Desarrollo de un modelo de Regresión para predecir el número de casos confirmados de personas con COVID-19

Autor: Bryam David Vega Moreno
Maestro: Diego Quisi
Materia: Simulación
Universidad: Universidad Politécnica Salesiana
Carrera: Ciencias de la computación

Introducción

Actualmente, el covid se ha vuelto uno de los virus más trascendentales a nivel mundial, perjudicando económicamente a los paises, ademas de ello afecto en gran medida a los sistemas de salud que no estaban preparados para un virus como este. En nuestro específico, en Ecuador se ha desatado una creciente ola de contagios desde que inicio el virus, en donde hospitales estan al máximo de su capacidad y la mayoria de ellos no pueden atender a más personas. En este ocasión proponemos un modelo de regresión que nos permite predecir el número de personas que existiran en un determiado día, con la finalidad de poder tener un número cercano para estar preparados para lo que pueda pasar.

Desarrollo del modelo de regresión

Librerias a importar

Librerias para la lectura y el análisis de datos

```
In [2]: import pandas as pd
        import numpy as np
```

Librerías para realizar procesos de transformación y división de datos

```
In [3]: from sklearn.model_selection import train_test_split
```

Librerias para realizar el proceso de regresión

```
In [4]: from sklearn.linear_model import LinearRegression
        from sklearn.pipeline import Pipeline
        from sklearn.preprocessing import PolynomialFeatures
```

Librerias para realizar las gráficas

```
In [5]: import altair as alt
        import matplotlib.pyplot as plt
```

Dataset

Para este análisis hemos preparado un dataset denominado **covid.csv** el cual es un dataset que tuvo un proceso de transformación mediante otros csv y se concentro la información para tener los datos de nuestro país que en este caso es Ecuador. El dataset contiene la siguiente información:

- **date:** Fecha en la que se obtiene los datos
- **death:** Número de personas fallecidas por el virus
- **confirmed:** Número de personas con el virus
- **recovered:** Número de personas recuperadas con el virus
- **day:** Número del dīa de la pandemia a partir del 1/01/2020

```
In [8]: df = pd.read_csv('./in/covid.csv')
        df.sample(5)
```

Out[8]:

	date	confirmed	deaths	recovered	day
393	2021-03-29	325124	16746	281684	453
314	2021-01-09	220349	14177	190350	374
242	2020-10-29	166302	12622	141759	302
293	2020-12-19	205920	13948	177951	353
121	2020-06-30	56432	4527	27594	181

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 409 entries, 0 to 408
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   date       409 non-null    object
 1   confirmed  409 non-null    int64 
 2   deaths     409 non-null    int64 
 3   recovered  409 non-null    int64 
 4   day        409 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 16.1+ KB
```

Convertir datos para realizar un análisis exploratorio

Un paso muy importante para realizar los modelos es realizar un análisis con el fin de ver como se esta comportando la data, en esta ocasión procedemos a hacer un proceso de transformación rápido para poder trabajar con el tipo de datos correctos

```
In [17]: df_copy = df.copy(deep=True)
         df_copy = df_copy.convert_dtypes()
         df_copy['date'] = pd.to_datetime(df_copy['date'])
```

```
In [18]: df_copy.sample(5)
```

Out[18]:

	date	confirmed	deaths	recovered	day
282	2020-12-08	198752	13794	174188	342
322	2021-01-17	231482	14319	199332	382
299	2020-12-25	208828	13984	181618	359
53	2020-04-23	11183	560	1328	113
311	2021-01-06	217377	14146	190350	371

```
In [19]: df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 409 entries, 0 to 408
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   date       409 non-null    datetime64[ns]
 1   confirmed  409 non-null    Int64         
 2   deaths     409 non-null    Int64         
 3   recovered  409 non-null    Int64         
 4   day        409 non-null    Int64         
dtypes: Int64(4), datetime64[ns](1)
memory usage: 17.7 KB
```

Ya con estos datos transformados, procedemos a realizar un análisis exploratorio para ver como se estan comportando nuestros datos

---

Análisis exploratorio

```
In [29]: df_time = df_copy.set_index(['date'])
         df_time.drop('day',axis=1,inplace=True)
```

Como se comportan los datos en corte de cada mes

```
In [35]: df_time=df_time.resample('M').sum()
```

```
In [40]: df_time
```

Out[40]:

	confirmed	deaths	recovered
date			
2020-03-31	16469	423	84
2020-04-30	321161	12586	23439
2020-05-31	1029314	76863	210913
2020-06-30	1429745	118831	702114
2020-07-31	2217004	160481	978383
2020-08-31	3121790	189640	2521381
2020-09-30	3678159	304684	3008490
2020-10-31	4714534	379203	4009025
2020-11-30	5411330	390757	4740172
2020-12-31	6306084	430061	5504187
2021-01-31	7125483	445664	6083704
2021-02-28	7472445	428747	6333926
2021-03-31	9480656	505563	8151264
2021-04-30	4416015	222210	3756482

Personas que estan en tratamiento

```
In [51]: df_treatment = df_time['confirmed']-df_time['deaths']-df_time['recovered']
```

```
In [52]: df_treatment
```

```
Out[52]: date
2020-03-31      15962
2020-04-30     285136
2020-05-31     741538
2020-06-30     608800
2020-07-31    1078140
2020-08-31     410769
2020-09-30     364985
2020-10-31     326306
2020-11-30     280401
2020-12-31     371836
2021-01-31     596115
2021-02-28     709772
2021-03-31     823829
2021-04-30     437323
Freq: M, dtype: Int64
```

Número de contagios, muetras, recuperaciones se da entre meses

```
In [46]: df_diff = df_time.diff()
```

```
In [49]: df_diff=df_diff.fillna(df_time.head(1).to_dict())
```

```
In [121]: df_diff
```

Out[121]:

	confirmed	deaths	recovered
date			
2020-03-31	16469	423	84
2020-04-30	304692	12163	23355
2020-05-31	708153	64277	187474
2020-06-30	400431	41968	491201
2020-07-31	787259	41650	276269
2020-08-31	904786	29159	1542998
2020-09-30	556369	115044	487109
2020-10-31	1036375	74519	1000535
2020-11-30	696796	11554	731147
2020-12-31	894754	39304	764015
2021-01-31	819399	15603	579517
2021-02-28	346962	-16917	250222
2021-03-31	2008211	76816	1817338
2021-04-30	-5064641	-283353	-4394782

Promedio de contiagios,muertes,recuperados

```
In [58]: df_diff.mean()
```

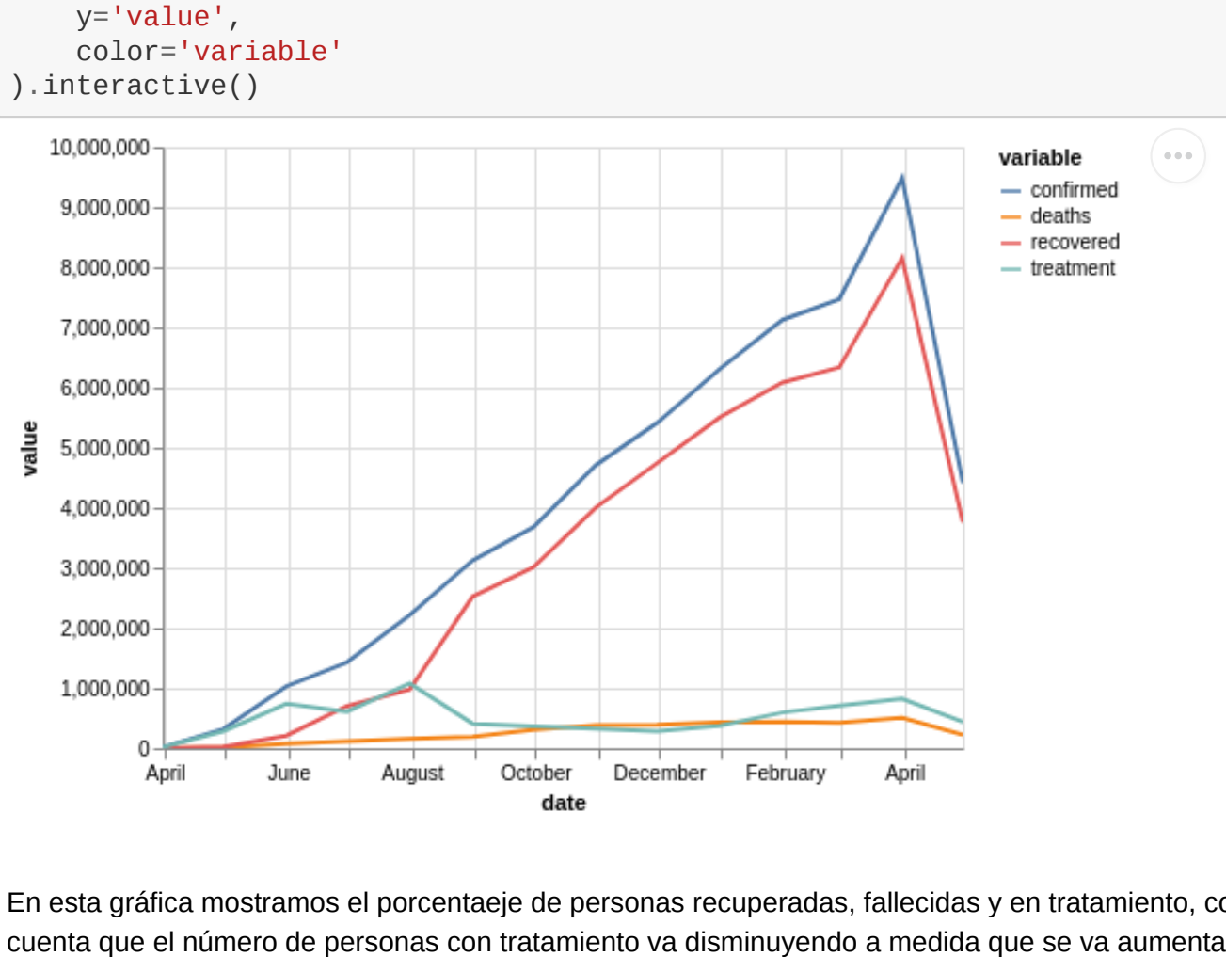
```
Out[58]: confirmed    315429.642857
deaths        15872.142857
recovered    268320.142857
dtype: float64
```

Gráficas

En esta gráfica estamos mostrante el comportamiento de los datos a lo largo del tiempo, como podemos apreciar, el número de confirmados recuperados tiene una tendencia paralela,es decir, a medida que se van confirmando casos, podriamos decir que una gran mayoria de ellas se recupera, mientras que el resto pueda estar en tratamiento o fallecido

```
In [123]: alt.Chart(df_time.drop('rate',axis=1).reset_index().melt('date')).mark_line().encode(
              x='date',
              y='value',
              color='variable'
          ).interactive()
```

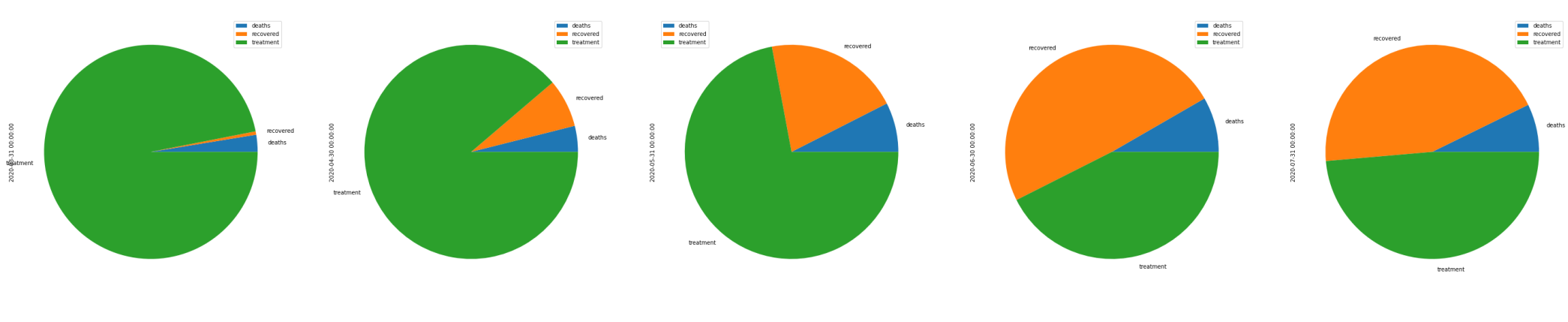
Out[123]:



En esta gráfica mostramos el porcentaeje de personas recuperadas, fallecidas y en tratamiento, como podemos darnos cuente en la gráfica, podemos darnos cuenta que el número de personas con tratamiento va disminuyendo a medida que se va aumentando el número de personas recuperadas lo cual tiene sentido

```
In [102]: df_time['treatment'] = df_time['confirmed']-df_time['deaths']-df_time['recovered']
          df_time[['deaths','recovered','treatment']].T.iloc[:,:5].plot(figsize = (50,10),kind='pie',subplots=True)
```

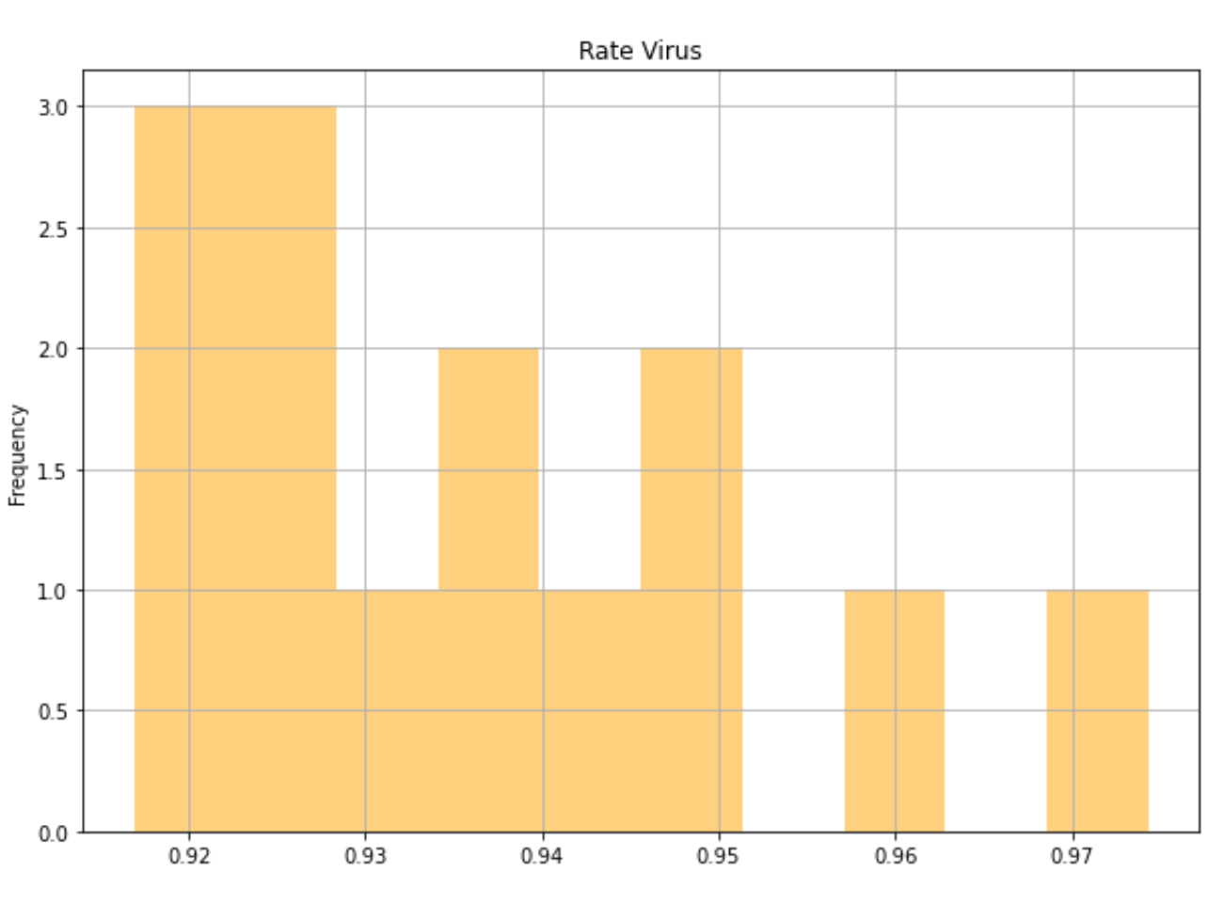
```
Out[102]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x7ffd15fc3cd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7ffd609f0550>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7ffd609cb880>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7ffd5fef7220>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7ffd60d90f40>],
      dtype=object)
```



Un análisis muy bueno seria ver la velocidad con la que el virus se propaga a lo largo del tiempo, esto es importante para ver cual es su frecuencia de velocidad de propagación con el fin de entender como se esta compartando el virus en nuestro país. Como podemos ver en el histograma que presentamos a continuación la velocidad de propagación del virus es alta a medida que avanza el tiempo, lo interesante es darnos cuenta que es un virus muy contagioso por lo que su velocidad de propagación no ha bajado de los 0.90

```
In [111]: df_time['rate'] = 1 - df_time['deaths']/df_time['confirmed']
          df_time['rate'].plot(kind='hist',figsize=(10,7),bins=10,color='orange',alpha=0.5,title='Rate Virus',grid=True)
```

```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffd17f51d00>
```



Una vez analizado los datos, procedemos a realizar la creación del modelo con el fin de poder escoger el mejor modelo para este tipo datos

---

Creación del modelo

Con el análisis de datos realizado, podemos darnos cuenta que nuestros datos tienen una tendencia a la alza, de manera lineal, por lo tanto un modelo de regresion seria ideal para poder estos datos. Lo primero que procedemos a hacer es obtener nuestro features y targets para poder hacer una división de datos en entrenamiento y test para poder entrenar a nuestro modelo y despues testearlo con los datos de test.

Selección de feautres y target

En este caso escogimos como features la columna **days** debido a que dicha columna juega el rol más improtante como hemos visto a lo largo del análisis, si bien en el análisis tratamos la columna date, no podemos utilizarla debido a ser un tipo de dato fecha, por lo que el día equivale a la fecha lo cual lo hace el feature más importate. Por otro lado escogimos la columna **confirmed** como nuestro target debido a que eso es lo que deseamos predecir, el número de contagios en un día a futuro

```
In [129]: features = df.iloc[:,4:5].values
          target = df.iloc[:,1:2].values
```

División de train y test

Una vez que ya tenemos nuestro conjunto de datos de entrada y salida, procedemos a dividir dichos datos en train y test a fin de realizar entreamiento con los datos de train y las pruebas con los datos de test, para ello hacemos uso de la libreria **train_test_split**. A continuación mostramos el código para realiarlo

```
In [131]: X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2)
```

Con estos datos dividos procedemos a realizar los modelos de regresión, a continuación procedemos a realizarlos.

Creación de un modelo de regesión

Como habiamos dicho, nuestros datos tienen una tendencia creciente y lineal, por lo que un modelo de regresión lineal es adecuado para este problema, para ello utilizamos el modelo **LinearRegression** para proceder a hacer nuestro entrenamiento y después las pruebas de nuestro modelo

```
In [134]: model_linear = LinearRegression()
          model_linear = model_linear.fit(X_train,y_train)
```

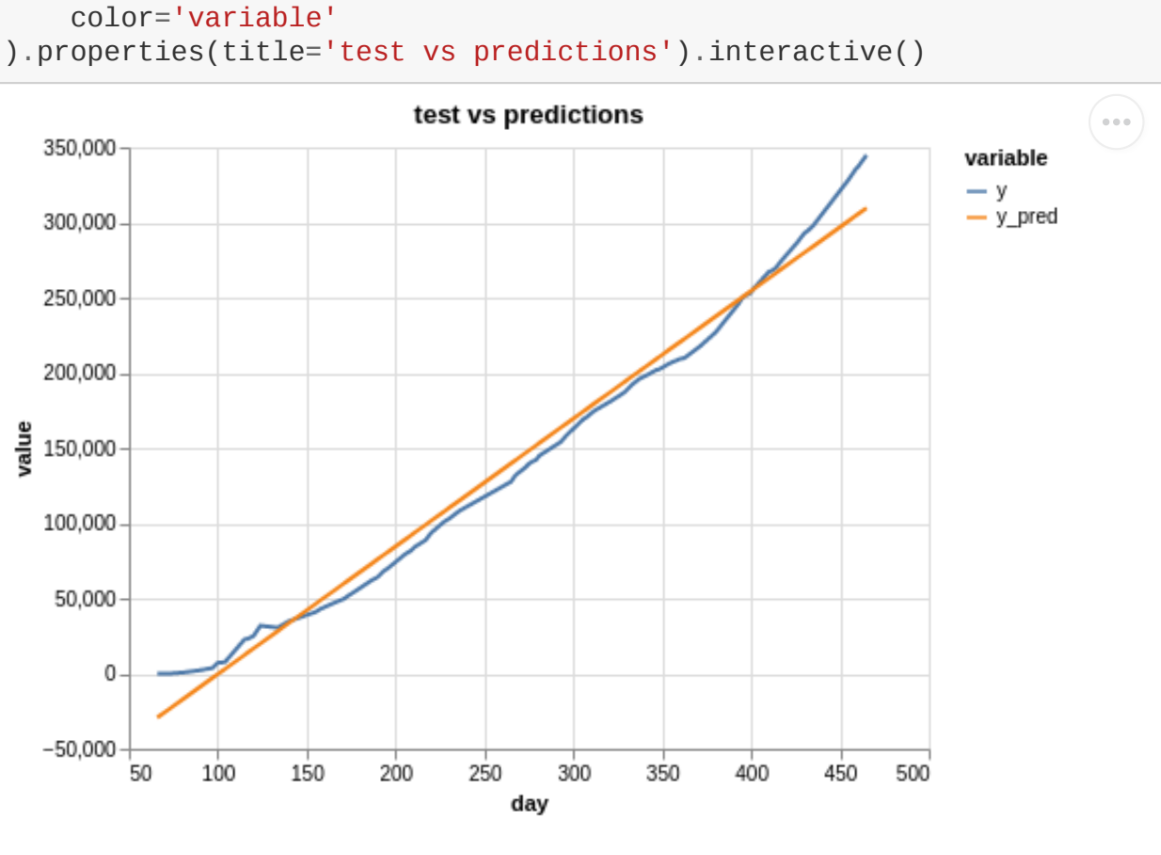
```
In [136]: print("Linear Score : ",model_linear.score(X_test,y_test))
```

```
Linear Score :  0.9857429540365887
```

Como podemos apreciar nuestro modelo lineal tiene una score de 0.98 lo cual es alto. Pero, seria mejor ver una gráfica de la misma con el fin de ver si nuestro modelo no tiene problema de overfitting

```
In [144]: data_linear=pd.DataFrame({'day':X_test.reshape(-1),'y':y_test.reshape(-1),'y_pred':model_linear.predict(X_test).resh
          ape(-1)})
          alt.Chart(data_linear.melt('day')).mark_line().encode(
              x='day',
              y='value',
              color='variable'
          ).properties(title='test vs predictions').interactive()
```

Out[144]:



Como podemos apreciar, nuestro modelo no sufre de overfitting,aunque tenemos la linea la cual se ajusta a los datos correctamente, podemos notar algo interesante, si bien dijimos que tiene una tendencia creciente, podemos notar que nuestro datos tienen una forma polinomial, por lo que agregar un grado polinomial a nuestro modelo de regresión podria ayudar a mejorar nuestro modelo. Por lo que procedemos a realizarlo

```
In [145]: model_poly = Pipeline([('poly', PolynomialFeatures(degree=4)),
                                  ('linear', LinearRegression(fit_intercept=True))])
          model_poly = model_poly.fit(X_train,y_train)
```

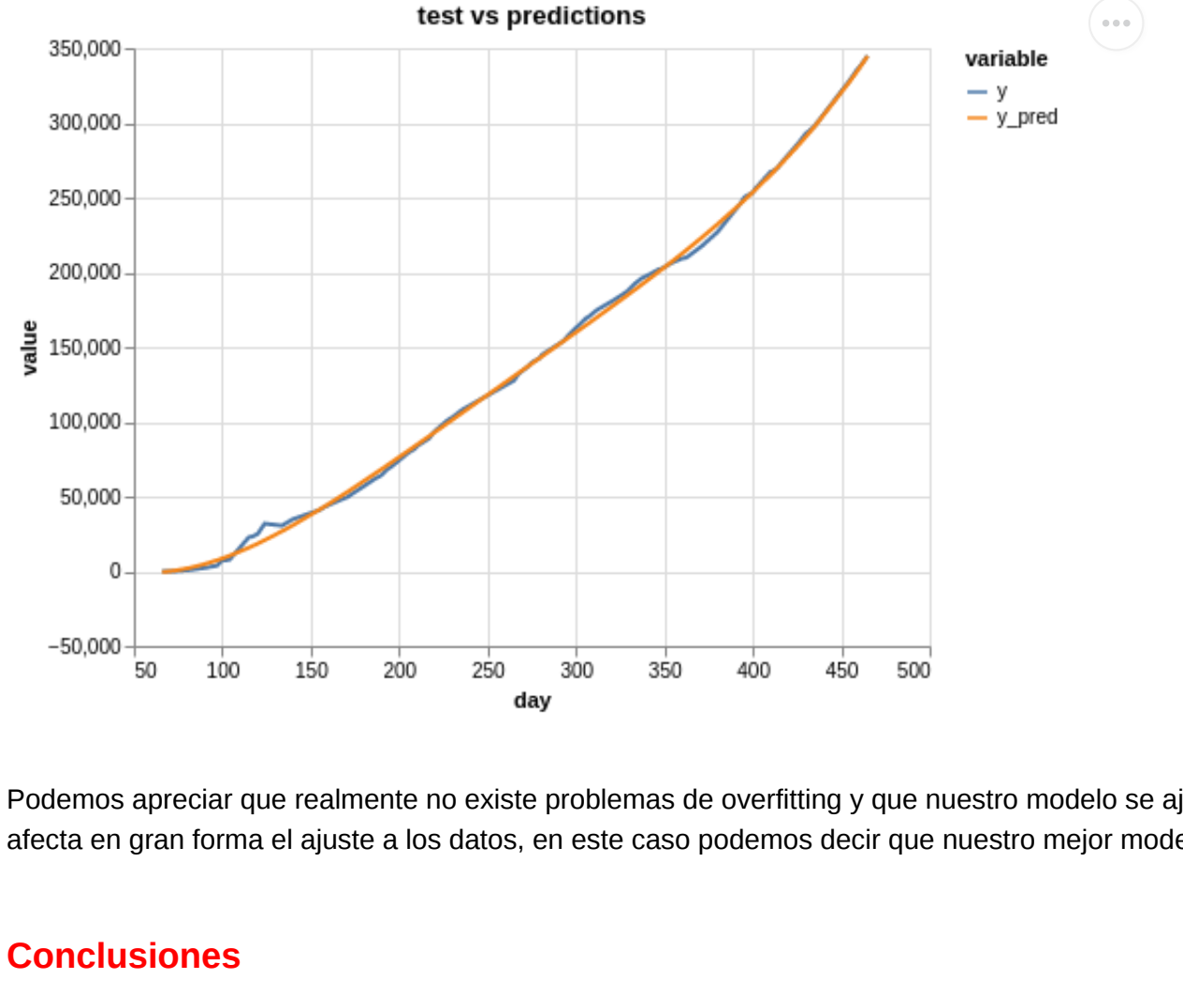
```
In [147]: print("Poly Score : ",model_poly.score(X_test,y_test))
```

```
Poly Score :  0.9990288410813205
```

Como podemos notar, nuestro modelo polinomial tiene un mejor score y esto es gracias al comportamiento de nuestros datos, procedemos a graficarlo para ver si no existe un problema de overfitting y mejoramos la curva

```
In [148]: data_poly=pd.DataFrame({'day':X_test.reshape(-1),'y':y_test.reshape(-1),'y_pred':model_poly.predict(X_test).reshape(
          -1)})
          alt.Chart(data_poly.melt('day')).mark_line().encode(
              x='day',
              y='value',
              color='variable'
          ).properties(title='test vs predictions').interactive()
```

Out[148]:



Podemos apreciar que realmente no existe problemas de overfitting y que nuestro modelo se ajusta a los datos, es interesante ver como el grado polinomial afecta en gran forma el ajuste a los datos, en este caso podemos decir que nuestro mejor modelo sera el **modelo polinomial**

Conclusiones

A lo largo de este informe hemos podido notar que realizar un análisis es un proceso importante para poder darnos cuenta como se estan comportando nuestro conjunto de datos, de la misma manera revisamos dos tipos de regresiones con el fin de ver cual es modelo que mejor se ajusta a nuestros datos y nos dimos cuenta que el **modelo polinomial** resulto ser el mejor modelo para este caso, **ojo** digo para este caso ya que no siempre sera el mejor, eso dependera de como se comportan los datos y del tratamiento que le demos a los mismos. Ademas de este informe, el cual presenta un análisis de todo lo realizado, se decidio tratar de automatizar dicho modelo, con la finaliadd de utilizarlo en producción cuando sea necesario, para ello creamos un entorno virtual con todas las librerias que se necesitan. Dicho programa automatiza el proceso de carga, transformación de datos, obtención de features, target, obtención del mejor modelo, ademas de ello exporta el modelo para poder ser utiliado en cualquier sitio web. A continuación dejo el comando que se requeire para poder ejecutar dicho programa:

```
python main.py in/covid.csv
```

Con ello ya no solamente pensamos en utilizar el modelo en cuadernos jupyter sino también utilizamos python de una manera más profesional con el fin de mejorar las skills del lenguaje.