Ajustes polinomiales orientados a casos COVID-19 Colombia 06 Marzo a 14 Septiembre 2020

JULIAN GIRALDO CARDONA

JUAN DAVID ÁLVAREZ

Toma de los datos

La página web esri nos proporcionó los datos correspondientes para así poder desarrollar los códigos que aplicamos en Python gracias a una serie de librerías muy avanzadas para el manejo estadístico y matemático.

Los datos específicamente fueron de los casos confirmados durante 192 días en Colombia, en ésa página web los datos están representados por unos gráficos de barras como se puede observar a continuación

Reporte

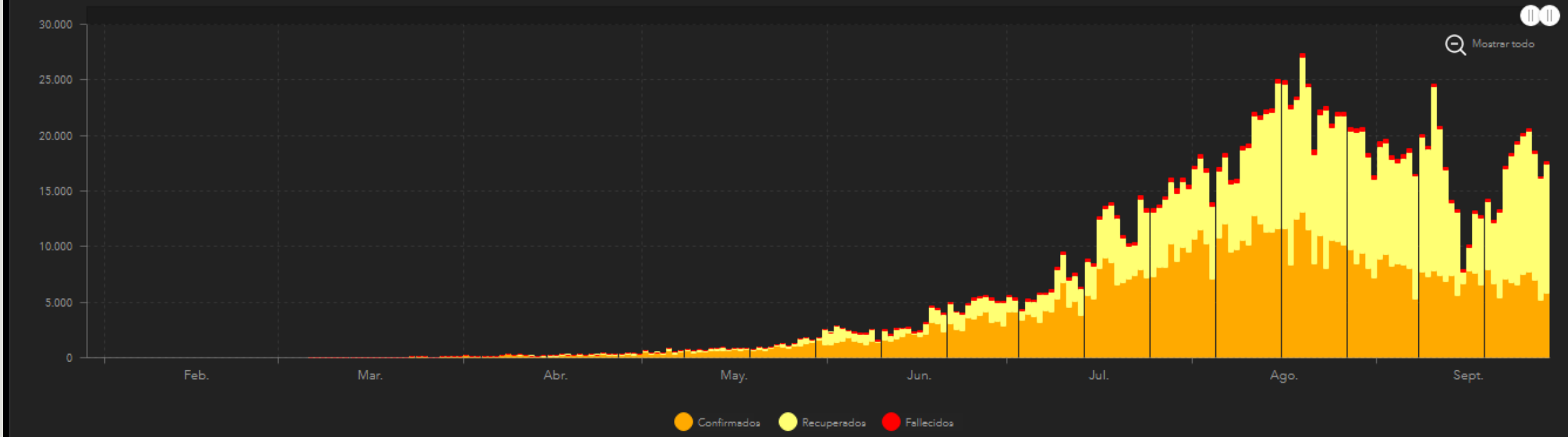
A Story Map

COVID-19 Colombia | COVID-19 Colombia (procedencia) | COVID-19 Mundo

Coronavirus COVID-19 en Colombia

Tablero de control realizado por Esri Colombia

Reporte diario de casos nuevos



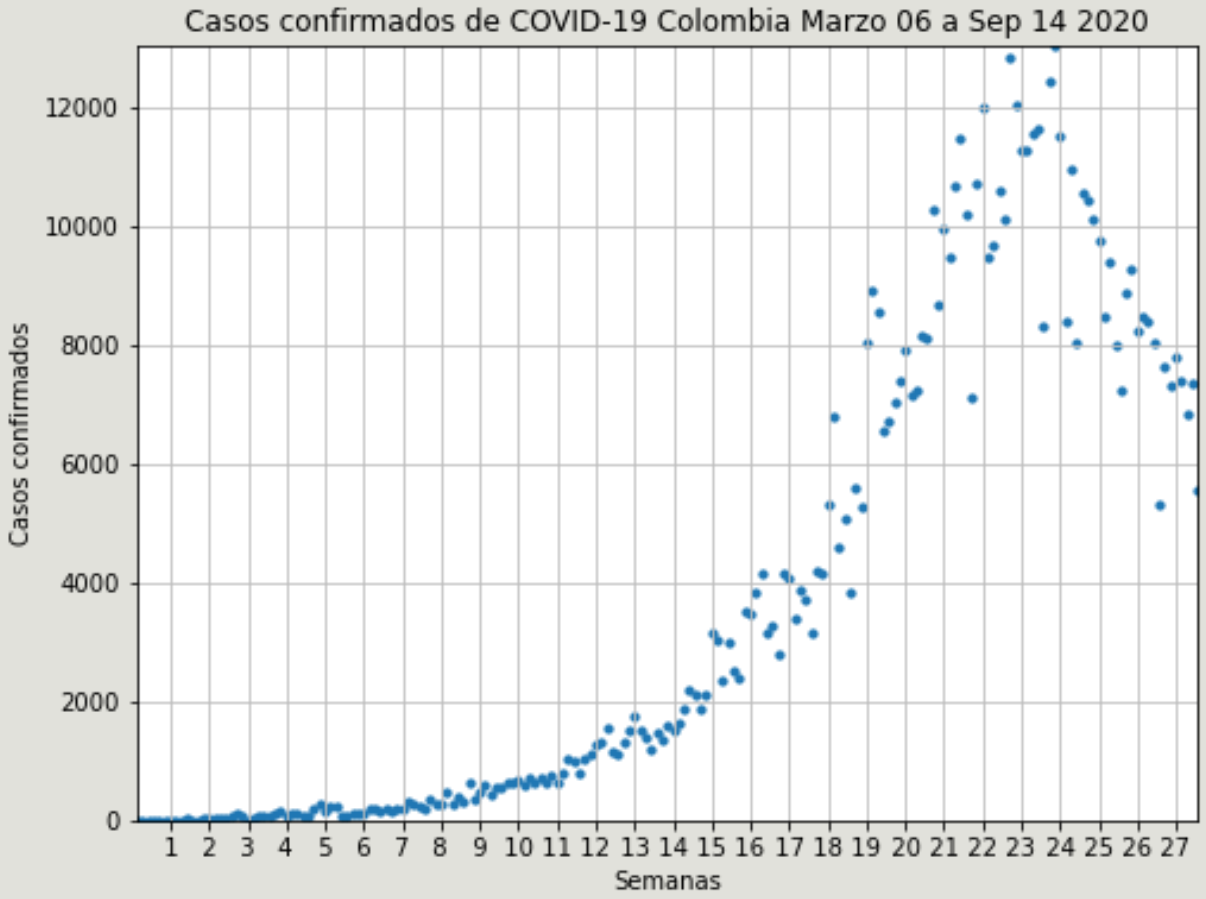
Casos nuevos | Género y Edad

Organización de los datos en archivo tsv

DIAS	CASOS CONFIRMADOS
1	1
2	0
3	0
4	2
5	0
6	6
7	4
8	3
9	8
10	21
11	12
12	18
13	27
14	26
…	…
189	7813
190	7424
191	6826
192	7355
193	5573

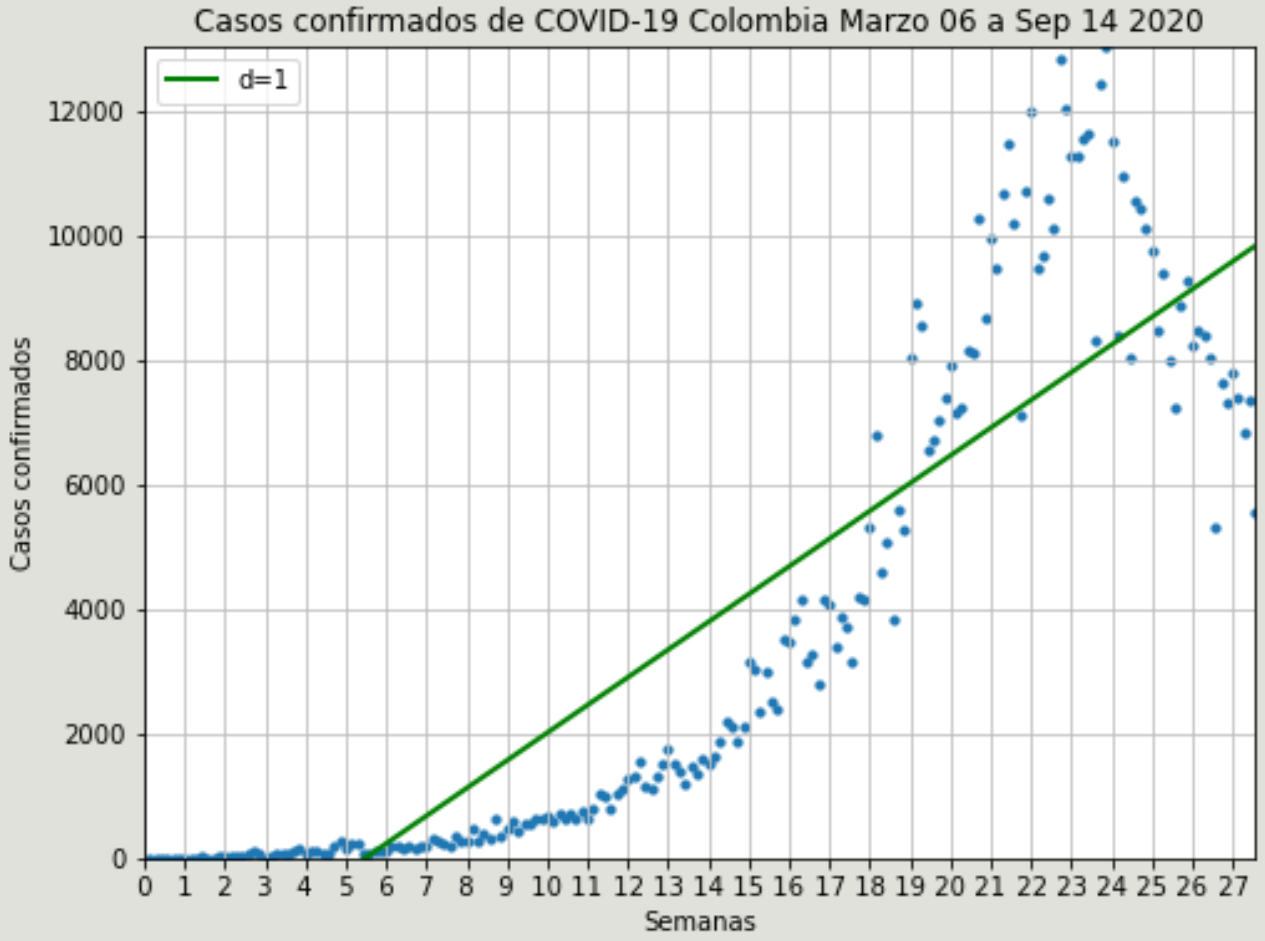
Los datos en total fueron 193, divididos en 2 columnas.
La primer columna indica el día y la segunda los casos confirmados.
En este caso no hubieron datos de tipo “nan” pero de todos modos se hizo el respectivo procesamiento en Python.
Se organizan los datos en dos variables separadas para poder crear “x” y “y”, y de esta forma podemos graficar y establecer los ajustes polinomiales

Se realiza la primer gráfica.



Establecimos el titulo del grafico, el nombre en el eje y, y el nombre en el eje x. Los puntos muestran una tendencia en crecimiento pero en los últimos días parece que se están disminuyendo los datos, por esta razón es importante aplicar la regresión lineal para realizar predicciones.
Registramos 27 semanas en total, donde el limite máximo es 12000 casos confirmados.

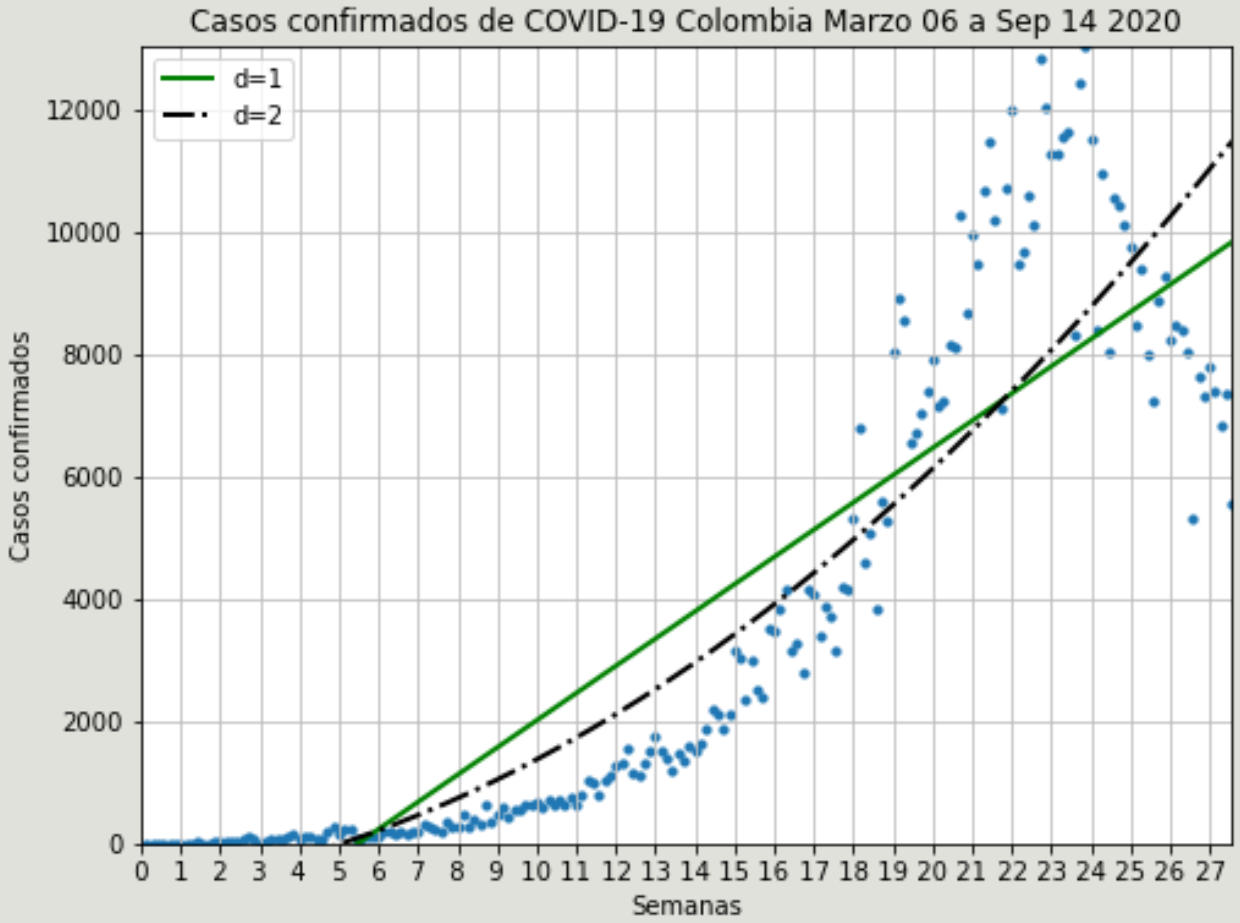
Análisis de la segunda gráfica



En esta grafica trazamos una recta con los valores de “x” y “y” mediante una instrucción que nos provee numpy que permite el ajuste polinomial de mínimos cuadrados y allí se indica el grado de polinomio que en este caso es 1.

Como se puede observar en la gráfica, la recta que trazamos acumula mucho error ya que no pasa por la gran mayoría de los puntos en la grafica.
El error en este modelo es de 6.71773991e+08

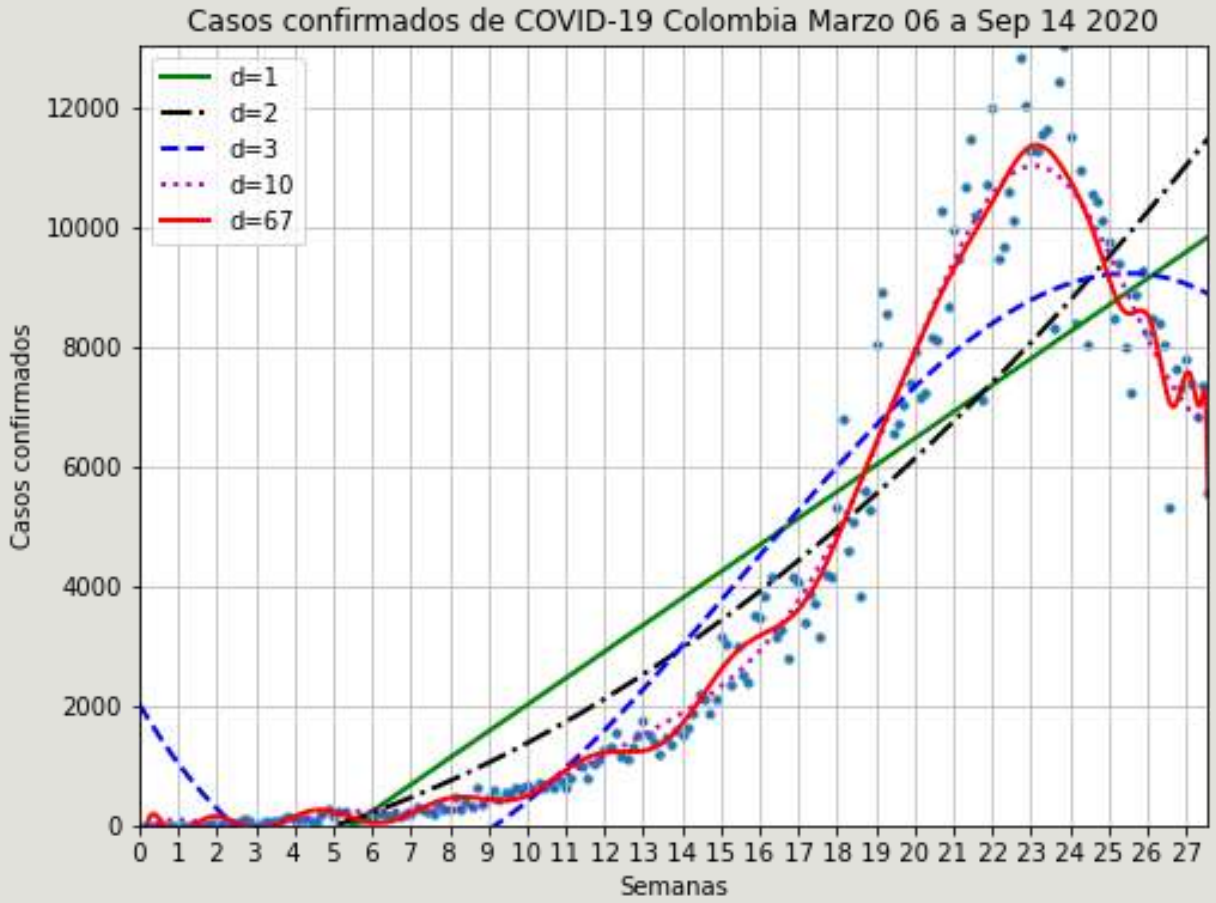
Análisis de la tercer gráfica con dos rectas de diferente orden polinomial



Como la primer recta que utilizamos acumula demasiado error, entonces procedemos a crear una nueva recta, pero en este caso es de orden polinomial 2. Se observa que dicha recta pasa por más puntos a comparación de la recta de orden 1.

El error del modelo de dimensión 2 es de 5.63769899e+08

Cuarta gráfica, creación de más rectas con un orden polinomial mayor.



Se agregaron nuevas rectas con un orden polinomial mayor y la diferencia se nota a gran escala ya que, como se puede observar en la figura. La función de orden 67 se acopla casi perfectamente a los diferentes puntos que se encuentran distribuidos en la gráfica.

Error para el conjunto de datos:

Error d=1: 671773991.227258

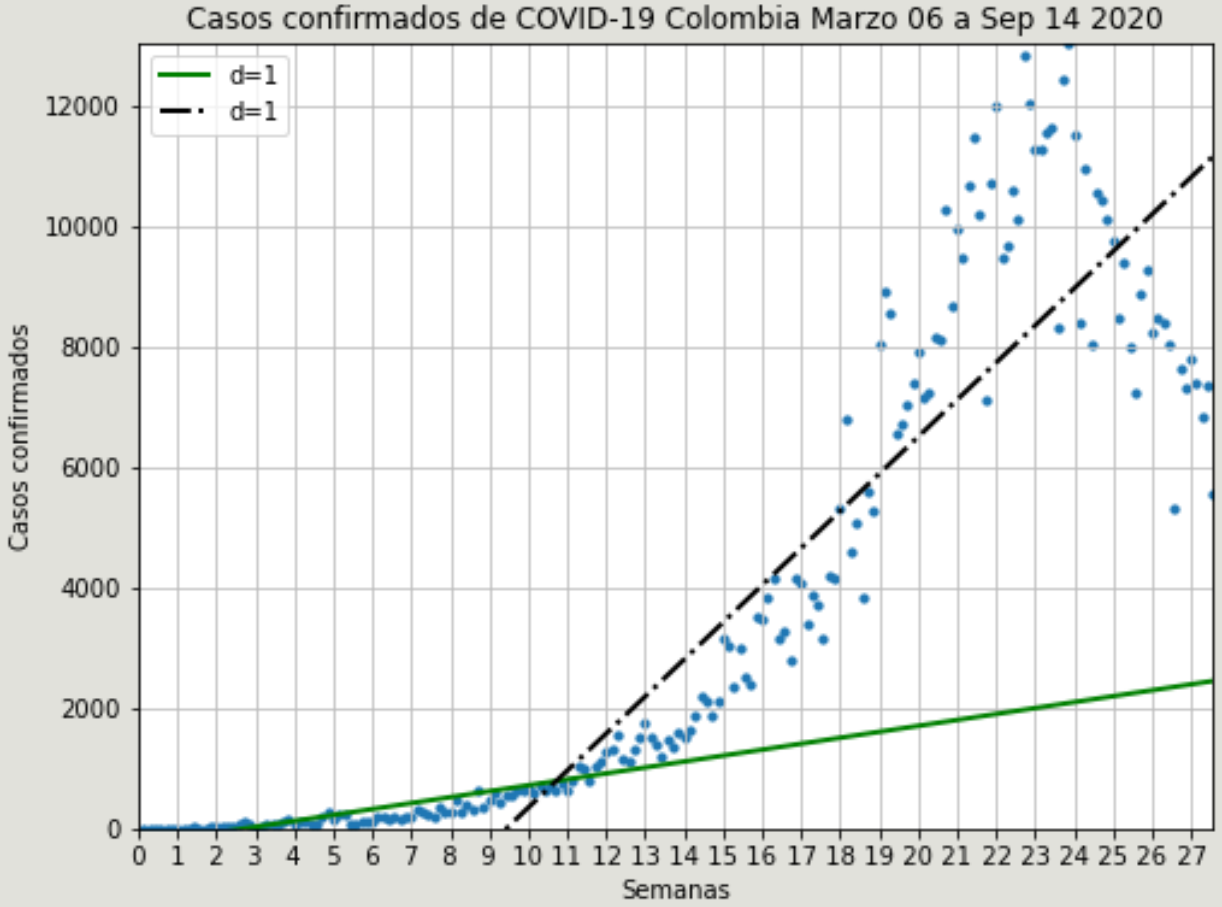
Error d=2: 563769899.476688

Error d=3: 365392650.863871

Error d=10: 97438288.097343

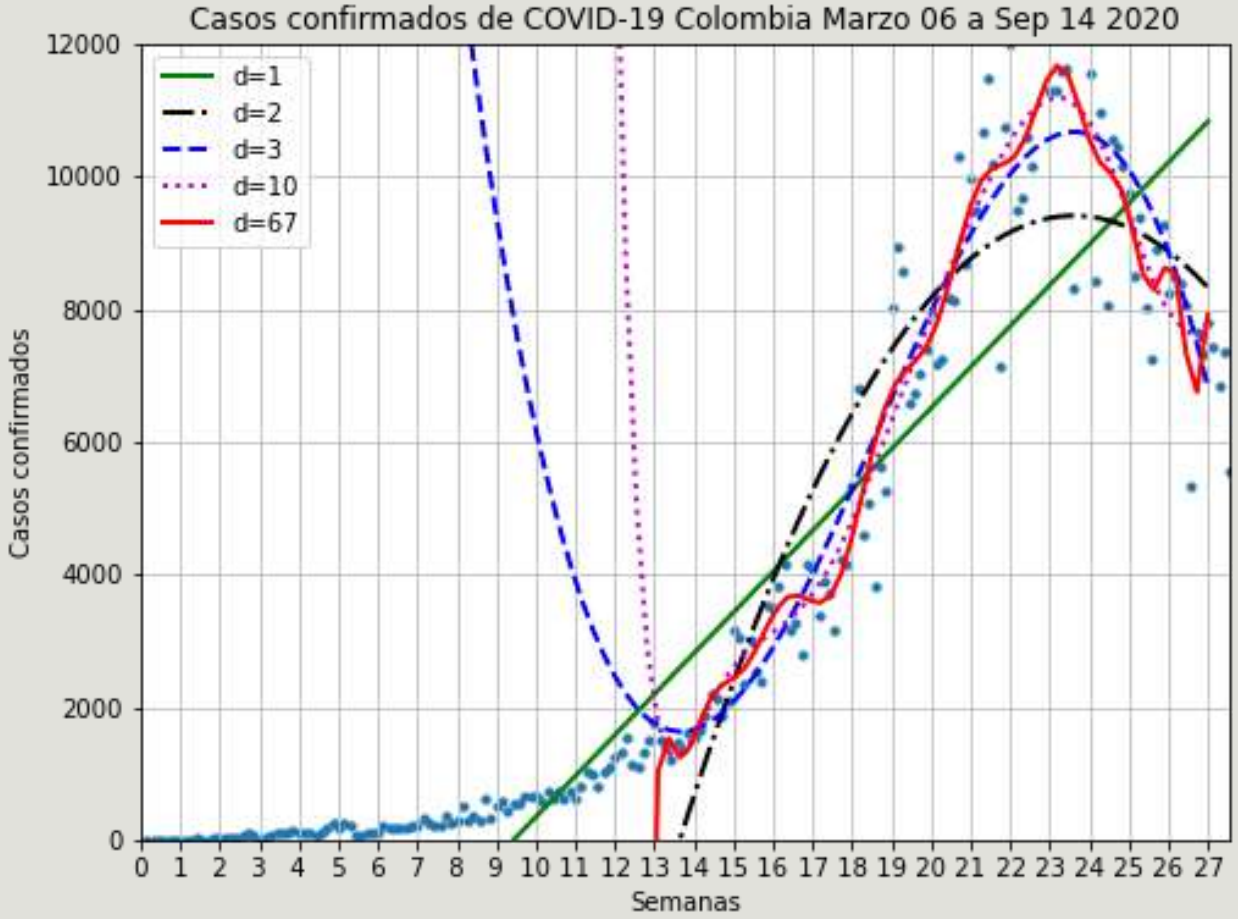
Error d=67: 89672810.096973

Establecemos punto de inflexión



Establecemos un punto de inflexión en la semana 10.5 para sí separar la grafica en dos partes, la primera, donde tenemos los puntos antes de que tengan esa curvatura y la segunda parte donde ya tienen la curvatura, se puede observar que las dos funciones cambiaron a comparación de las graficas mostradas anteriormente.

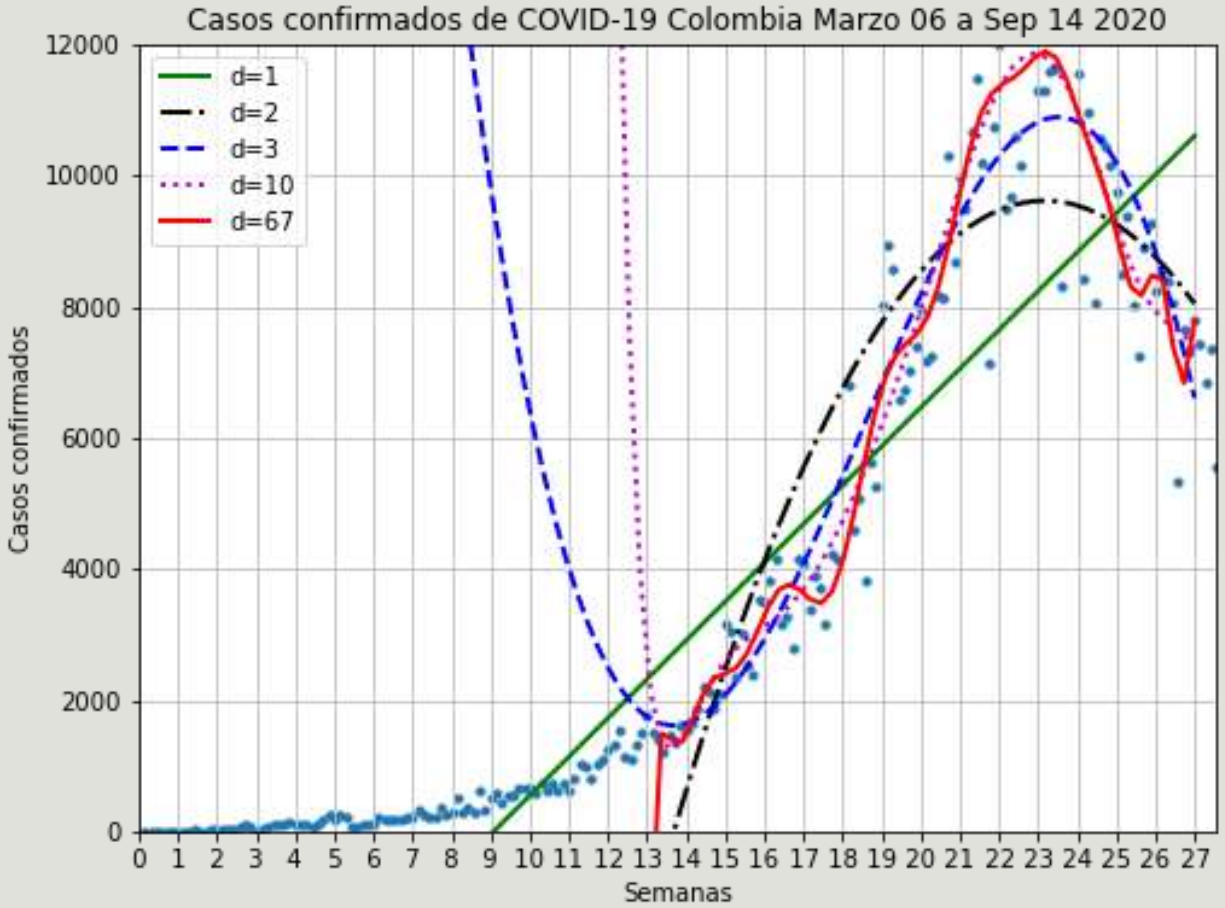
Se entrena el sistema para acomodar mejor las funciones.



Luego de establecer el punto de inflexión, se entrenan los datos en nuevas variables. El entrenamiento se hace gracias a unas librerías de Numpy y Matplot. Se puede observar que todas las funciones ahora están más acorde a los puntos que hay en la gráfica.

Errores solamente después del punto de inflexión:
Error d=1: 513115894.568864
Error d=2: 529894835.423422
Error d=3: 312533065.239517
Error d=10: 96475465.133951
Error d=67: 87983590.298546
Error de inflexión=463130100.470557

Se crean unas variables súper entrenadas



Mediante unas instrucciones que nos provee Numpy, una llamada shuffled podemos obtener una serie de datos y barajarlos para así conseguir una completa aleatoriedad.

Prueba de error para después del punto de inflexión:
Error d=1: 122120722.379765
Error d=2: 57442612.226852
Error d=3: 29145628.537626
Error d=10: 35921049.912863
Error d=67: 35082768.733571

Realizamos 2 tipos de predicción

La primer predicción que realizamos es utilizando la función fbt2 y de esta forma podremos utilizar la instrucción fsolve que es para resolver una función, le restamos 100000 para reducir la escala y determinar en que momento ese valor en “y” hará una intersección en el eje 0 de “y”, y así determinar el valor del eje “x”.

Esta predicción nos arroja como resultado que en la semana 50 se van a confirmar 100.000 casos positivos de COVID-19.

Predicción utilizando sklearn y linear regression

Mediante esta instrucción de linear regression podemos saber el valor que tendrá la variable y respecto a la variable x. primero tenemos que entrenar la variable con model.fit(x,y) y de esta forma ya se puede hacer la predicción que uno quiera. En este caso queremos averiguar la predicción de los casos acumulados en los días de predicción 208 que restándolos con el total e los días registrados tenemos que son 15 días los que se van a predecir. El resultado es:

Se esperan 877181 casos confirmados en 208 días.

Esta predicción se acerca bastante a la realidad pero ocurre un problema cuando se agrega demasiados días para predecir porque lo que hace es aumentar y aumentar el número de casos confirmados y sabemos que esto del virus no durará para siempre.