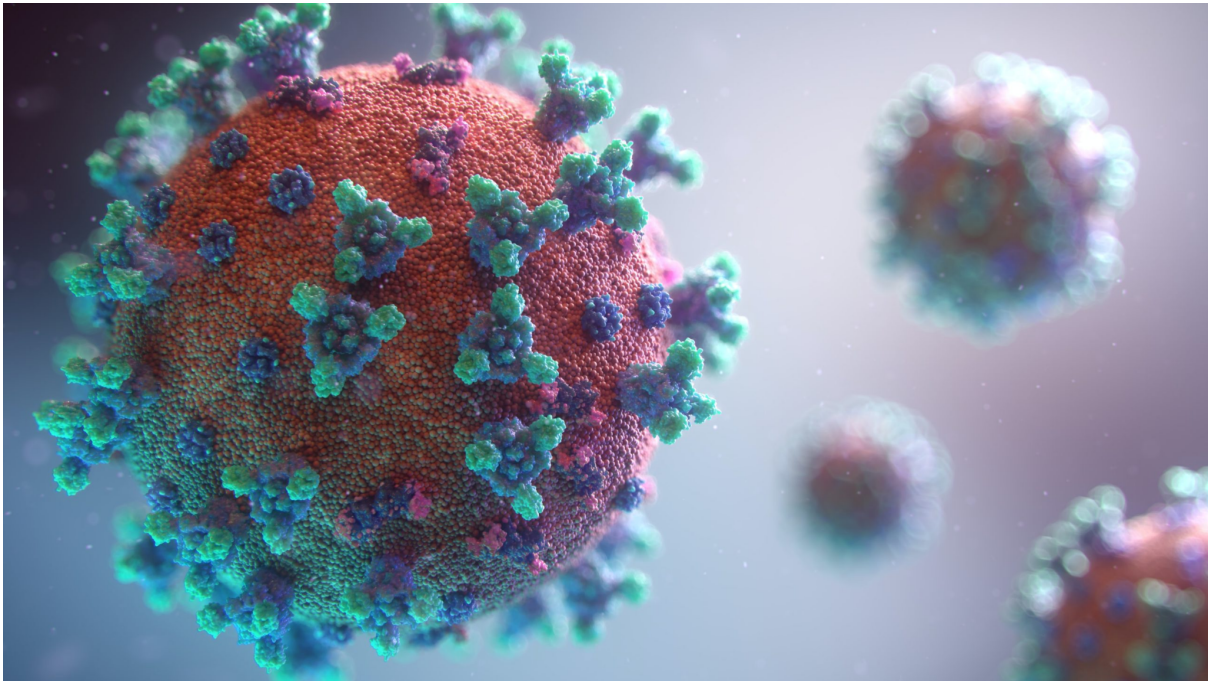


ANALYSIS DATA PUNCAK KASUS DAN BERAKHIRNYA SIKLUS COVID-19



RAKA PUTRA ESHARDIANSYAH

Tools yang digunakan:

1. Google Collabs : <https://colab.research.google.com/>
2. Google Drive : <https://drive.google.com>

Langkah-langkah analisis :

1. Angka dan Data Covid-19 tahun 2020 diambil dari link berikut ini :

https://drive.google.com/file/d/1UpkuGUt1upHC_94UMre96aO7cROzLWlw/view

2. Data Preparation

- Simpan data tersebut di Google Drive lalu copy link file yang terdapat pada Google Drive tersebut.

Contoh :

https://drive.google.com/file/d/1q4MSpRyH5KIQieEZUXSIkcWaVOHRaHoI/view?usp=share_link

- Kemudian pada IDE Google Collabs, inputkan script sebagai berikut untuk melakukan konfirmasi Credentials akses dari Akun Google Collabs yang digunakan ke Akun Penyimpanan Drive.

```
!pip install -U -q PyDrive
```

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

```
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials =
GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

- Kemudian Google akan otomatis melakukan konfirmasi terhadap request dari credentials akun Google yang digunakan saat ini.

- Kemudian import library dari Python yang akan digunakan. Pada project ini, library Python yang digunakan ialah numpy, pandas dan matplotlib.pyplot dengan script sebagai berikut :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- Langkah berikutnya ialah melakukan import terhadap dokumen yang akan dilakukan analysis dengan menggunakan script sebagai berikut :

```
link = downloaded.GetContentFile('covid-data-2020.csv')
df =
pd.read_csv('covid-data-2020.csv') 'https://drive.google.com/
file/d/1q4MSpRyH5KIQieEZUXSlkcWaVQHRahol/view?usp=share_link
'
# to get the id part of the file
id = link.split("/")[-2]
downloaded = drive.CreateFile({'id':id})
```

- Untuk melakukan pengecekan terhadap data yang telah di import, dapat melakukan check dengan script df sehingga menampilkan tampilan sebagai berikut:

✓ [4] df

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	gdp_per_capita	ex
0	AFG	Asia	Afghanistan	2020-02-24	1.0	1.0	NaN	NaN	NaN	NaN	...	1803.987	
1	AFG	Asia	Afghanistan	2020-02-25	1.0	0.0	NaN	NaN	NaN	NaN	...	1803.987	
2	AFG	Asia	Afghanistan	2020-02-26	1.0	0.0	NaN	NaN	NaN	NaN	...	1803.987	
3	AFG	Asia	Afghanistan	2020-02-27	1.0	0.0	NaN	NaN	NaN	NaN	...	1803.987	
4	AFG	Asia	Afghanistan	2020-02-28	1.0	0.0	NaN	NaN	NaN	NaN	...	1803.987	
...
73991	ZWE	Africa	Zimbabwe	2021-03-06	36260.0	12.0	28.857	1485.0	1.0	3.143	...	1899.775	
73992	ZWE	Africa	Zimbabwe	2021-03-07	36271.0	11.0	26.000	1485.0	0.0	3.143	...	1899.775	

- Kemudian untuk proses checking total row yang terdapat pada tersebut dapat menggunakan script :

```
df.shape
```

Dan menampilkan output sebagai berikut :



- Untuk melakukan view by grouping dari beberapa negara dapat menggunakan script sebagai berikut :

```
print(df['location'].value_counts())
```

Dan menghasilkan tampilan sebagai berikut :

```
Argentina      435
Mexico          435
Thailand        432
Taiwan          420
South Korea     415
...
Falkland Islands 16
Montserrat       16
Macao            11
Northern Cyprus   9
Saint Helena      1
Name: location, Length: 215, dtype: int64
```

3. Data Cleansing

- Karena pada pembahasan kali ini menganalisis case Covid-19 yang hanya terjadi di Indonesia, maka perlu untuk memisahkan kasus yang dapat digunakan dengan script sebagai berikut :

```
##DATA CLEANSING
df_ina = df.loc[df['location'] == "Indonesia"]

#digunakan untuk mereset index data menjadi nomor 0
df_ina = df_ina.reset_index(drop=True)
```

- Kemudian view data akan menjadi seperti berikut :

```
#menampilkan 5 data teratas
df_ina.head()
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	gdp_per_capita
0	IDN	Asia	Indonesia	2020-03-02	2.0	2.0	NaN	NaN	NaN	NaN	...	11188.744
1	IDN	Asia	Indonesia	2020-03-03	2.0	0.0	NaN	NaN	NaN	NaN	...	11188.744
2	IDN	Asia	Indonesia	2020-03-04	2.0	0.0	NaN	NaN	NaN	NaN	...	11188.744
3	IDN	Asia	Indonesia	2020-03-05	2.0	0.0	NaN	NaN	NaN	NaN	...	11188.744
4	IDN	Asia	Indonesia	2020-03-06	4.0	2.0	NaN	NaN	NaN	NaN	...	11188.744

5 rows x 59 columns

#pengecekan total record pada data

df_ina.shape

(374, 59)

- Kemudian dari angka diatas, kita dapat mengetahui jika total case yang ada pada tersebut memiliki records sebanyak **374** buah dari **59** kolom yang ada.
- Karena pada data tersebut memiliki 59 kolom dan hal tersebut terlalu banyak untuk di tampilkan satu per satu di tabel, kita dapat menggunakan script sebagai berikut untuk menemukan kolom apa saja yang terdapat pada data Covid-19 dari df_ina.

df_ina.columns

```
Index(['iso_code', 'continent', 'location', 'date',
      'total_cases', 'new_cases',
      'new_cases_smoothed', 'total_deaths', 'new_deaths',
      'new_deaths_smoothed', 'total_cases_per_million',
      'new_cases_per_million',
      'new_cases_smoothed_per_million',
      'total_deaths_per_million', 'new_deaths_per_million',
      'new_deaths_smoothed_per_million',
      'reproduction_rate', 'icu_patients',
      'icu_patients_per_million', 'hosp_patients',
      'hosp_patients_per_million', 'weekly_icu_admissions',
      'weekly_icu_admissions_per_million',
      'weekly_hosp_admissions',
      'weekly_hosp_admissions_per_million', 'new_tests',
      'total_tests',
```

```

        'total_tests_per_thousand', 'new_tests_per_thousand',
        'new_tests_smoothed',
        'new_tests_smoothed_per_thousand',
        'positive_rate', 'tests_per_case', 'tests_units',
        'total_vaccinations',
        'people_vaccinated', 'people_fully_vaccinated',
        'new_vaccinations',
        'new_vaccinations_smoothed',
        'total_vaccinations_per_hundred',
        'people_vaccinated_per_hundred',
        'people_fully_vaccinated_per_hundred',
        'new_vaccinations_smoothed_per_million',
        'stringency_index',
        'population', 'population_density', 'median_age',
        'aged_65_older',
        'aged_70_older', 'gdp_per_capita', 'extreme_poverty',
        'cardiovasc_death_rate', 'diabetes_prevalence',
        'female_smokers',
        'male_smokers', 'handwashing_facilities',
        'hospital_beds_per_thousand',
        'life_expectancy', 'human_development_index'],
        dtype='object')

```


- Dari hasil tampilan diatas, kita dapat mengetahui semua kolom yang terdapat pada data yang kita analisis. Kemudian pada langkah selanjutnya, kita pisahkan beberapa field yang kita butuhkan untuk meningkatkan efektifitas pada hasil data analysis. Field yang dibutuhkan pada analisis kali ini ialah "date", "total_cases", "new_cases", "total_deaths" dan "new_deaths".
- Untuk memisahkan field yang diperlukan dapat menggunakan script sebagai berikut :

```

#mengambil beberapa data yang diperlukan untuk analisis
final_df_ina = df_ina[["date", "total_cases", "new_cases",
"total_deaths", "new_deaths"]]

```

- Dengan menggunakan variabel baru, kita dapat melakukan pengecekan data yang sudah dipisahkan.

✓ 0s  final_df_ina

	date	total_cases	new_cases	total_deaths	new_deaths
0	2020-03-02	2.0	2.0	NaN	NaN
1	2020-03-03	2.0	0.0	NaN	NaN
2	2020-03-04	2.0	0.0	NaN	NaN
3	2020-03-05	2.0	0.0	NaN	NaN
4	2020-03-06	4.0	2.0	NaN	NaN
...
369	2021-03-06	1373836.0	5767.0	37154.0	128.0
370	2021-03-07	1379662.0	5826.0	37266.0	112.0
371	2021-03-08	1386556.0	6894.0	37547.0	281.0
372	2021-03-09	1392945.0	6389.0	37757.0	210.0

- Jika dilihat dari data yang ada, terdapat beberapa missing values dari record yang ada. Untuk melihat semua data yang missing, kita dapat menggunakan script sebagai berikut :

```
final_df_ina.isnull().sum()

date          0
total_cases   0
new_cases     0
total_deaths  9
new_deaths    9
dtype: int64
```

- Dari informasi diatas, kita dapat mengetahui jika terdapat sebanyak 9 record pada column total_deaths dan new_deaths yang tidak memiliki isi records.

- Untuk mengatasi missing values, kita dapat menggunakan beberapa metode diantaranya menggunakan Mean/Median, menghapus record atau mengisi values dengan angka yang sesuai tergantung dari case yang sedang di analisis. Pada case analysis ini, nilai NaN sebagian besar missing values terdapat diawal data. Dimana data ini di order dari field date sehingga hal ini merupakan case dari awal kejadian Covid-19. Sehingga dari case yang dijabarkan diatas dapat disimpulkan jika nilai 0 sebagai anggapan awal menjadi **valid** untuk mengisi missing values yang ada. Hal tersebut dapat di eksekusi dengan menggunakan field sebagai berikut.

```
#cleaning data yang memiliki records NaN agar data yang
dihasilkan tidak bias.
```

```
final_df_ina = final_df_ina.fillna(0)
```

```
#data setelah cleaning
final_df_ina
```

	date	total_cases	new_cases	total_deaths	new_deaths
0	2020-03-02	2.0	2.0	0.0	0.0
1	2020-03-03	2.0	0.0	0.0	0.0
2	2020-03-04	2.0	0.0	0.0	0.0
3	2020-03-05	2.0	0.0	0.0	0.0
4	2020-03-06	4.0	2.0	0.0	0.0
...
369	2021-03-06	1373836.0	5767.0	37154.0	128.0
370	2021-03-07	1379662.0	5826.0	37266.0	112.0
371	2021-03-08	1386556.0	6894.0	37547.0	281.0
372	2021-03-09	1392945.0	6389.0	37757.0	210.0
373	2021-03-10	1398578.0	5633.0	37932.0	175.0

374 rows x 5 columns

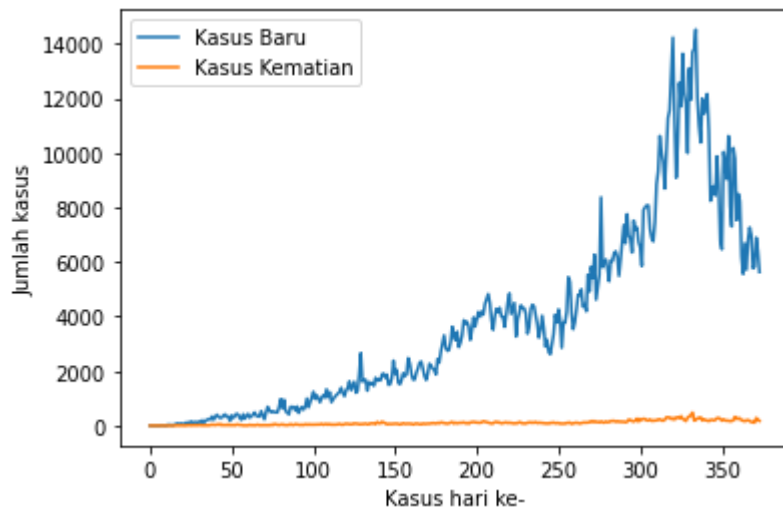
4. Exploration Data Analyst

- Pada tahapan EDA terbagi menjadi 3 jenis yaitu : Univariat Analysis (Analisa deskriptif dengan 1 variabel), Bivariat Analysis (Analisa deskriptif dengan 2 variabel) dan Multivariat (Analisa deskriptif dengan 3 variabel). Ketiga jenis EDA diatas dapat disesuaikan kembali dengan case yang dibahas. Pada case kali ini hanya menggunakan 2 dari 3 jenis EDA, yakni Univariat Analysis dan Bivariat Analysis.

- Pada cases kali ini menggunakan 2 operator pembandingan yaitu new_cases dan new deaths yang kemudian di visualisasikan menggunakan library matplotlib dengan script sebagai berikut.

```
final_df_ina["new_cases"].plot(label = "Kasus Baru")
final_df_ina["new_deaths"].plot(label = "Kasus Kematian")
plt.xlabel("Kasus hari ke-")
plt.ylabel("Jumlah kasus")
plt.legend(loc="best")
plt.show()
```

dan menghasilkan visualisasi sebagai berikut :



- Kemudian untuk melakukan eksplorasi terhadap data puncak kasus yang terjadi dapat menggunakan script sebagai berikut.

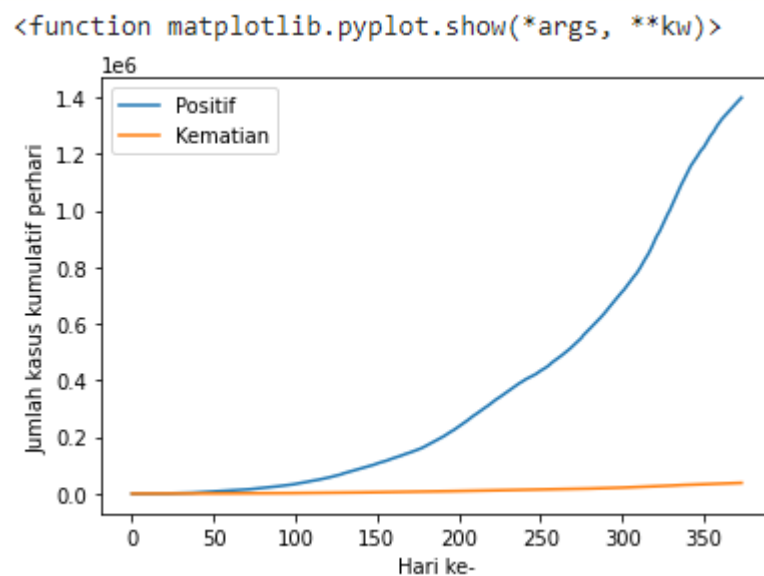
```
#mencari puncak kasus pada data yang ada
for i in range(len(final_df_ina)):
    if final_df_ina['new_cases'].iloc[i] ==
final_df_ina['new_cases'].max():
        print("Tanggal kasus positif tertinggi {}".
format(final_df_ina['date'].iloc[i]))
        print("Dengan jumlah
{}".format(final_df_ina['new_cases'].iloc[i]))
```

```
Tanggal kasus positif tertinggi 2021-01-30
Dengan jumlah 14518.0
```

- Dari data diatas dapat diketahui jika data kasus tertinggi di Indonesia terjadi pada tanggal 30 Januari 2021 dengan jumlah kasus sebanyak **14.518**.
- Kemudian untuk Total Case Positif dengan Total Case Kematian juga perlu untuk dianalisis untuk mendapatkan angka kumulatif harian sehingga dapat diketahui

seberapa besar jumlah kasus sehingga dapat menemukan solusi pertolongan yang tepat. Perbangan tersebut dapat menggunakan script berikut untuk mendapatkan informasi yang diinginkan.

```
final_df_ina['total_cases'].plot(label = "Positif")
final_df_ina['total_deaths'].plot(label = "Kematian")
plt.xlabel("Hari ke-")
plt.ylabel("Jumlah kasus kumulatif perhari")
plt.legend(loc="best")
plt.show
```



- Dari data diatas terlihat jika jumlah penularan perhari sangat cepat dibandingkan dengan jumlah kematian yang ada akibat Covid-19.

5. MODEL PREDIKSI

- Model prediksi yang digunakan pada case ini ialah untuk melakukan prediksi dimana Sigmoid diperlukan pada aktifitasnya. Rumus original Sigmoid ialah sebagai berikut.

$$sig(t) = \frac{1}{1 + e^{-t}}$$

$$sig_m(t) = \frac{c}{1 + e^{-(t-t_0)/a}}$$

- Pada sebuah grafik Sigmoid utuh memiliki beberapa proses diantaranya ialah proses Saturasi dan proses Ekspansional. Disisi ujung sigmoid terdapat sebuah garis lurus bernama Saturasi. Saturasi merupakan angka tidak lagi mengalami pertambahan atau dapat disebut sebagai titik puncak. Sedangkan angka kenaikan pada grafik Sigmoid bernama Ekspansional. Proses ekspansional merupakan titik penambahan tertinggi.
- Dimana sigmoid original dilakukan modifikasi yang menyesuaikan dengan case Covid-19 dimana c merupakan puncak kasus, t-t0 selisih dari waktu ke waktu dibagi dengan a yang merupakan kumulatif kenaikannya.

c = Nilai Puncak

a = koefisien curve fitting (pencocokan kurva)

t = jumlah hari

t0 = jumlah kasus pada hari pertama

- Untuk menghitung dengan menggunakan kurva Sigmoid di Python, tentunya perlu untuk melakukan mendefinisikan terlebih dahulu dengan sebuah function dengan script sebagai berikut.

```
def kurva_sigmoid(t, a, t0, c):
    return c / (1 + np.exp(-(t - t0) / (a)))
```

- Kemudian variabel yang digunakan pada analisis ini ialah dengan menggunakan **total kasus** pada records data dan **total case**.

```
#Menentukan Index
```

```
x = list(final_df_ina.index)
y = list(final_df_ina['total_cases'])
```

- Untuk mengimplementasikan hal tersebut, perlu library tambahan untuk menghitung sebuah nilai prediksi actual + nilai errornya.

```
from scipy.optimize import curve_fit, fsolve
```

- Dengan menggunakan metode curve fitting pada Python memerlukan rumus yang ada dan perlu mengindikasikan nilai variable yang akan dibandingkan. (Pada case ini menggunakan **total records** dan **total kasus**)

```
#fit membutuhkan inputan dari kurva, x = jumlah semua
    records di data, y = total kumulatif, dan method trf
    as default
fit=curve_fit(kurva_sigmoid, x, y, method='trf')

#Memberikan 2 nilai untuk fit karena membutuhkan 2 variabel
    nilai actual yang dibandingkan dengan nilai error
varA, varB=fit
```

- Jika kita mengetikkan script : varA, Maka akan menghasilkan output sebagai berikut.

```
array([6.91350214e+01, 3.71790857e+02, 2.84763171e+06])
```

Angka tersebut menunjukkan nilai a,t0 dan c secara berurutan dan jika mengetikkan script : varB, maka akan menghasilkan output sebagai berikut.

```
array([[1.28640987e-33, 3.43454479e-34, 1.14555555e-34],
       [3.43454479e-34, 1.76272864e-33, 1.48498795e-34],
       [1.14555555e-34, 1.48498795e-34, 4.95301790e-35]])
```

Nilai tersebut merupakan beberapa nilai error yang dihasilkan pada hasil fitting.

- Untuk menghitung nilai prediksi, perlu untuk mendefinisikan nilai standar error dari varA yang telah di deklarasikan diatas untuk mendapatkan nilai hasil yang akurat.

```
#Memperoleh standar nilai error dari varB dengan standar
diagonal dari varA
std_er = np.zeros(len(varA))
for i in range(len(varA)):
    std_er[i] = np.sqrt(varB[i][i])
```

- Kemudian ketika mengetikkan script `std_er` akan menghasilkan output sebagai berikut.

```
array([7.60500011e-01, 3.68695257e+00, 8.35528444e+04])
```

Nilai diatas merupakan nilai error secara berurut dari variable a,t0 dan c.

- Kemudian untuk mengisi nilai prediksi dengan menggabungkan nilai yang ada di varA secara diagonal dengan nilai standar error yang dihasilkan.

```
#mengisikan nilai prediksi dengan menggabungkan varA dengan
standar Error
a = varA[0]+std_er[0]
t0 = varA[1]+std_er[1]
c = varA[2]+std_er[2]
```

- Setelah mendapatkan angka prediksi, kita dapat membuat sebuah fungsi dengan mengimplementasikan kurva Sigmoid kembali dengan angka prediksi yang telah didapatkan sebelumnya.

```
#fungsi prediksi nilai puncak
def puncak(x):
    return kurva_sigmoid(x,a,t0,c) - int(c)
```

- Untuk mengeksekusi hasil prediksi dengan menggunakan `fsolve`.

```
#inisialisasi dengan menggunakan fsolve
n_puncak = int(fsolve(puncak,t0))

print("Hari Puncak Adalah "+str(n_puncak))
```

Hari Puncak Adalah 1458

- Sehingga terlihat jika hari puncak dari Covid-19 ialah hari ke 1458 sedangkan records yang terdapat di Indonesia saat ini masih berada di angka 370-an. Hal ini dapat disimpulkan jika pandemi yang dialami oleh Indonesia masih akan berlanjut diperkirakan hingga hari ke-1458 atau lebih.

- Saat ini data masih berada di range 370an. Untuk mengetahui range hingga ke hari puncak dapat menggunakan script sebagai berikut.

```
#Memprediksi range hari sebelum menuju puncak
n_0 = max(x)+1
pred_x = list(range(n_0,n_puncak))

print("Kita memiliki {} hari lagi hingga mencapai hari
puncak".format(str(len(pred_x))))

Kita memiliki 1084 hari lagi hingga mencapai hari puncak
```

- Dari data diatas dapat disimpulkan dari hari ini hingga hari puncak masih terdapat 1084 hari lagi. Kemudian untuk menentukan jumlah case terinfeksi pada hari tersebut dapat menggunakan script sebagai berikut.

```
#Menentukan
pred_y = np.zeros(len(x+pred_x))

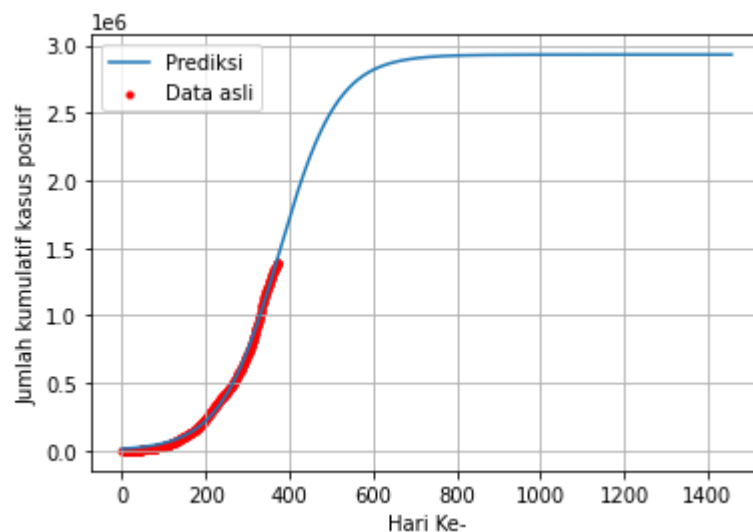
for i in range(n_puncak):
    pred_y[i] = kurva_sigmoid(i,a,t0,c)

print("Prediksi jumlah puncak berjumlah {}
orang".format(int(pred_y[-1])))
```

Prediksi jumlah puncak berjumlah 2931183 orang

- Dimana prediksi puncak akan menginfeksi sebanyak 2.931.183 orang.
- Setelah menentukan jika variabel x merupakan total kasus saat ini, pred x merupakan prediksi puncak, y sebagai total kasus dan pred_y sebagai prediksi jumlah kasus kumulatif kemudian dapat dilakukan visualisasi data agar mudah untuk dipahami.

```
plt.plot(x+pred_x, pred_y, label = "Prediksi")
plt.scatter(x,y,label = "Data asli", s=10, color="Red")
plt.xlabel("Hari Ke-")
plt.ylabel("Jumlah kumulatif kasus positif")
plt.grid()
plt.legend(loc = "best")
plt.show()
```



- Dari data diatas dapat disimpulkan jika case yang terjadi saat ini masih di fase pertengahan hingga mencapai puncak kasus.
- Untuk uji akurasi case kali ini menggunakan R2 Score. Dimana R2 score membandingkan nilai Asli dengan nilai

Prediksi. R2 score memiliki rumus secara matematis sebagai berikut.

$$R^2 = 1 - \frac{S_{res}}{S_{tot}}$$
$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

dimana :

- y_i = nilai total kasus saat ini
 - y_{topi} = nilai prediksi total kasus
 - y_{bar} = nilai prediksi rata-rata
- Kemudian rumus tersebut di transformasikan kedalam bentuk script python dalam sebuah fungsi akurasi_r2.

#Fungsi menghitung akurasi sesuai dengan nilai prediksi

```
def akurasi_r2(y_asli, y_prediksi, x):  
    atas = sum((y_asli-y_prediksi[0:len(x)])**2)  
    bawah = sum((y_asli-np.mean(y))**2)  
    r = 1-atas/bawah  
    return r
```

#Menghitung nilai akurasi

```
akurasi = akurasi_r2(y, pred_y, x)  
akurasi*100
```

99.74351981377066

- Dari hasil akurasi yang dihasilkan, penelitian pada case ini memiliki tingkat akurasi sebesar 99,74%.
- Pada penjelasan diatas, case dari Covid-19 akan berakhir di hari ke-1458. Untuk menentukan hari apa atau tanggal berapa saat itu terjadi, maka kita perlu menganalisis kembali dengan menggunakan library tambahan yaitu datetime.

```
from datetime import datetime, timedelta
```

- Kemudian untuk menentukan tanggal puncak, kita perlu mengetahui case saat pertama kali terjadi. Case covid-19 di Indonesia pertama kali ditemukan pada 2020-03-02. Untuk menentukan hari puncak, tentunya perlu dirubah format dari String ke Datetime terlebih dahulu.

```
n1 = final_df_ina['date'][0]
tgl_0 = datetime.strptime(n1, '%Y-%m-%d')
print(tgl_0)
```

```
2020-03-02 00:00:00
```

- Setelah merubah format ke time, kita dapat menentukan nilai dari n_puncak dengan menggunakan library timedelta.

```
tgl_puncak = tgl_0+timedelta(days=n_puncak)
print(tgl_puncak)
```

```
2024-02-28 00:00:00
```

- Maka case puncak/terakhir untuk Covid-19 akan berakhir pada tanggal 28-Februari-2024.