# Implementasi Data Covid dengan Linear Regression

Kelompok 2 :

- Arfian Ramdhani - 41519120005
- Kartika Nirwana Silalahi - 41519120006
- Baeningrum Syahfutri - 41519120036

# Linear Regression

# Pengertian Linear Regression

- Regresi linear adalah sebuah pendekatan untuk memodelkan hubungan antara variabel terikat Y dan satu atau lebih variabel bebas yang disebut X. Salah satu kegunaan dari regresi linear adalah untuk melakukan prediksi berdasarkan data-data yang telah dimiliki sebelumnya.

- **Regresi Linier** atau **Linear Regression** adalah suatu model statistik yang umum dan paling sederhana yang digunakan untuk Machine Learning untuk melakukan prediksi dengan cara supervised learning.

# Pengertian Linear Regression

Regresi Linier melibatkan 2 variabel dimana salah satunya adalah variabel independen (x) dan satu lagi adalah variabel dependen (y).

- Independen berarti variabel ini sebagai variabel utama yang mungkin akan mempengaruhi nilai variabel kedua (dependen).
- Dependen berarti nilai variabel ini akan tergantung dari nilai variabel independennya, jika korelasi tinggi maka dependensi juga tinggi.

# Pengertian Polynomial Regression

Regresi polinomial merupakan regresi di mana fungsinya adalah kuadratik.

Jika kita ingin mencari hubungan antara 1 variabel dependen dengan 1 variabel independen, maka bisa menggunakan simple atau poli.

Regresi Polinomial adalah suatu bentuk regresi linier di mana hubungan antara variabel independen x dan variabel dependen y dimodelkan sebagai polinomial derajat ke-n.

# Kenapa Polynomial Regression ?

- Ada beberapa hubungan yang akan dihipotesiskan oleh peneliti sebagai kurva. Jelas, jenis kasus seperti itu akan mencakup istilah polinomial.
- Pemeriksaan residu. Jika kita mencoba menyesuaikan model linier dengan data melengkung, sebaran residual (sumbu Y) pada prediktor (sumbu X) akan memiliki tambalan dari banyak residu positif di tengah. Maka dalam situasi seperti itu tidak tepat.
- Asumsi dalam analisis regresi linier berganda yang biasa adalah bahwa semua variabel independen adalah independen. Dalam model regresi polinomial, asumsi ini tidak puas

# Studi Kasus (Covid-19)

# Implementasi Prediksi Covid 19

Covid-19 adalah adalah virus yang diidentifikasi sebagai penyebab meningkatnya jumlah penyakit pernafasan di Wuhan, Cina secara mendadak. Awalnya, pasien-pasien pertama di Wuhan diberitakan memiliki hubungan dengan pusat penjualan makanan-makanan laut dan hewan,menandakan bahwa kemungkinan virus ini menjangkit dari hewan ke manusia. Namun, setelah itu, pasien-pasien bermunculan yang tidak memiliki hubungan dengan pusat penjualanan makanan tersebut, menunjukkan bahwa virus ini juga dapat menyebar dari manusia ke manusia. Karena virus ini baru muncul, maka belum jelas berapa mudah virus ini menyebar dan apa saja efeknya bagi yang terjangkit.

Dataset ini berisi informasi harian jumlah kasus infeksi, kematian, dan penyembuhan dari 2019 Novel Coronavirus.

# Import Module

- Untuk Olahdata

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

  - Persiapan Dataset

```
[ ] import os
    for dirname, _, filenames in os.walk('./covid/data'):
        for filename in filenames:
            print(os.path.join(dirname, filename))
```

```
./covid/data/train.csv
./covid/data/test.csv
./covid/data/submission.csv
```

# Import Module

- Untuk Plot

```
import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly
plotly.offline.init_notebook_mode() # For not show up chart error

import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML
%matplotlib inline

from tqdm import tqdm
def RMSLE(pred,actual):
    return np.sqrt(np.mean(np.power((np.log(pred+1)-np.log(actual+1)),2)))
```

# Pengambilan Dataset

```
pd.set_option('mode.chained_assignment', None)
test = pd.read_csv("./covid/data/test.csv")
train = pd.read_csv("./covid/data/train.csv")
train['Province_State'].fillna('', inplace=True)
test['Province_State'].fillna('', inplace=True)
train['Date'] =  pd.to_datetime(train['Date'])
test['Date'] =  pd.to_datetime(test['Date'])
train = train.sort_values(['Country_Region','Province_State','Date'])
test = test.sort_values(['Country_Region','Province_State','Date'])
```

- **Untuk Fix error di train data**

```
train[['ConfirmedCases', 'Fatalities']] =
train.groupby(['Country_Region', 'Province_State'])[['ConfirmedCases',
```

```
[21] , 'Fatalities']] = train.groupby(['Country_Region', 'Province_State'])[['ConfirmedCases', 'Fatalities']].transform('cummax')
```

# Algoritma Linear Regression

- **Forecast with BayesianRidge**

  *Forecasting* atau perkiraan adalah kegiatan yang bertujuan untuk meramalkan atau memprediksi

```
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

feature_day = [1,20,50,100,200,500,1000]
def CreateInput(data):
    feature = []
    for day in feature_day:
        #Get information in train data
        data.loc[:,'Number day from ' + str(day) + ' case'] = 0
        if (train[(train['Country_Region'] == country) & (train['Province_State'] == province) & (train['ConfirmedCases'] < day)]['Date'].count() > 0):
            fromday = train[(train['Country_Region'] == country) & (train['Province_State'] == province) & (train['ConfirmedCases'] < day)]['Date'].max()
        else:
            fromday = train[(train['Country_Region'] == country) & (train['Province_State'] == province)]['Date'].min()
        for i in range(0, len(data)):
            if (data['Date'].iloc[i] > fromday):
                day_denta = data['Date'].iloc[i] - fromday
                data['Number day from ' + str(day) + ' case'].iloc[i] = day_denta.days
        feature = feature + ['Number day from ' + str(day) + ' case']

    return data[feature]
```

# Algoritma Linear Regression

```
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

feature_day = [1,20,50,100,200,500,1000]
def CreateInput(data):
    feature = []
    for day in feature_day:
        #Get information in train data
        data.loc[:,'Number day from ' + str(day) + ' case'] = 0
        if (train[(train['Country_Region'] == country) & (train['Province_State'] == province) & (train['ConfirmedCases'] < day)]['Date'].count() > 0):
            fromday = train[(train['Country_Region'] == country) & (train['Province_State'] == province) & (train['ConfirmedCases'] < day)]['Date'].max()
        else:
            fromday = train[(train['Country_Region'] == country) & (train['Province_State'] == province)]['Date'].min()
        for i in range(0, len(data)):
            if (data['Date'].iloc[i] > fromday):
                day_denta = data['Date'].iloc[i] - fromday
                data['Number day from ' + str(day) + ' case'].iloc[i] = day_denta.days
        feature = feature + ['Number day from ' + str(day) + ' case']

    return data[feature]
pred_data_all = pd.DataFrame()
with tqdm(total=len(train['Country_Region'].unique())) as pbar:
    for country in train['Country_Region'].unique():
    #for country in ['Japan']:
        for province in train[(train['Country_Region'] == country)]['Province_State'].unique():
            df_train = train[(train['Country_Region'] == country) & (train['Province_State'] == province)]
            df_test = test[(test['Country_Region'] == country) & (test['Province_State'] == province)]
            X_train = CreateInput(df_train)
            y_train_confirmed = df_train['ConfirmedCases'].ravel()
            y_train_fatalities = df_train['Fatalities'].ravel()
            X_pred = CreateInput(df_test)
```

```
            # Define feature to use by X_pred
            feature_use = X_pred.columns[0]
            for i in range(X_pred.shape[1] - 1,0,-1):
                if (X_pred.iloc[0,i] > 0):
                    feature_use = X_pred.columns[i]
                    break
            idx = X_train[X_train[feature_use] == 0].shape[0]
            adjusted_X_train = X_train[idx:][feature_use].values.reshape(-1, 1)
            adjusted_y_train_confirmed = y_train_confirmed[idx:]
            adjusted_y_train_fatalities = y_train_fatalities[idx:] #.values.reshape(-1, 1)

            adjusted_X_pred = X_pred[feature_use].values.reshape(-1, 1)

            model = make_pipeline(PolynomialFeatures(2), BayesianRidge())
            model.fit(adjusted_X_train,adjusted_y_train_confirmed)
            y_hat_confirmed = model.predict(adjusted_X_pred)

            model.fit(adjusted_X_train,adjusted_y_train_fatalities)
            y_hat_fatalities = model.predict(adjusted_X_pred)

            pred_data = test[(test['Country_Region'] == country) & (test['Province_State'] == province)]
            pred_data['ConfirmedCases_hat'] = y_hat_confirmed
            pred_data['Fatalities_hat'] = y_hat_fatalities
            pred_data_all = pred_data_all.append(pred_data)
        pbar.update(1)

df_val = pd.merge(pred_data_all,train[['Date','Country_Region','Province_State','ConfirmedCases','Fatalities']],on=['Date','Country_Region','Province_State'], how='left')
df_val.loc[df_val['Fatalities_hat'] < 0,'Fatalities_hat'] = 0
df_val.loc[df_val['ConfirmedCases_hat'] < 0,'ConfirmedCases_hat'] = 0

df_val_1 = df_val.copy()
```

```
100%|          | 173/173 [01:05<00:00,  2.63it/s]
```

# Confirmed Cases dan Fatalities

```
[ ] RMSLE(df_val[(df_val['ConfirmedCases'].isnull() == False)]['ConfirmedCases'].values,df_val[(df_val['ConfirmedCases'].isnull() == False)]['ConfirmedCases_hat'].values)

    0.19742156774212843

[ ] RMSLE(df_val[(df_val['Fatalities'].isnull() == False)]['Fatalities'].values,df_val[(df_val['Fatalities'].isnull() == False)]['Fatalities_hat'].values)

    0.1828241464847915
```

# Confirmed Cases dan Fatalities Country

```
val_score = []
for country in df_val['Country_Region'].unique():
    df_val_country = df_val[(df_val['Country_Region'] == country) & (df_val['Fatalities'].isnull() == False)]
    val_score.append([country, RMSLE(df_val_country['ConfirmedCases'].values,df_val_country['ConfirmedCases_hat'].values),RMSLE(df_val_country['Fatalities'].values,df_val_country['

df_val_score = pd.DataFrame(val_score)
df_val_score.columns = ['Country','ConfirmedCases_Scored','Fatalities_Scored']
df_val_score.sort_values('ConfirmedCases_Scored', ascending = False)
```

| | Country | ConfirmedCases_Scored | Fatalities_Scored |
|---|---|---|---|
| 101 | Mali | 1.121247 | 0.133608 |
| 164 | Ukraine | 0.854126 | 0.244006 |
| 163 | Uganda | 0.656863 | 0.000000 |
| 65 | Grenada | 0.565423 | 0.000000 |
| 68 | Guinea-Bissau | 0.522935 | 0.000000 |
| ... | ... | ... | ... |
| 99 | Malaysia | 0.020022 | 0.212704 |
| 151 | Sweden | 0.019551 | 0.183070 |
| 128 | Poland | 0.014691 | 0.082859 |
| 44 | Diamond Princess | 0.010608 | 0.084689 |
| 136 | Saint Vincent and the Grenadines | 0.000000 | 0.000000 |

173 rows × 3 columns

# Cases Indonesia

```
country = "Indonesia"
df_val = df_val_1
df_val[df_val['Country_Region'] == country].groupby(['Date','Country_Region']).sum().reset_index()
```

| | Date | Country_Region | ForecastId | ConfirmedCases_hat | Fatalities_hat | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|---|
| 0 | 2020-03-19 | Indonesia | 5806 | 297.940261 | 26.445855 | 311.0 | 25.0 |
| 1 | 2020-03-20 | Indonesia | 5807 | 363.453291 | 30.708079 | 369.0 | 32.0 |
| 2 | 2020-03-21 | Indonesia | 5808 | 436.044137 | 35.892138 | 450.0 | 38.0 |
| 3 | 2020-03-22 | Indonesia | 5809 | 515.712798 | 41.998033 | 514.0 | 48.0 |
| 4 | 2020-03-23 | Indonesia | 5810 | 602.459274 | 49.025764 | 579.0 | 49.0 |
| 5 | 2020-03-24 | Indonesia | 5811 | 696.283566 | 56.975330 | 686.0 | 55.0 |
| 6 | 2020-03-25 | Indonesia | 5812 | 797.185673 | 65.846731 | 790.0 | 58.0 |
| 7 | 2020-03-26 | Indonesia | 5813 | 905.165596 | 75.639968 | 893.0 | 78.0 |
| 8 | 2020-03-27 | Indonesia | 5814 | 1020.223334 | 86.355041 | 1046.0 | 87.0 |
| 9 | 2020-03-28 | Indonesia | 5815 | 1142.358887 | 97.991948 | 1155.0 | 102.0 |
| 10 | 2020-03-29 | Indonesia | 5816 | 1271.572256 | 110.550692 | 1285.0 | 114.0 |
| 11 | 2020-03-30 | Indonesia | 5817 | 1407.863440 | 124.031270 | 1414.0 | 122.0 |
| 12 | 2020-03-31 | Indonesia | 5818 | 1551.232439 | 138.433684 | 1528.0 | 136.0 |
| 13 | 2020-04-01 | Indonesia | 5819 | 1701.679254 | 153.757934 | 0.0 | 0.0 |
| 14 | 2020-04-02 | Indonesia | 5820 | 1859.203884 | 170.004019 | 0.0 | 0.0 |
| 15 | 2020-04-03 | Indonesia | 5821 | 2023.806330 | 187.171939 | 0.0 | 0.0 |
| 16 | 2020-04-04 | Indonesia | 5822 | 2195.486591 | 205.261695 | 0.0 | 0.0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 2020-04-05 | Indonesia | 5823 | 2374.244667 | 224.273287 | 0.0 | 0.0 |
| 18 | 2020-04-06 | Indonesia | 5824 | 2560.080559 | 244.206714 | 0.0 | 0.0 |
| 19 | 2020-04-07 | Indonesia | 5825 | 2752.994266 | 265.061976 | 0.0 | 0.0 |
| 20 | 2020-04-08 | Indonesia | 5826 | 2952.985788 | 286.839074 | 0.0 | 0.0 |
| 21 | 2020-04-09 | Indonesia | 5827 | 3160.055126 | 309.538007 | 0.0 | 0.0 |
| 22 | 2020-04-10 | Indonesia | 5828 | 3374.202279 | 333.158775 | 0.0 | 0.0 |
| 23 | 2020-04-11 | Indonesia | 5829 | 3595.427248 | 357.701379 | 0.0 | 0.0 |
| 24 | 2020-04-12 | Indonesia | 5830 | 3823.730032 | 383.165819 | 0.0 | 0.0 |
| 25 | 2020-04-13 | Indonesia | 5831 | 4059.110631 | 409.552094 | 0.0 | 0.0 |
| 26 | 2020-04-14 | Indonesia | 5832 | 4301.569046 | 436.860204 | 0.0 | 0.0 |
| 27 | 2020-04-15 | Indonesia | 5833 | 4551.105276 | 465.090150 | 0.0 | 0.0 |
| 28 | 2020-04-16 | Indonesia | 5834 | 4807.719322 | 494.241932 | 0.0 | 0.0 |
| 29 | 2020-04-17 | Indonesia | 5835 | 5071.411182 | 524.315548 | 0.0 | 0.0 |
| 30 | 2020-04-18 | Indonesia | 5836 | 5342.180859 | 555.311001 | 0.0 | 0.0 |
| 31 | 2020-04-19 | Indonesia | 5837 | 5620.028350 | 587.228288 | 0.0 | 0.0 |
| 32 | 2020-04-20 | Indonesia | 5838 | 5904.953657 | 620.067411 | 0.0 | 0.0 |
| 33 | 2020-04-21 | Indonesia | 5839 | 6196.956780 | 653.828370 | 0.0 | 0.0 |
| 34 | 2020-04-22 | Indonesia | 5840 | 6496.037717 | 688.511164 | 0.0 | 0.0 |
| 35 | 2020-04-23 | Indonesia | 5841 | 6802.196470 | 724.115793 | 0.0 | 0.0 |
| 36 | 2020-04-24 | Indonesia | 5842 | 7115.433039 | 760.642258 | 0.0 | 0.0 |
| 37 | 2020-04-25 | Indonesia | 5843 | 7435.747423 | 798.090559 | 0.0 | 0.0 |
| 38 | 2020-04-26 | Indonesia | 5844 | 7763.139622 | 836.460695 | 0.0 | 0.0 |
| 39 | 2020-04-27 | Indonesia | 5845 | 8097.609637 | 875.752666 | 0.0 | 0.0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 20 | 2020-04-08 | Indonesia | 5826 | 2952.985788 | 286.839074 | 0.0 | 0.0 |
| 21 | 2020-04-09 | Indonesia | 5827 | 3160.055126 | 309.538007 | 0.0 | 0.0 |
| 22 | 2020-04-10 | Indonesia | 5828 | 3374.202279 | 333.158775 | 0.0 | 0.0 |
| 23 | 2020-04-11 | Indonesia | 5829 | 3595.427248 | 357.701379 | 0.0 | 0.0 |
| 24 | 2020-04-12 | Indonesia | 5830 | 3823.730032 | 383.165819 | 0.0 | 0.0 |
| 25 | 2020-04-13 | Indonesia | 5831 | 4059.110631 | 409.552094 | 0.0 | 0.0 |
| 26 | 2020-04-14 | Indonesia | 5832 | 4301.569046 | 436.860204 | 0.0 | 0.0 |
| 27 | 2020-04-15 | Indonesia | 5833 | 4551.105276 | 465.090150 | 0.0 | 0.0 |
| 28 | 2020-04-16 | Indonesia | 5834 | 4807.719322 | 494.241932 | 0.0 | 0.0 |
| 29 | 2020-04-17 | Indonesia | 5835 | 5071.411182 | 524.315548 | 0.0 | 0.0 |
| 30 | 2020-04-18 | Indonesia | 5836 | 5342.180859 | 555.311001 | 0.0 | 0.0 |
| 31 | 2020-04-19 | Indonesia | 5837 | 5620.028350 | 587.228288 | 0.0 | 0.0 |
| 32 | 2020-04-20 | Indonesia | 5838 | 5904.953657 | 620.067411 | 0.0 | 0.0 |
| 33 | 2020-04-21 | Indonesia | 5839 | 6196.956780 | 653.828370 | 0.0 | 0.0 |
| 34 | 2020-04-22 | Indonesia | 5840 | 6496.037717 | 688.511164 | 0.0 | 0.0 |
| 35 | 2020-04-23 | Indonesia | 5841 | 6802.196470 | 724.115793 | 0.0 | 0.0 |
| 36 | 2020-04-24 | Indonesia | 5842 | 7115.433039 | 760.642258 | 0.0 | 0.0 |
| 37 | 2020-04-25 | Indonesia | 5843 | 7435.747423 | 798.090559 | 0.0 | 0.0 |
| 38 | 2020-04-26 | Indonesia | 5844 | 7763.139622 | 836.460695 | 0.0 | 0.0 |
| 39 | 2020-04-27 | Indonesia | 5845 | 8097.609637 | 875.752666 | 0.0 | 0.0 |
| 40 | 2020-04-28 | Indonesia | 5846 | 8439.157467 | 915.966473 | 0.0 | 0.0 |
| 41 | 2020-04-29 | Indonesia | 5847 | 8787.783112 | 957.102115 | 0.0 | 0.0 |
| 42 | 2020-04-30 | Indonesia | 5848 | 9143.486573 | 999.159592 | 0.0 | 0.0 |

# Fatalities Indonesia

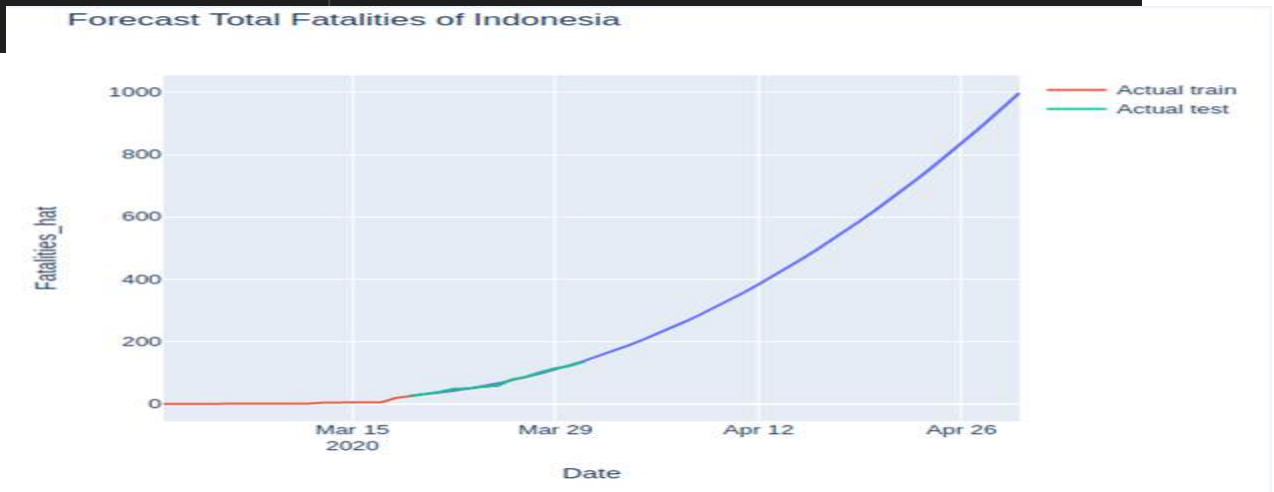
Displaying Indonesia Fatalities

```
[ ]  country = "Indonesia"
     df_val = df_val_1=0;
     df_country = df_val[df_val['Country_Region'] == country].groupby(['Date','Country_Region']).sum().reset_index()
     df_train = train[(train['Country_Region'].isin(df_country['Country_Region'].unique())) & (train['ConfirmedCases'] > 0)].groupby(['Date']).sum().reset_index()

     idx = df_country[((df_country['ConfirmedCases'].isnull() == False) & (df_country['ConfirmedCases'] > 0))].shape[0]
     fig = px.line(df_country, x="Date", y="ConfirmedCases_hat", title='Forecast Total Cases of ' + df_country['Country_Region'].values[0])
     fig.add_scatter(x=df_train['Date'], y=df_train['ConfirmedCases'], mode='lines', name="Actual train", showlegend=True)
     fig.add_scatter(x=df_country['Date'][0:idx], y=df_country['ConfirmedCases'][0:idx], mode='lines', name="Actual test", showlegend=True)
     fig.show()

     fig = px.line(df_country, x="Date", y="Fatalities_hat", title='Forecast Total Fatalities of ' + df_country['Country_Region'].values[0])
     fig.add_scatter(x=df_train['Date'], y=df_train['Fatalities'], mode='lines', name="Actual train", showlegend=True)
     fig.add_scatter(x=df_country['Date'][0:idx], y=df_country['Fatalities'][0:idx], mode='lines', name="Actual test", showlegend=True)

     fig.show()
```



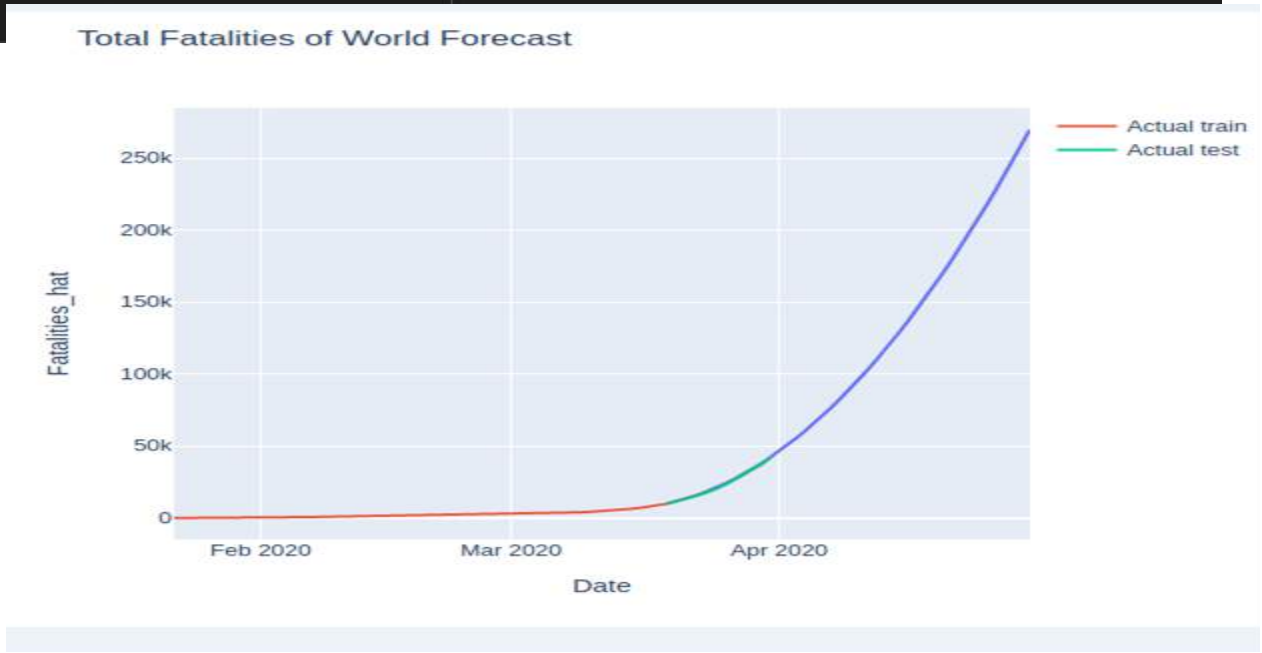
# Global Fatalities

Displaying World Total Fatalities

```
[ ]  df_total = df_val.groupby(['Date']).sum().reset_index()
     df_train = train[(train['Country_Region'].isin(df_val['Country_Region'].unique())) & (train['ConfirmedCases'] > 0)].groupby(['Date']).sum().reset_index()

     idx = df_total[((df_total['ConfirmedCases'].isnull() == False) & (df_total['ConfirmedCases'] > 0))].shape[0]
     fig = px.line(df_total, x="Date", y="ConfirmedCases_hat", title='Total Cases of World Forecast')
     fig.add_scatter(x=df_train['Date'], y=df_train['ConfirmedCases'], mode='lines', name="Actual train", showlegend=True)
     fig.add_scatter(x=df_total['Date'][0:idx], y=df_total['ConfirmedCases'][0:idx], mode='lines', name="Actual test", showlegend=True)
     fig.show()

     fig = px.line(df_total, x="Date", y="Fatalities_hat", title='Total Fatalities of World Forecast')
     fig.add_scatter(x=df_train['Date'], y=df_train['Fatalities'], mode='lines', name="Actual train", showlegend=True)
     fig.add_scatter(x=df_total['Date'][0:idx], y=df_total['Fatalities'][0:idx], mode='lines', name="Actual test", showlegend=True)
     fig.show()
```



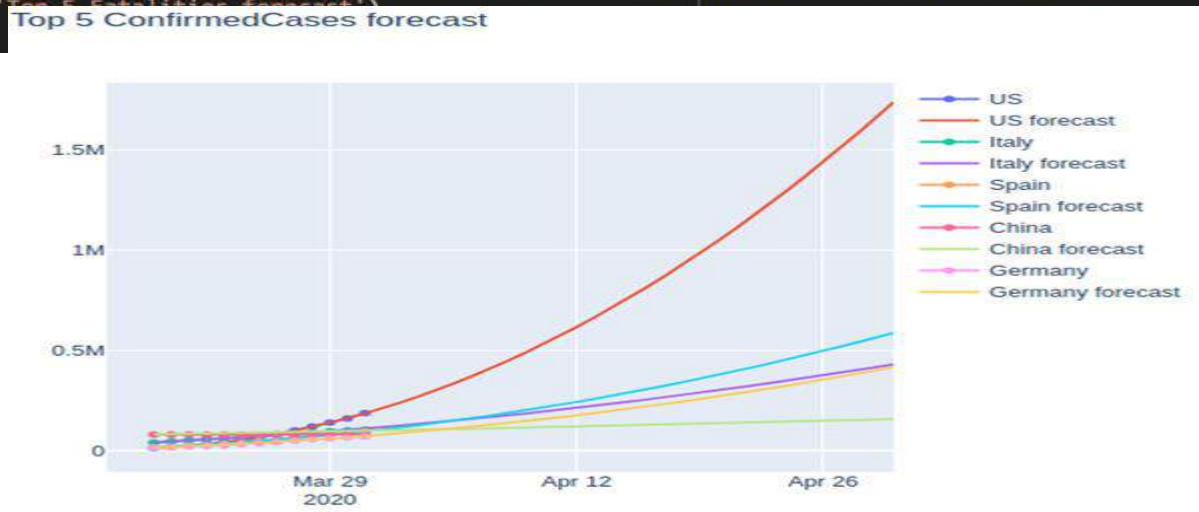
# 5 Top Confirmed Country

Displaying Top 5 Countries Confirmed & Fatalities

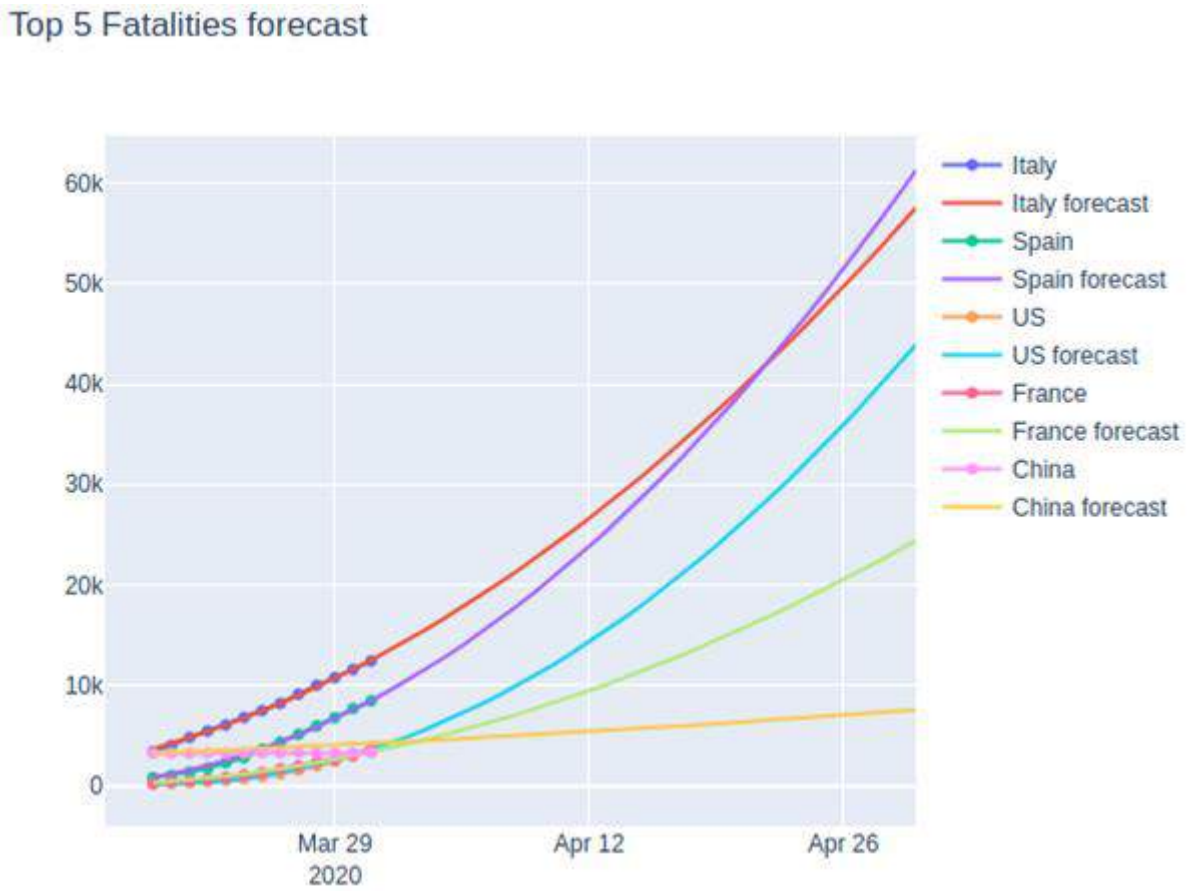
```
[ ]  df_now = train.groupby(['Date','Country_Region']).sum().sort_values(['Country_Region','Date']).reset_index()
     df_now['New Cases'] = df_now['ConfirmedCases'].diff()
     df_now['New Fatalities'] = df_now['Fatalities'].diff()
     df_now = df_now.groupby('Country_Region').apply(lambda group: group.iloc[-1:]).reset_index(drop = True)

     fig = go.Figure()
     for country in df_now.sort_values('ConfirmedCases', ascending=False).head(5)['Country_Region'].values:
         df_country = df_val[df_val['Country_Region'] == country].groupby(['Date','Country_Region']).sum().reset_index()
         idx = df_country[((df_country['ConfirmedCases'].isnull() == False) & (df_country['ConfirmedCases'] > 0))].shape[0]
         fig.add_trace(go.Scatter(x=df_country['Date'][0:idx],y= df_country['ConfirmedCases'][0:idx], name = country))
         fig.add_trace(go.Scatter(x=df_country['Date'],y= df_country['ConfirmedCases_hat'], name = country + ' forecast'))
     fig.update_layout(title_text='Top 5 ConfirmedCases forecast')
     fig.show()

     fig = go.Figure()
     for country in df_now.sort_values('Fatalities', ascending=False).head(5)['Country_Region'].values:
         df_country = df_val[df_val['Country_Region'] == country].groupby(['Date','Country_Region']).sum().reset_index()
         idx = df_country[((df_country['Fatalities'].isnull() == False) & (df_country['Fatalities'] > 0))].shape[0]
         fig.add_trace(go.Scatter(x=df_country['Date'][0:idx],y= df_country['Fatalities'][0:idx], name = country))
         fig.add_trace(go.Scatter(x=df_country['Date'],y= df_country['Fatalities_hat'], name = country + ' forecast'))
     fig.update_layout(title_text='
     fig.show()
```



# 5 Top Fatalities Country



# Top 10 Country

Top 10 Countries

```
[ ]  df_now = df_now.sort_values('ConfirmedCases', ascending = False)
     fig = make_subplots(rows = 2, cols = 2)
     fig.add_bar(x=df_now['Country_Region'].head(10), y = df_now['ConfirmedCases'].head(10), row=1, col=1, name = 'Total cases')
     df_now = df_now.sort_values('Fatalities', ascending=False)
     fig.add_bar(x=df_now['Country_Region'].head(10), y = df_now['Fatalities'].head(10), row=1, col=2, name = 'Total Fatalities')


     df_now = df_now.sort_values('New Cases', ascending=False)
     fig.add_bar(x=df_now['Country_Region'].head(10), y = df_now['New Cases'].head(10), row=2, col=1, name = 'New Cases')
     df_now = df_now.sort_values('New Fatalities', ascending=False)
     fig.add_bar(x=df_now['Country_Region'].head(10), y = df_now['New Fatalities'].head(10), row=2, col=2, name = 'New Fatalities')

     fig.update_layout({'title_text':'Top 10 Country','legend_orientation':'h','legend_y':1.1,'legend_yanchor':'auto'})
```

# Referensi


- https://www.geeksforgeeks.org/python-implementation-of-polynomial-regression/#:~:text=Polynomial%20Regression%20is%20a%20form,denoted%20E(y%20%7Cx)
- https://id.wikipedia.org/wiki/Regresi_linear
- https://www.kaggle.com/binhlc/sars-cov-2-exponential-model-week-2
- https://www.kaggle.com/c/covid19-global-forecasting-week-2/data
- https://medium.com/@jrendz/regresi-linier-dengan-r-dan-python-ebb80662c6da
- https://www.megabagus.id/polynomial-regression/

TERIMAKASIH