>	<b>(Pág. 8)</b>
<b>2.4-Entorno de Desarrollo y Bibliotecas utilizadas.</b>	<b>(Pág. 9)</b>
<b>3-Desarrollo.</b>	<b>(Pág. 14)</b>
<b>3.1-Implementación en Código: Caso A.</b>	<b>(Pág. 14)</b>
<b>3.1.1-Lectura y análisis del Dataset.</b>	<b>(Pág. 14)</b>
<b>3.1.2-Implementación del modelo de Regresión Lineal.</b>	<b>(Pág. 17)</b>
<b>3.1.3-Resultados.</b>	<b>(Pág. 18)</b>
<b>3.2-Implementación en Código: Caso B.</b>	<b>(Pág. 19)</b>
<b>3.2.1-Lectura y análisis del Dataset.</b>	<b>(Pág. 19)</b>
<b>3.2.2.-Resultados.</b>	<b>(Pág. 20)</b>
<b>4-Conclusiones.</b>	<b>(Pág. 24)</b>
<b>5-Mejoras a desarrollar.</b>	<b>(Pág. 25)</b>
<b>6-Bibliografía.</b>	<b>(Pág. 26)</b>

## 1- Objetivo.

Este Trabajo de Laboratorio consistirá en analizar y manipular datos, y, si corresponde, aplicar la técnica de Regresión Lineal y, a partir de esta, obtener modelos y visualizaciones de los datos que nos permitan predecir el comportamiento de distintas variables.

### Se analizarán 2 casos:

- **Caso A:** Disponemos de un **dataset estático** (no se actualiza, es un archivo CSV) que muestra la presión medida en dos puntos distintos de un datacenter. Para conocer si la presión se propaga de manera lineal entre estos dos puntos del lugar analizaremos distintos conceptos de probabilidad, tal como la media y el desvío de cada dataset. Luego construiremos un modelo matemático basado en Regresión Lineal para obtener la relación de presión entre estos dos puntos. Para este caso debemos considerar si el modelo es lineal o no utilizando las herramientas matemáticas necesarias (llevadas a código).
- **Caso B:** Disponemos de un **dataset dinámico** (ya que se actualiza constantemente) sobre la evolución del COVID-19 a nivel global. Realizaremos una exploración y visualización de datos obteniendo así distintos gráficos, y de esta manera, tratar de encontrar patrones en el crecimiento de contagios, muertes y recuperados en Argentina; y además compararemos estos resultados con otros países.

## 2-Introducción y Conceptos Previos.

En esta Sección describiremos los temas teóricos y matemáticos para tratarlos a lo largo del Informe y llevarlos a cabo en los Algoritmos de Python.

### 2.1-Conceptos de estadística.

**2.1.1-Muestras y Población:** Se los denomina **muestras** o **valores** a todos los datos obtenidos, mientras que, al conjunto de estos se los denomina **población**. Esta población estará conformada por las **N** muestras o valores. Para el **Caso A** nuestras muestras son cada uno de los datos de las presiones A y B; y para el **Caso B** nuestras muestras son cada una de las personas (sean infectados, muertos o recuperados).

**2.1.2. Media:** Es una herramienta que permite describir la **tendencia** de un conjunto de **N** valores o muestras, que se encuentran dentro de una **población**. Matemáticamente, la media es la **suma de cada uno de los valores, dividido por el total N**. La *Fórmula 1* describe el cálculo de la media para una distribución NORMAL (generalmente los datos siguen esta distribución).

$$\mu = \frac{1}{N} \cdot \sum_i^N X_i$$

*Formula 1*

-Donde:

$\mu$  = Media de la muestra.

$N$  = Cantidad de valores de la muestra ó población obtenida.

$X_i$  = Muestra i-ésima.

**2.1.3 Desvío:** Herramienta que muchas veces no resulta suficiente para describir a un conjunto de datos. Por esta razón, se puede utilizar el desvío estándar ( $\sigma$ ), el cual **describe cuán dispersos se encuentran los valores respecto de la media**. Cuánto más grande sea el desvío estándar, más grande será la dispersión entre los valores de la población. Matemáticamente se la describe en la *Fórmula 2*.

$$\sigma = \sqrt{\frac{1}{n} \cdot \sum (x_i - u)^2}$$

*Fórmula 2*

-Donde:

$\sigma$  = Desvío estándar  $\mu$  = Media de la muestra.

$N$  = Cantidad de valores de la muestra o población obtenida.

$X_i$  = Muestra i-ésima.

**2.1.4. Varianza:** Similar a la Varianza, ya que también mide la dispersión de los valores, pero matemáticamente se la obtiene de distinta forma (observar *Fórmula 3*).

$$\sigma^2 = \frac{1}{n} \cdot \sum (x_i - u)^2$$

*Fórmula 3*

Donde:

$\sigma^2$  = Varianza.

Es decir, que si bien ambos conceptos miden la dispersión de las muestras de la población, el desvío estándar tiene las mismas unidades que la media, por eso es que muchas veces se suele utilizar el desvío a la varianza. En conclusión **la varianza mide algo similar al desvío pero en otras unidades.**

**2.1.5 Covarianza:** Es la **medida de la relación que existe entre un par de variables aleatorias (X e Y)** en donde un cambio en una de ellas es correspondido por un cambio equivalente en otra variable, siempre que exista una relación entre dichas variables. Matemáticamente se la puede calcular como en la *Fórmula 4*.

$$\text{cov}(x_i, y) = \frac{1}{n} \sum (x_i - \bar{x}_i)(y - \bar{y})$$

*Fórmula 4*

La covarianza puede tomar cualquier valor entre  $-\infty$  a  $+\infty$ , donde un valor negativo es un indicador de una relación negativa, mientras que un valor positivo representa una relación positiva. Por lo tanto, **cuando el valor es cero, indica que no hay una relación lineal directa.**

**2.1.6. Factor/Coeficiente de Correlación:** Si la covarianza de X e Y se la divide por el producto de los desvíos estándar de X e Y, el resultado es una cantidad sin dimensiones llamado coeficiente de correlación. Matemáticamente se calcula mediante la *Fórmula 5*.

$$\rho = \frac{\text{cov}(x,y)}{\sigma_x \cdot \sigma_y}$$

*Fórmula 5*

El coeficiente  $\rho$  se encuentra contenido en el intervalo  $-1 < \rho < 1$ . El caso de  $\rho = 0$  indica la ausencia de cualquier asociación lineal, mientras que los valores  $-1$  y  $1$  indican relaciones lineales perfectas. En forma más sencilla: **nos dice qué tan lineal es la relación entre X e Y**. Estamos NORMALIZANDO la covarianza. De esta manera  $\rho$  la ponemos entre  $1$  y  $-1$  y así... si  $\rho$  vale  $1$  entonces X e Y son lineales y de pendiente positiva. Si  $\rho$  vale  $-1$  son lineales y de pendiente negativa. Y si  $\rho$  vale  $0,1$  entonces NO tendrían una relación lineal, ya que NO podríamos encontrar una recta que relacione ambas variables (esto suponemos generalmente que sucede cuando  $\rho$  está aproximadamente en el rango:  $-0,7 < \rho < 0,7$ ).

## 2.2-Regresión Lineal: Modelo Matemático.

La regresión lineal es un procedimiento estadístico que busca establecer una relación directa o inversa entre dos o más variables. Permite realizar una **predicción del comportamiento** de alguna variable en un determinado punto o momento. La Regresión nace de la necesidad de **ajustar alguna función a un conjunto de datos**.

De esta manera, deberemos comprobar que el conjunto de datos es lineal y luego aplicar esta técnica para así poder predecir comportamientos de nuestras variables. Matemáticamente, un modelo lineal se basa en que es posible aproximar valores de salida a través de un proceso de Regresión basado en la *Fórmula 5*.

$$f(x_i) = \alpha_0 + \sum_{i=1}^m \alpha_i x_i \quad \text{donde } A = \{\alpha_0, \alpha_1, \dots, \alpha_m\}$$

*Fórmula 5*

-Donde:

**X** representa un dataset de vectores de entrada con valores reales:

$$\mathbf{X} = \{x_1, x_2, \dots, x_n\} \text{ donde } x_i \in \mathbb{R}$$

**Y** representa cada valor real asociado a estos vectores de entrada:

$$\mathbf{Y} = \{y_1, y_2, \dots, y_n\} \text{ donde } y_i \in \mathbb{R}$$

Para obtener las constantes del vector A, existen dos métodos:

- El primero es conocido como el **método de mínimos cuadrados** (es el que utilizaremos para predecir nuestros modelos de datos) en el cual se eligen las constantes de manera que **minimicen la suma de los cuadrados de los residuos (RSS o Residual Sum of Squares)**. Llamando residuo a la diferencia entre el valor de la predicción y el valor real en un punto. Ver el cálculo de los RSS en la *Fórmula 6*.

$$RSS(A) = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$
$$RSS(A) = \sum_{i=1}^N (y_i - f(x_i))^2$$

*Fórmula 6*

- El segundo método se lo denomina **Descenso del gradiente**, que consiste en calcular las derivadas parciales de todas las variables que intervienen en el modelo de datos y que son necesarias para poder predecir el vector “Y” del sistema, en otras palabras consiste en calcular la derivada de la función en diferentes puntos con el objetivo de calcular la pendiente de dicho parámetro, en donde mi derivada en ese punto valga 0 (cero), y de esta manera poder obtener un mínimo local de la función. En Sistemas bidimensionales, pueden existir varios puntos locales mínimos, y lo que se buscará es obtener derivadas parciales en distintos puntos que conformará un vector que indicará la dirección en la que la pendiente asciende, este vector resultante se lo denomina Gradiente, cómo queremos obtener el mínimo, tomaremos el sentido opuesto de este vector.

En nuestro trabajo realizaremos la exploración y análisis de datos con **sistema bidimensionales** (x e y). Este es un caso de **Regresión Simple**, el cual se da cuando existe un único vector de entrada y un vector de salida. De esta manera, la expresión se reduce a la *Fórmula 7*.

$$f(x) = a_0 + a_1 \cdot x$$

*Fórmula 7*

-Donde:

$f(x)$  = vector de salida.

$f(x)$  = vector de entrada.

$a_0, a_1$  = coeficientes obtenidos a partir de los datos.

## 2.3-Data Sets e importancia del Análisis de Datos.



Un **Data Set** (conjunto de datos), es una tabla de una base de datos o, matemáticamente, una matriz estadística de datos. Cada columna de la tabla representa una variable del Data Set; y cada fila representa a un miembro determinado del conjunto de datos.

Los datasets que analizaremos en este Trabajo Práctica son los siguientes:

- Para el caso A utilizamos un Dataset “estático” (un archivo CSV) con datos de presiones en un determinado rango de tiempo. En una columna tenemos las presiones para un punto y en la otra columna las presiones para el otro punto. Este Data Set contiene 999 filas y 2 columnas (Cada columna representa la presión en un punto determinado); podemos observar como ejemplo 10 registros de este dataset en la *Imagen 1*.
- Para el caso B utilizamos un Dataset “dinámico” (archivo CSV que se actualiza constantemente a medida que surjan nuevos casos de COVID-19 y se suban al Repositorio). Este Dataset tiene 4 columnas ‘fijas’ (Provincia/Estado, Pais/Region, Lat y Long) y luego cientos de columnas ‘variables’ (son fechas, se agrega 1 por cada día que pasa). Una porción de este Data Set lo observamos en la *Imagen 2*.

Previamente a utilizar algoritmos de predicción (como el caso de Regresión Lineal) necesitamos que los datos estén LIMPIOS... por esto es importantísimo manipular y analizar estos datos en una primera instancia para luego si, meter estos datos limpios en los algoritmos y que estos funcionen correctamente; ya que de lo contrario las máquinas podrían clasificar o predecir de forma errónea. Este análisis previo sobre los datos debe ser minucioso ya que puede haber valores incoherentes o absurdos. Esto lo logramos analizando los conceptos estadísticos que vimos previamente (media y varianza principalmente).



 **Datos\_Presión.csv** 15.6 KB 

1	1000.83,1000.33
2	1000.64,1000.24
3	1000.81,1000.48
4	1000.80,1000.56
5	1001.28,1001.01
6	1001.69,1001.47
7	1002.02,1001.77
8	1001.64,1001.44
9	1001.17,1001.06
10	1001.52,1001.31

Imagen 1. Data Set - Caso A

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20
2		Afghanistan	33.0	65.0	0	0	0	0	0	0
3		Albania	41.1533	20.1683	0	0	0	0	0	0
4		Algeria	28.0339	1.6596	0	0	0	0	0	0
5		Andorra	42.5063	1.5218	0	0	0	0	0	0
6		Angola	-11.2027	17.8739	0	0	0	0	0	0
7		Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0
8		Argentina	-38.4161	-63.6167	0	0	0	0	0	0
9		Armenia	40.0691	45.0382	0	0	0	0	0	0
10	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0
11	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	0	0
12	Northern Territory	Australia	-12.4634	130.8456	0	0	0	0	0	0

Imagen 2. Data Set - Caso B

## 2.4-Entorno de Desarrollo y Bibliotecas utilizadas.

La plataforma de trabajo utilizada en nuestro proyecto es Anaconda, la cual es la plataforma Open Source más utilizada en la ciencia de datos y el aprendizaje automático que permite lograr de manera más eficiente el tratamiento masivo de volúmenes de datos (Big Data). Anaconda Navigator es una interfaz gráfica de usuario (GUI) de escritorio que permite lanzar aplicaciones y manejar paquetes sin la necesidad de usar comandos. En la *Imagen 3* podremos observar el front de desarrollo de la plataforma mencionada.

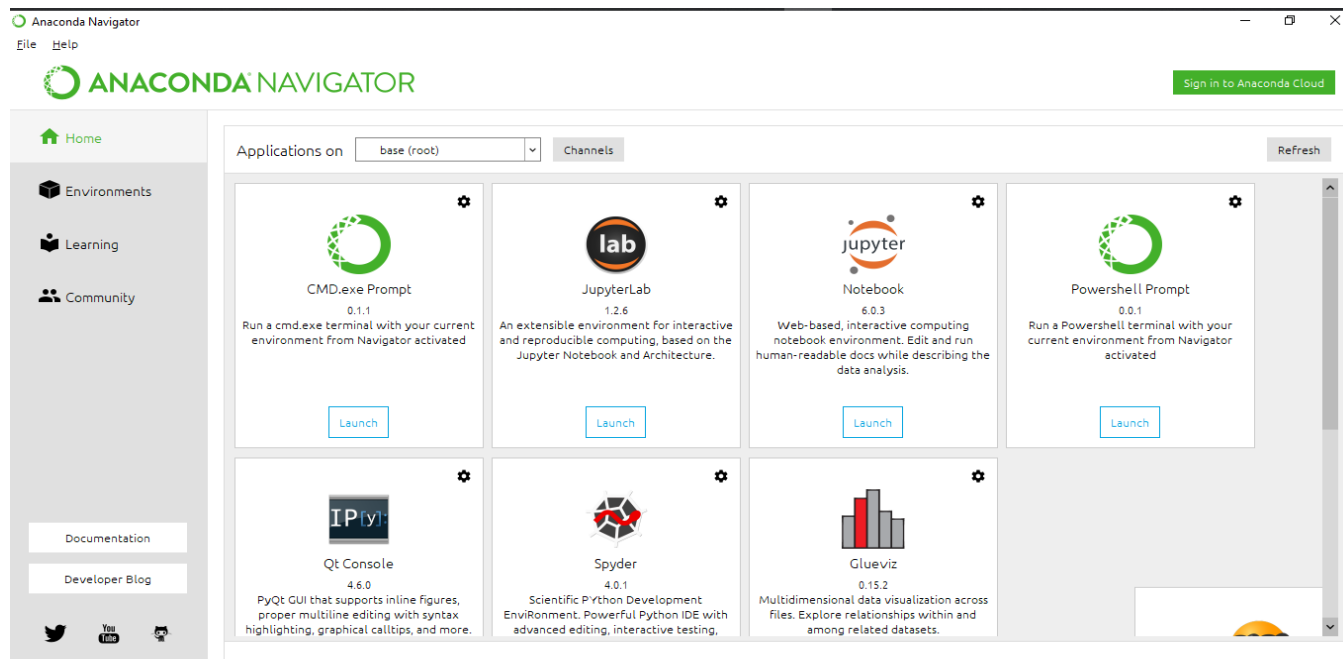
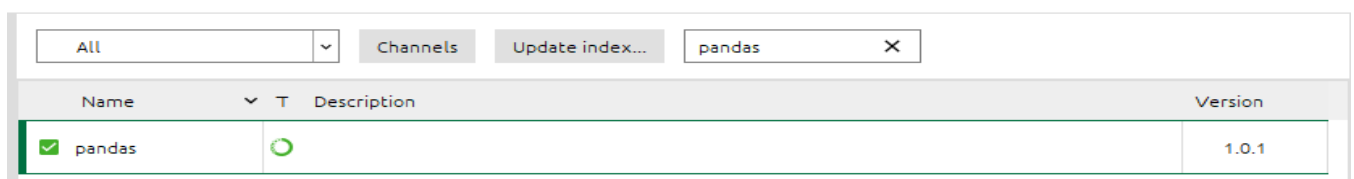


Imagen 3 - Anaconda Navigator

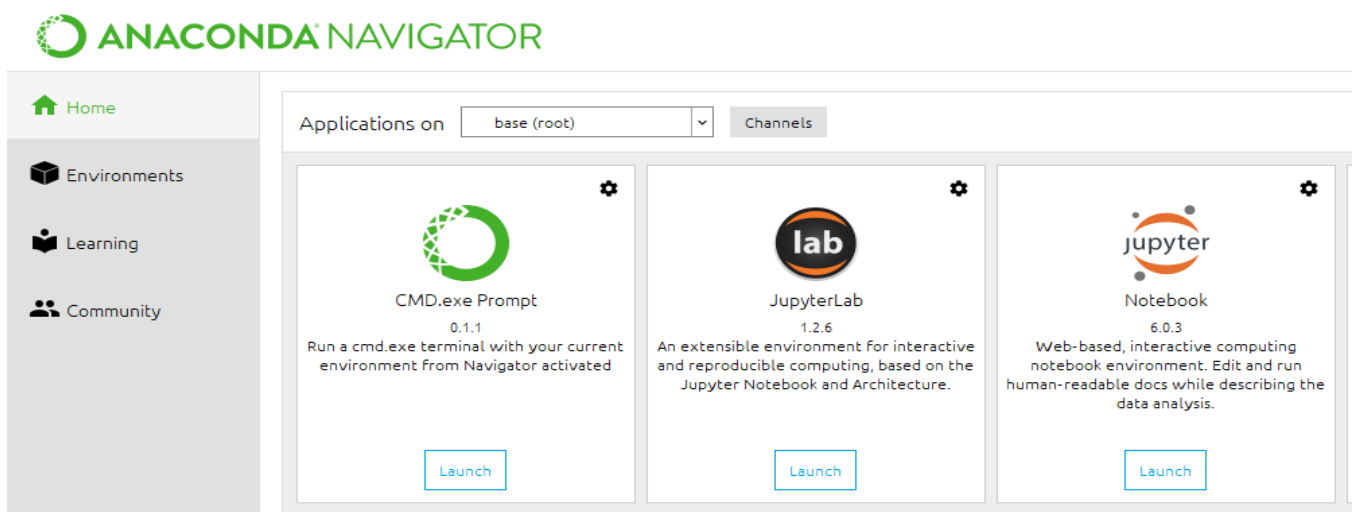
Anaconda aporta una gran ayuda en la gestión y administración de paquetes/librerías, ya que incluye más de mil (1000) paquetes y permite de esta manera abstraer al usuario de aprender cómo instalar una librería en particular, ya que directamente hacemos uso de la misma. Además, en caso de ser necesario instalar una librería, posee un entorno de instalación muy intuitivo y fácil de usar que simplemente se debe escribir el nombre de la librería, dar click en buscar y seleccionar instalar independientemente del sistema operativo que tengamos instalado. Asimismo, menciona la versión de cada paquete instalado, y esto es de gran utilidad a la hora de programar y ver la documentación oficial de la librería de acuerdo a su versión (Ver *Imagen 4*).



Name	Description	Version
✓ pandas		1.0.1

*Imagen 4 - Librerías Anaconda Navigator.*

Como se mencionó anteriormente, Anaconda permite lanzar varias aplicaciones, entre ellas RStudio, Spyder, QT Console, entre otras. La que finalmente nosotros utilizamos para el desarrollo de nuestro proyecto es **Jupyter Notebook**, el cual es un entorno computacional interactivo basado en la web (es una aplicación web) que nos permite crear una hoja de cuaderno, que es un documento de tipo Json, que utiliza la extensión .ipynb. Dicho entorno lo podemos ejecutar directamente desde Anaconda (Ver *Imagen 5*).



*Imagen 5 - Jupyter Notebook en Anaconda Navigator*

Jupyter Notebook, como dijimos anteriormente, utiliza hojas de cuadernos (Notebook). Estas hojas nos sirven para denotar documentos que contienen elementos de código, textos explicativos y demás elementos (tal como figuras, enlaces, ecuaciones y visualizaciones). De esta manera, Jupyter Notebook sirve a modo de “puente” entre el código y los textos explicativos... de modo que estos documentos son el lugar ideal para reunir una descripción de análisis y sus resultados, así como también, se pueden ejecutar para realizar el análisis de datos en tiempo real. Como una aplicación servidor-cliente, la aplicación Jupyter Notebook permite editar y ejecutar los notebooks a través de un navegador web. La misma se puede ejecutar en una PC sin acceso a Internet, o se puede instalar en un servidor remoto, donde se puede acceder a través de Internet.

---

### **Librerías de Python utilizadas:**

- **Numpy:**

Paquete fundamental para la computación científica con Python. Consiste en una extensión de Python, que le agrega mayor soporte para vectores y matrices n-dimensional, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. Además, permite trabajar con algebra lineal, transformada de Fourier y capacidades de números aleatorios. En este sentido, también se puede utilizar como un contenedor eficiente multidimensional de datos genéricos. Se pueden definir tipos de datos arbitrarios. Esto permite que NumPy se integre sin problemas y rápidamente con una amplia variedad de bases de datos.

- **Pandas:**

Paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Pandas nos facilita la manipulación de los datasets. Es una librería muy sencilla de usar. Es flexible y versátil, permitiéndonos cambiar la forma a los datasets si así lo necesitamos, filtrar información, realizar operaciones matemáticas con sus campos, etc.

Pandas depende de Numpy, la librería que añade un potente tipo matricial a Python.

Los principales tipos de datos que pueden representarse con Pandas son:

- ❖ Datos tabulares con columnas de tipo heterogéneo con etiquetas en columnas y filas. Se los denomina **Dataframes**.
- ❖ Series temporales.
- ❖ Panels (tablas 3D).

Por otro lado, Pandas proporciona funciones o métodos que permiten:

- ❖ Leer y escribir datos en diferentes formatos: CSV, Microsoft Excel, bases SQL, etc.
- ❖ Seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas.
- ❖ Fusionar y unir datos.
- ❖ Transformar datos aplicando funciones tanto en global como por ventanas.
- ❖ Manipulación de series temporales.

- Matplotlib:

Biblioteca de Python para la generación de gráficos a partir de datos contenidos en listas, arrays, diccionarios, Dataframes o series. Permite analizar el comportamiento de un conjunto de datos / data set y cómo éste evoluciona en una medida de tiempo o en función de una o más variables. Además, permite poder determinar la distribución que se asemeje al conjunto de datos, es decir, si sigue un comportamiento lineal, exponencial, distribución normal, entre otras. De esta manera, y con el uso de Data Science podremos predecir el comportamiento de un modelo y obtener una predicción del valor real en función de una o más variables.

- Scikit Learn:

Biblioteca para Python de aprendizaje automático y de código abierto. Admite el aprendizaje supervisado y no supervisado. También proporciona varias herramientas para el ajuste de modelos, el preprocesamiento de datos, la selección y evaluación de modelos, y muchas otras utilidades. Además, incluye: método de validación cruzada, varios conjuntos de datos / datasets, extracción y selección de características, etc.

Scikit Learn está basada en las siguientes librerías:

- ❖ Numpy.
- ❖ Pandas.
- ❖ SciPy.
- ❖ Matplotlib.
- ❖ IP [y].
- ❖ SynPy.

El uso de la librería Sickit Learn se puede dividir en 3 categorías principales:

- Clasificación (Clustering): Consiste en determinar a qué categoría pertenece un objeto. Por ejemplo detección de spam o reconocimiento de imágenes. Entre los algoritmos utilizados se puede mencionar: SVM , vecinos más cercanos , bosque aleatorio.
- Regresión: Predecir un atributo de valor continuo asociado con un objeto. Ejemplo de usos: respuesta a medicamentos, precios de acciones, precios de viviendas. Comparte los mismos algoritmos que el modelo de Clasificación.
- Agrupación: Consiste en la agrupación automática de objetos similares en conjuntos. Aplicaciones: segmentación de clientes o de góndolas de supermercados. Algoritmos que utiliza: k-medias , agrupación espectral , desplazamiento medio, entre otros.

### 3-Desarrollo.

#### 3.1-Implementación en Código: Caso A.

##### 3.1.1-Lectura y análisis del Dataset.

Para la lectura de los datos es necesario importar la librería de Python conocida como Pandas, dado que se trabajará con el tipo de datos denominado **Dataframe** (estructura básica de datos utilizado en la librería **Pandas**; es una colección ordenada de columnas con nombres y tipos). Para importar la librería Pandas se debe utilizar el siguiente código:

```
import pandas as pd
```

Luego, importamos el modelo de datos (dataset) que contiene las presiones medidas en dos puntos diferentes de un datacenter. Dado que el dataset tiene un formato csv, hacemos uso de la función de Pandas para leer este tipo de archivo. Para ello, utilizamos la siguiente función “read\_csv” de la siguiente manera:

```
dataset = pd.read_csv('TL_Nº1_mediciones_csv_Datos_Presión.csv',  
                      names = ("Presion 1", "Presion 2"),  
                      dtype = {'Presion 1': np.float64, 'Presion 2': np.float64})
```

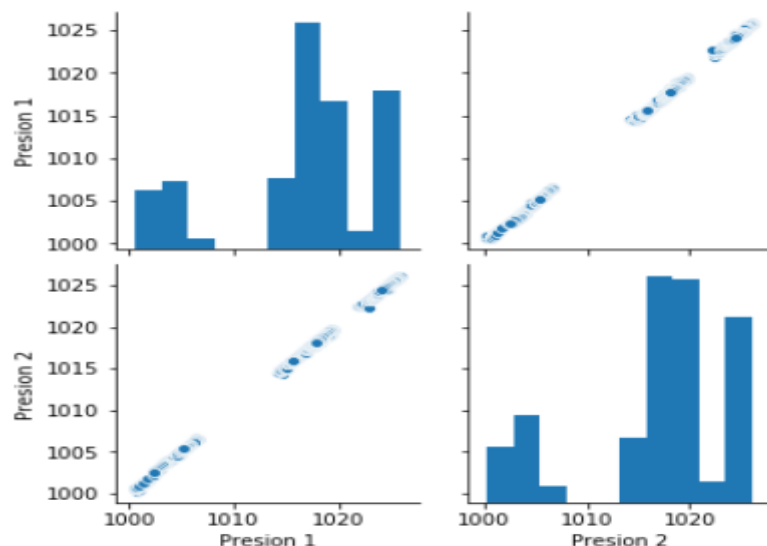
Donde el primer parámetro representa el nombre del archivo csv, en este caso está contenido en el mismo directorio que el archivo “.ipynb” del programa. Dado que el archivo csv no contiene los nombres de las columnas, fue necesario declarar el parámetro “names” indicando los nombres de las columnas que contendrá el dataset, y “dtype” especifica el tipo de datos que tiene el modelo, en este caso son todos de tipo flotante.

A continuación, se procedió a analizar la relación de las variables que intervienen en el modelo que son Presión 1 y Presión 2. Por lo tanto, se utilizó la función “pairplot” de la librería “seaborn” de Python. Para poder ejecutar esta función, primero es necesario importar la librería, de la siguiente manera:

```
import seaborn as sns
```

Luego ejecutamos la función “Pairplot” pasando como parámetro el dataset y analizamos los resultados obtenidos (Ver Gráfico 1):

```
sns.pairplot(dataset)
```



*Gráfico 1 – Resultados obtenidos*

Cómo el modelo presenta dos variables, nos basamos en la información representada en el 2do. y 3er. cuadrante que representa el comportamiento de los datos de las presiones 1 en función de las presiones dos y viceversa respectivamente. En este sentido, verificamos que tanto P1 (P2) ó P2 (P1) sigue un comportamiento lineal de acuerdo al gráfico obtenido. De esta manera nos damos un indicio del comportamiento del modelo de datos.

A continuación, procedemos a analizar la correlación de estas dos variables mencionadas, ejecutando el siguiente comando y analizando los resultados obtenidos (Ver Imagen 6):

```
pearson_corr = dataset.corr(method='pearson')
pearson_corr
```

	Presion 1	Presion 2
Presion 1	1.000000	0.999862
Presion 2	0.999862	1.000000

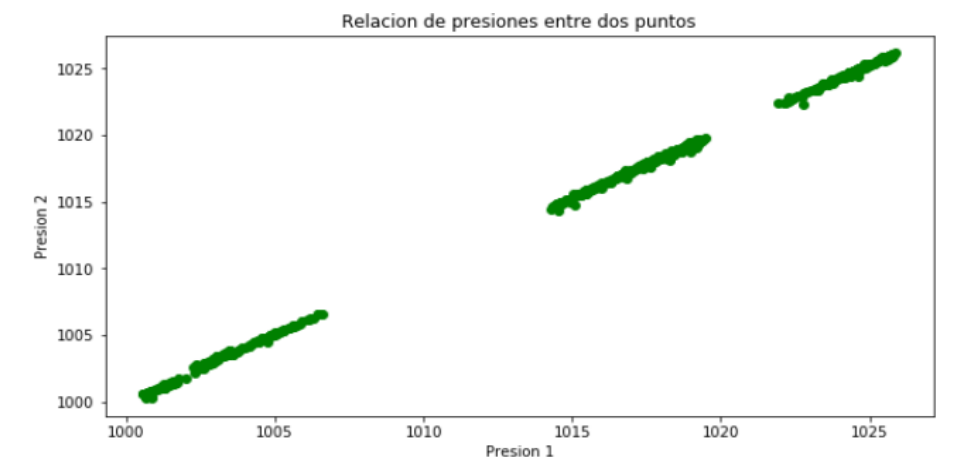
*Imagen 6 – Correlación entre variables Presión 1 y Presión 2*

De esta manera, verificamos que la correlación representa un 99.99% esto quiere decir que nuestro modelo de datos sigue una distribución lineal.

Cómo se comentó anteriormente, se validó el comportamiento de datos analizándolo con las funciones provistas por Python, ya sea viendo la correlación de las variables o graficando la relaciones de todas las variables del modelo con la función “pairplot”. Pero también, se puede observar el

comportamiento del modelo simplemente haciendo un plot del modelo de datos real, para ellos utilizamos la libreria Matplotlib de Python y graficamos el modelo de datos de P1 en función de P2 (Gráfico 2). De esta manera nuestro código para esto es el siguiente:

```
plt.figure(figsize = (10,5))
plt.scatter(X, y, color = 'g')
plt.title('Relacion de presiones entre dos puntos')
plt.xlabel('Presion 1')
plt.ylabel('Presion 2')
plt.show()
```



*Gráfico 2 - P1 en función de P2.*

Cómo puede observarse, el comportamiento que representa el modelo de datos es lineal, y por este motivo es posible predecir los valores de P2 a partir de las presiones en el punto 1, para ello es necesario construir un modelo de regresión lineal que mediante un entrenamiento, se ajuste y minimice el error para poder predecir correctamente los valores de las presiones en el punto 2.



### 3.1.2-Implementación del modelo de Regresión Lineal.

Para la implementación del modelo de regresión lineal se utilizó la librería de Python denominada “Scikit Learn”, ver punto 2.4 del presente documento.

Como se comentó en el punto anterior, las dos variables que intervienen en el modelo de datos son las presiones en el punto 1 y las presiones en el punto 2. Por tal motivo, se decidió representar estos dataset en dos variables, denominadas “X” e “y” y se procedió a reformar el dataset sin modificar el contenido para poder ser procesado por el modelo de regresión lineal que se verá más adelante. De esta manera, definimos dichas variables:

```
X = dataset["Presion 1"]
y = dataset["Presion 2"]

X = X.values.reshape(-1,1)
y = y.values.reshape(-1,1)
```

Luego le indicamos que el 30% de los datos serán para la prueba y el resto permanece en el conjunto de entrenamiento y se define el algoritmo a utilizar en este caso el modelo de regresión Lineal “linear\_model.LinearRegression”. Esto lo realizamos mediante...:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
|
#Defino el algoritmo a utilizar
lr = linear_model.LinearRegression()
```

Por último, se ajusta el algoritmo con los datos de entrenamiento “X train” e “y train” y se realiza la predicción de los datos de prueba “X test”:

```
#Se ajusta al algoritmo
lr.fit(X_train, y_train)

#Realizo una predicción con los datos de prueba
Y_pred = lr.predict(X_test)
```

### 3.1.3-Resultados.

A continuación, graficamos los resultados obtenidos en la predicción “Y\_pred”, y los valores de pruebas para comparar si el modelo definido en el punto anterior se ajusta al modelo de datos reales (Gráfico 3):

```
plt.figure(figsize = (10,5))
plt.scatter(X_test, y_test, color='green', label="Datos del Modelo")
plt.plot(X_test, Y_pred, color='red', linewidth=3, label = "Predicción del modelo" )
plt.title('Regresión Lineal Simple')
plt.xlabel('Presiones punto 1')
plt.ylabel('Presiones punto 2')
plt.legend(loc="upper left")
plt.show()
```

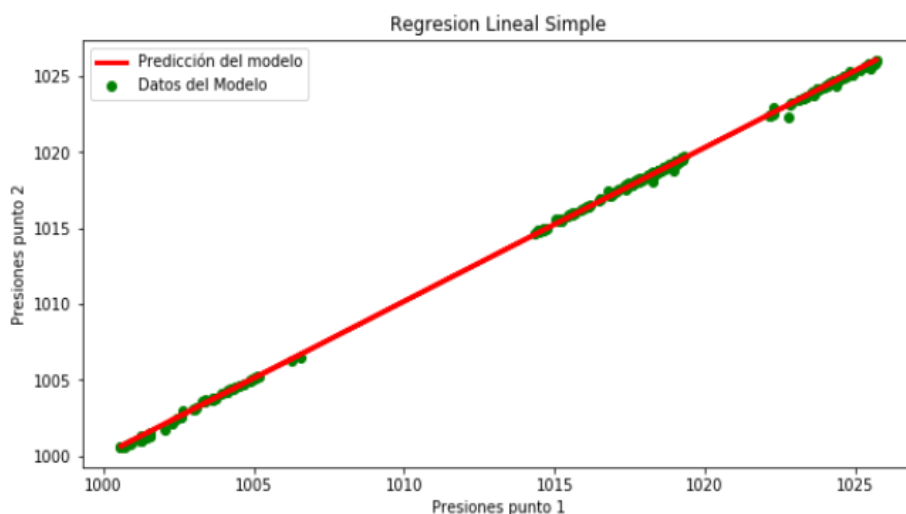


Gráfico 3 - Y\_pred y los valores de pruebas

Cómo puede observarse, la recta roja representa los valores predichos y los puntos verdes los valores del modelo, en este caso está considerando casi la totalidad de puntos del modelo, dando una muy buena predicción. En ese sentido, se evaluó el error cuadrático medio del modelo para verificar cuanto es la diferencia entre los puntos reales y los valores predichos. Ver el detalle a continuación:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mse = mean_squared_error(y_test, Y_pred)
print("Error Cuadrático medio (mse): ", mse)
rmse = np.sqrt(mse)
|
```

```
Error Cuadrático medio (mse): 0.015038281404435052
```

El mse indica que tan cerca están los puntos de datos observados de los valores predichos del modelo. Como puede verificarse el error cuadrático medio es del 0.015 casi nulo, indicando que la diferencia entre ambos puntos es mínima y estableciendo un buen ajuste, dando soporte a las conclusiones arribadas anteriormente sobre el comportamiento lineal del modelo.

## 3.2-Implementación en Código: Caso B.

### 3.2.1-Lectura y análisis del Dataset.

Al igual que en el punto 3.1.1, para la lectura y análisis del dataset se utilizara Pandas. Lo único que difiere que esta vez en vez de leer un archivo CSV, leeremos una URL, que contiene una tabla con formato CSV. Al proveer nuestro dataset de una fuente externa de datos nos aseguramos que lo que corramos tenga **datos actuales** (ya que continuamente van actualizando este Repositorio).

Esta lectura de datasets la vemos a continuación:

```
#asignamos la URL que contiene el CSV a una variable
url_recovered = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/recovered_location.csv"
url_confirmed = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/confirmed_location.csv"
url_deaths="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/deaths_location.csv"

#creamos nuestro dataframe a partir de la URL
dataset_recovered = pd.read_csv(url_recovered)
dataset_confirmed = pd.read_csv(url_confirmed)
dataset_deaths = pd.read_csv(url_deaths)
```

Otra diferencia es que, para este caso, solo nos interesa un país específico, por lo tanto, en el siguiente bloque filtraremos nuestro país deseado en el dataframe. Por ejemplo, como queremos únicamente información de Argentina, nuestra variable “pais” tendrá asignado el string “Argentina”:

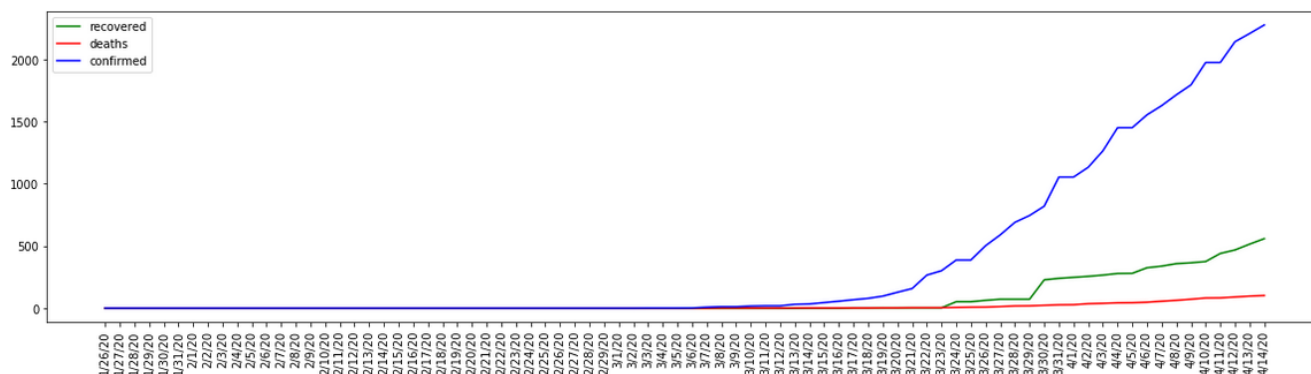
```
#filtramos en nuestro dataframe segun un pais deseado, que estara contenido en la variable "pais"
df_recovered = dataset_recovered[dataset_recovered['Country/Region']==pais]
df_confirmed = dataset_confirmed[dataset_confirmed['Country/Region']==pais]
df_deaths = dataset_deaths[dataset_deaths['Country/Region']==pais]
```

Por último, haremos que en nuestro dataframe solo se encuentran las columnas que contienen fechas:

```
#filtramos en nuestro dataframe unicamente las columnas en el slot 4 en adelante
df_recovered=df_recovered[df_recovered.columns[4:]]
df_deaths=df_deaths[df_deaths.columns[4:]]
df_confirmed=df_confirmed[df_confirmed.columns[4:]]
```

### 3.2.2.-Resultados.

Como se comentó anteriormente, podremos observar ciertos datos (casos confirmados del COVID-19, muertes y recuperaciones) a través del tiempo con una frecuencia diaria para cada país. Primero, analizamos el caso solo para Argentina, obteniendo la visualización del *Gráfico 4*.



*Gráfico 4 – Argentina (casos confirmados, recuperados y muertes)*

De esta manera, observamos que pudimos encontrar un patrón de crecimiento en los contagios (y también para el caso de muertes y recuperaciones aunque en una escala mucho menor). Se podría decir que el conjunto de datos de prueba predichos para los casos confirmados (Gráfico representado en color azul), sigue un crecimiento exponencial, mientras que los demás casos es decir las muertes y los recuperados la curva recién empezó a crecer a partir del 24 de Marzo de 2020, creciendo una escala muy debajo respecto con los demás países comparados, ya que se obtuvo una buena organización y se tomaron medidas rápidamente para minimizar el impacto del COVID-19 en Argentina, teniendo como referencia las medidas tomadas en otros países. Esta visualización de datos la realizamos mediante la función “miFuncion(país)” (Ver *Algoritmo 1*).

```

def miFuncion(pais):
    url_recovered = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/
    url_confirmed = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/
    url_deaths="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/tim
    dataset_recovered = pd.read_csv(url_recovered)
    dataset_confirmed = pd.read_csv(url_confirmed)
    dataset_deaths = pd.read_csv(url_deaths)

    df_recovered = dataset_recovered[dataset_recovered['Country/Region']==pais]
    df_confirmed = dataset_confirmed[dataset_confirmed['Country/Region']==pais]
    df_deaths = dataset_deaths[dataset_deaths['Country/Region']==pais]

    df_recovered=df_recovered[df_recovered.columns[4:]]
    df_deaths=df_deaths[df_deaths.columns[4:]]
    df_confirmed=df_confirmed[df_confirmed.columns[4:]]

    df_recovered=df_recovered[df_recovered.columns[4:]]
    df_deaths=df_deaths[df_deaths.columns[4:]]
    df_confirmed=df_confirmed[df_confirmed.columns[4:]]

    dic_recovered=df_recovered.to_dict(orient='records')[0]
    dic_deaths=df_deaths.to_dict(orient='records')[0]
    dic_confirmed=df_confirmed.to_dict(orient='records')[0]

    plt.figure(figsize=(20,5))
    plt.plot(*zip(*(dic_recovered.items()))),color='g')
    plt.plot(*zip(*(dic_deaths.items()))),color='r')
    plt.plot(*zip(*(dic_confirmed.items()))),color='b')
    plt.xticks(rotation=90)

    plt.show()

```

*Algoritmo 1 – Función “miFuncion(pais)”*

Cómo se explicó en el punto anterior, una vez obtenida la información de los Datasets y filtrado el conjunto de datos por el país a analizar, se realiza un gráfico (utilizando la librería Matplotlib) del modelo de datos representando para el país requerido, el cual contendrá 3 gráficos, uno para la cantidad de casos de muertes, otro para la cantidad de recuperados y finalmente un gráfico para la cantidad de infectados, todos en función del tiempo diario.

Luego de esto, realizamos una comparación con ciertos países. Nuestro criterio de selección se basó en elegir los 3 países con mayor cantidad de casos confirmados. Esta selección la realizamos gracias a la función “miTop(numero)” (Ver *Algoritmo 2*) la cual nos devuelve dichos países; y realizamos el ploteo de los gráficos para cada uno de los países con la función “mifuncion(pais)” ya anteriormente explicada.

```
def miTop(numero):
    url_recovered = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data_recovered.csv"
    url_confirmed = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data_confirmed.csv"
    url_deaths = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data_deaths.csv"
    dataset_recovered = pd.read_csv(url_recovered)
    dataset_confirmed = pd.read_csv(url_confirmed)
    dataset_deaths = pd.read_csv(url_deaths)

    df_recovered = dataset_recovered
    df_recovered["Suma"] = dataset_recovered[dataset_recovered.columns[-1]].sum(axis=1)

    df_confirmed = dataset_confirmed
    df_confirmed["Suma"] = dataset_confirmed[dataset_confirmed.columns[-1]].sum(axis=1)

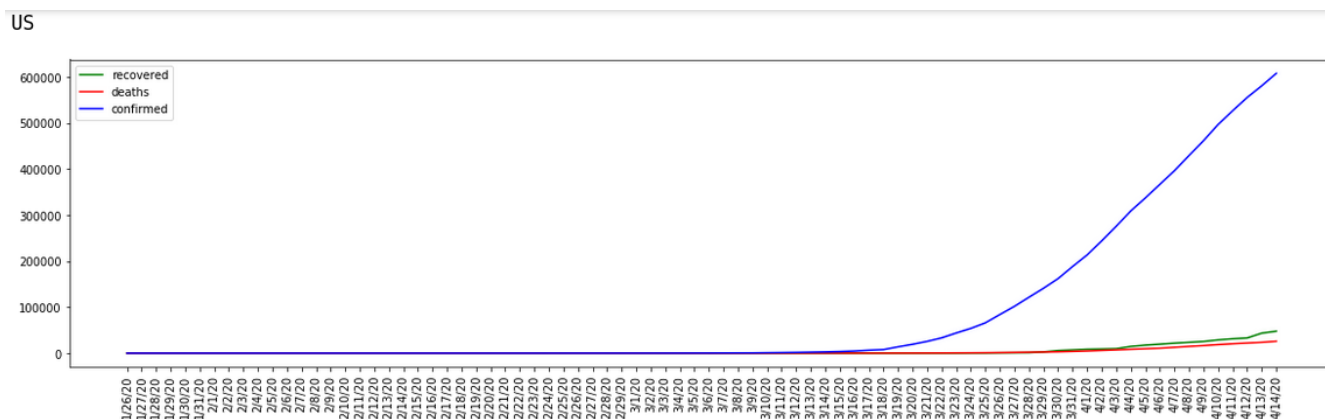
    df_deaths = dataset_deaths
    df_deaths["Suma"] = dataset_deaths[dataset_deaths.columns[-1]].sum(axis=1)

    Top3Recovered = df_recovered.sort_values(by='Suma', ascending=False)[["Country/Region", "Suma"]].head(numero)
    Top3Confirmed = df_confirmed.sort_values(by='Suma', ascending=False)[["Country/Region", "Suma"]].head(numero)
    Top3Deaths = df_deaths.sort_values(by='Suma', ascending=False)[["Country/Region", "Suma"]].head(numero)
    #print("prueba")
    #print(Top3Confirmed)
    list = Top3Confirmed['Country/Region'].tolist()
    #print(list)
    return list
```

*Algoritmo 2 – Función “miTop(numero)”*

En la primera parte del *Algoritmo 2* (función “mi Top(numero)”) leemos los 3 CSVs de la Web (de Github) y almacenamos los datos en 3 data sets / variables distintas. Luego, sumamos la cantidad de casos diarios consolidados en la última columna del dataset (por esto son los “-1”). Y por último, ordenamos los datos obtenidos y nos quedamos con el número de casos que nos interesa analizar. En nuestro modelo elegimos el top 3 de los países con mayor cantidad de casos de cada categoría.

De esta manera, **obtenemos el plot de los 3 países con mayor cantidad de casos confirmados** (Ver los *Gráficos 5, 6 y 7*).



*Gráfico 5– Estados Unidos (casos confirmados, recuperados y muertes)*

## Spain

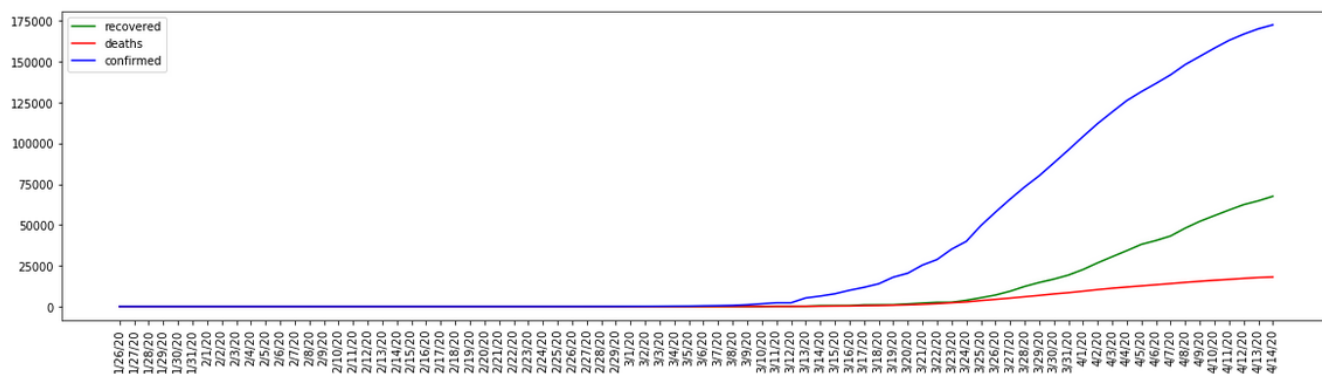


Gráfico 6– España (casos confirmados, recuperados y muertes)

## Italy

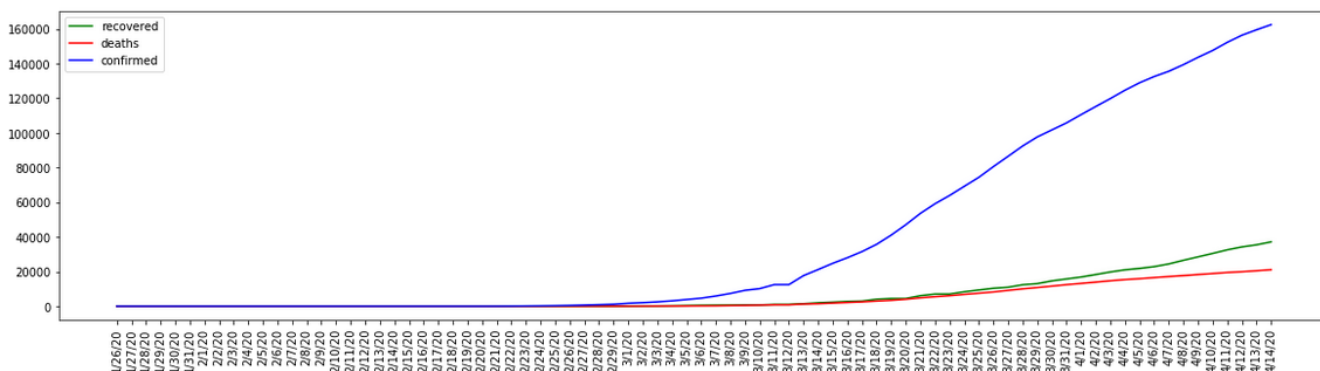


Gráfico 7– Italia (casos confirmados, recuperados y muertes)

De las visualizaciones realizadas, podemos notar que Argentina está muy bien posicionada, es decir que muestra muy buen comportamiento inicial y un crecimiento paulatino y lento comparado con los demás países.

#### 4-Conclusiones.

Este Trabajo de Laboratorio nos deja una muy buena idea respecto a cómo aplicar la técnica de regresión lineal a un modelo de datos y por qué es tan importante realizar un análisis y exploración de datos previa. De esta manera pudimos poner en práctica conceptos de estadística y así lograr el manejo y visualización de datos provenientes de distintas fuentes (obteniendo así distintos Data Sets). Todo este manejo y visualización de datos nos ayudó a adquirir conocimientos en la utilización de muchas librerías de Python y los entornos Anaconda y Jupyter Notebook. El módulo que más aprendimos a utilizar fue Pandas, una de las librerías más versátiles de Python que nos facilitó muchísimo la manipulación y visualización de los datasets: sobre todo para el **Caso B**.

Aplicar la técnica de Regresión Lineal para el **Caso A** nos resultó bastante útil. Tanto su forma de única variable como su forma multivariable. Siempre y cuando cumplamos el requisito de linealidad, podemos utilizarla para predecir outputs.



## 5-Mejoras a desarrollar.

- En el **Caso B** únicamente hicimos una exploración y visualización de datos. NO realizamos ninguna predicción. Como mejora podríamos analizar si existe una relación lineal entre las variables del punto 2 (COVID-19) para así poder **aplicar Regresión Lineal y PREDECIR** algo (por ejemplo el número de muertes en función del tiempo).
- Para el **Caso A** (donde ya tenemos implementada la predicción) podemos aplicar Regresión Lineal con el método de **descenso por el gradiente** (enés de utilizar el método de mínimos cuadrados) para predecir así nuestros modelos de datos; ya que resulta **más eficiente**. Luego podríamos implementar lo mismo para el **Caso B**.

## 6-Bibliografía.

- Apunte de la materia: [https://gitlab.com/Inteligencia\\_Artificial\\_UNDAV/teor-a/-/blob/master/Inteligencia%20Artificial.pdf](https://gitlab.com/Inteligencia_Artificial_UNDAV/teor-a/-/blob/master/Inteligencia%20Artificial.pdf)
- <https://medium.com/@calaca89/entendiendo-la-regresi%C3%B3n-lineal-con-python-ed254c14c20>
- <https://iartificial.net/regresion-lineal-con-ejemplos-en-python/>
- <https://numpy.org/>
- <https://matplotlib.org/>
- <https://pandas.pydata.org/>
- <https://scikit-learn.org/stable/>