In [1]:

```
import pandas as pd
```

In [3]:

```
dane = pd.read_csv('C:/Users/Laptop/Desktop/Grypa/Dane/total.csv', index_col=['data'])
dane = dane['Total']
dane_test_train = dane[(dane.index > '2012-08-23' ) & (dane.index < '2020-03-16')]

train_87 = dane_test_train[0:314]
series_train_87 = train_87.values
test_13 = dane_test_train[314:]
series_test_13 = test_13.values
```

In [6]:

```
import warnings
from pandas import read_csv
from pandas import datetime
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

# evaluate an ARIMA model for a given order (p,d,q)
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    #train_size = int(len(X) * 0.66)
    train, test = train_87.values, test_13.values
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    return error

# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(dataset, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                    print('ARIMA%s MSE=%.3f' % (order,mse))
                except:
                    continue
    print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))

# load dataset
def parser(x):
    return datetime.strptime('190'+x, '%Y-%m')
series = dane_test_train
# evaluate parameters
p_values = [0, 1, 2, 3, 4, 5, 6, 7, 8]
d_values = [0, 1, 2]
q_values = [0, 1, 2]
warnings.filterwarnings("ignore")
evaluate_models(series.values, p_values, d_values, q_values)
```

```
ARIMA(0, 0, 1) MSE=1532472872.249
ARIMA(0, 1, 1) MSE=509835744.879
ARIMA(0, 1, 2) MSE=463494723.895
ARIMA(0, 2, 1) MSE=487645440.000
ARIMA(1, 0, 0) MSE=466649883.604
ARIMA(1, 0, 1) MSE=492158162.311
ARIMA(1, 0, 2) MSE=429484659.125
ARIMA(1, 1, 0) MSE=519946799.932
ARIMA(1, 1, 1) MSE=508918160.901
ARIMA(1, 1, 2) MSE=453740346.794
ARIMA(1, 2, 0) MSE=561221860.103
ARIMA(1, 2, 1) MSE=524690023.963
ARIMA(2, 0, 0) MSE=503853469.211
ARIMA(2, 0, 1) MSE=471725203.735
ARIMA(2, 0, 2) MSE=445062962.735
ARIMA(2, 1, 0) MSE=460502778.085
ARIMA(2, 2, 0) MSE=567876753.895
ARIMA(3, 0, 0) MSE=421933666.969
ARIMA(3, 0, 2) MSE=397444542.522
ARIMA(3, 1, 0) MSE=441784260.403
ARIMA(3, 2, 0) MSE=504819628.835
ARIMA(4, 0, 0) MSE=416164380.667
ARIMA(4, 0, 1) MSE=389985553.019
ARIMA(4, 0, 2) MSE=394018348.558
ARIMA(4, 1, 0) MSE=448406438.100
ARIMA(4, 2, 0) MSE=496561947.502
ARIMA(4, 2, 1) MSE=451780026.973
ARIMA(5, 0, 0) MSE=409471041.081
ARIMA(5, 0, 2) MSE=387621275.081
ARIMA(5, 1, 0) MSE=457663286.714
ARIMA(5, 2, 0) MSE=592216170.959
ARIMA(6, 0, 0) MSE=415708391.039
ARIMA(6, 0, 2) MSE=389035136.877
ARIMA(6, 1, 0) MSE=494569260.673
ARIMA(6, 2, 0) MSE=590717986.849
ARIMA(7, 0, 0) MSE=463757642.072
ARIMA(7, 0, 2) MSE=405287885.672
ARIMA(7, 1, 0) MSE=498298574.823
ARIMA(7, 2, 0) MSE=558077340.976
ARIMA(7, 2, 1) MSE=564321577.554
ARIMA(8, 0, 0) MSE=468103518.633
ARIMA(8, 1, 0) MSE=495301185.584
ARIMA(8, 2, 0) MSE=567490538.603
Best ARIMA(5, 0, 2) MSE=387621275.081
```