Team members: Alvin Wang / Dayou Li / Tianjian Che / Xiaoyang Chi

Import Data

Raw Data - US CPI and commodity

```
import numpy as np
import pandas as pd
from pylab import mpl, plt
import datetime as dt
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'
% matplotlib inline
```

```
# 2022.05.01
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

```
# 2022.05.01
import os
os.chdir("/content/gdrive/MyDrive/Colab Shared Files")
```

```
inflation = pd.read_csv("US CPI.csv",index_col='Yearmon')
inflation.index.name = 'Date'
inflation.index = [dt.datetime.strptime(i[-4:]+'-'+i[3:5],"%Y-%m") for i in inflation.index]
inflation = inflation.loc['2000-01':]
inflation.plot(figsize=(10,6),marker='.');
```


```
inflation['year-on-year growth rate'] = (inflation['CPI'] - inflation['CPI'].shift(12))/infla
inflation = inflation.loc['2009-01-01':]
inflation
```

	CPI	year-on-year growth rate
2009-01-01	211.143	0.000298
2009-02-01	212.193	0.002362
2009-03-01	212.709	-0.003836
2009-04-01	213.240	-0.007369
2009-05-01	213.856	-0.012814
...	...	...
2021-03-01	264.877	0.026198
2021-04-01	267.054	0.041597
2021-05-01	269.195	0.049927
2021-06-01	271.696	0.053915

```
inflation['year-on-year growth rate'].plot(figsize=(10,6), marker='.')
plt.title('year-on-year growth rate of CPI');
```

```
commodity = pd.read_csv("commodity.csv",index_col='Date')
commodity.index = [dt.datetime.strptime(i,"%Y-%m-%d") for i in commodity.index]
commodity['Symbol'].unique()
```

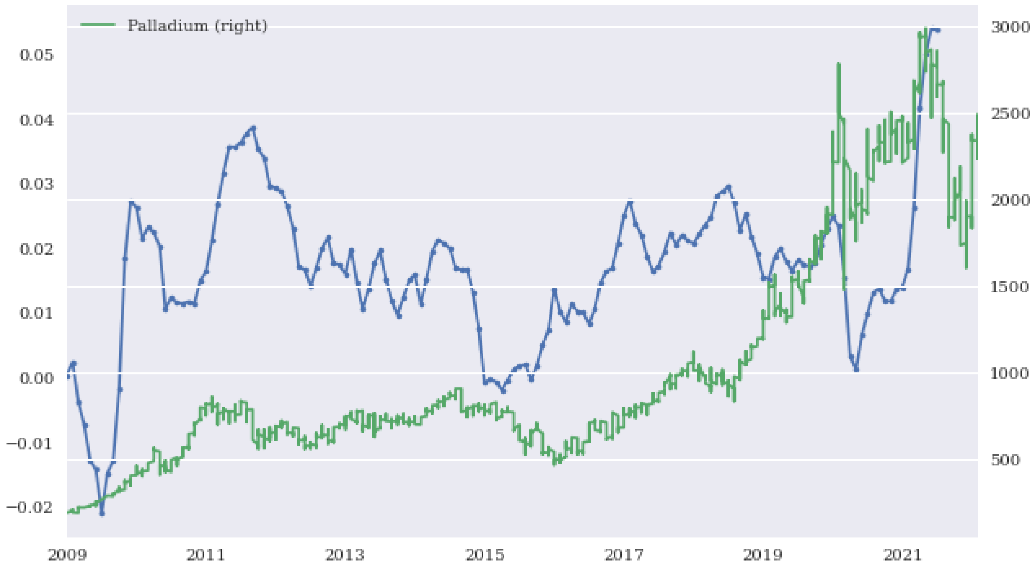
```
array(['Gold', 'Palladium', 'Nickel', 'Brent Oil', 'Natural Gas',
       'US Wheat'], dtype=object)
```

```
gold = commodity[commodity['Symbol'] == 'Gold']
palladium = commodity[commodity['Symbol'] == 'Palladium']
nickel = commodity[commodity['Symbol'] == 'Nickel']
brent = commodity[commodity['Symbol'] == 'Brent Oil']
ng = commodity[commodity['Symbol'] == 'Natural Gas']
wheat= commodity[commodity['Symbol'] == 'US Wheat']
```

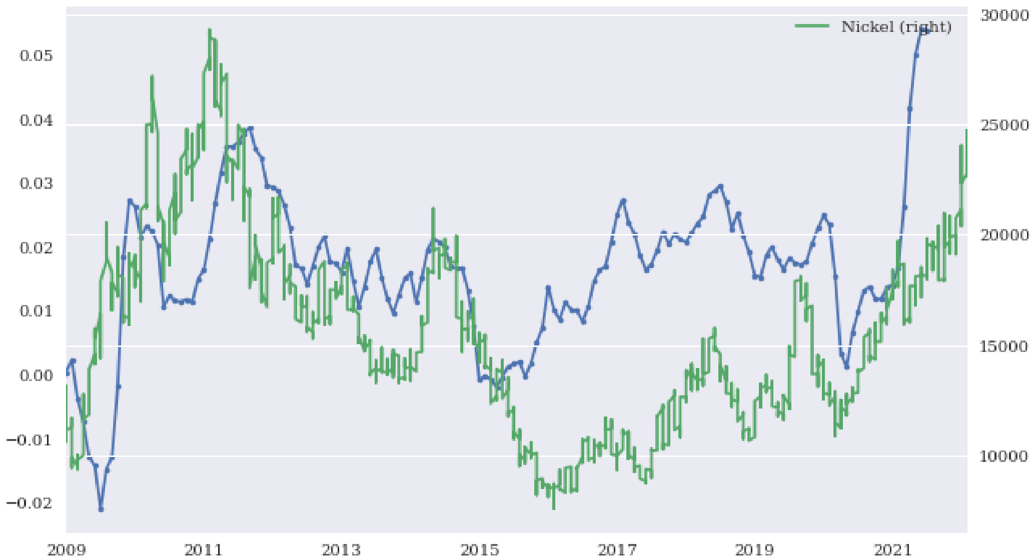
```
gold = gold.loc['2009-01-01':'2022-02-28']
palladium = palladium.loc['2009-01-01':'2022-02-28']
nickel = nickel.loc['2009-01-01':'2022-02-28']
brent = brent.loc['2009-01-01':'2022-02-28']
ng = ng.loc['2009-01-01':'2022-02-28']
wheat = wheat.loc['2009-01-01':'2022-02-28']
commodity = pd.concat([gold,palladium,nickel,brent,ng,wheat],axis=0)
```

```
plt.figure(figsize=(10,6))
inflation['year-on-year growth rate'].plot(marker='.',label='CPI')
gold['Close'].plot(secondary_y=True,label='Gold')
plt.legend(loc=0);
```


```
plt.figure(figsize=(10,6))
inflation['year-on-year growth rate'].plot(marker='.',label='CPI')
palladium['Close'].plot(secondary_y=True,label='Palladium')
plt.legend(loc=0);
```

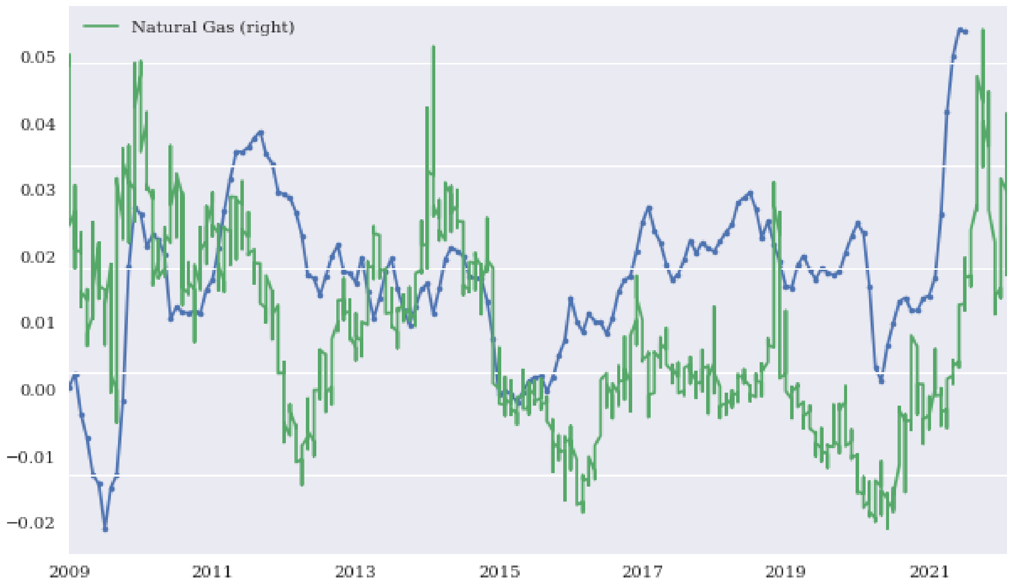



```
plt.figure(figsize=(10,6))
inflation['year-on-year growth rate'].plot(marker='.',label='CPI')
nickel['Close'].plot(secondary_y=True,label='Nickel')
plt.legend(loc=0);
```




```
plt.figure(figsize=(10,6))
inflation['year-on-year growth rate'].plot(marker='.',label='CPI')
brent['Close'].plot(secondary_y=True,label='Brent oil')
plt.legend(loc=0);
```


```
plt.figure(figsize=(10,6))
inflation['year-on-year growth rate'].plot(marker='.',label='CPI')
ng['Close'].plot(secondary_y=True,label='Natural Gas')
plt.legend(loc=0);
```



Mixed-Frequency DataFrame

```
inflation
```

	CPI	year-on-year growth rate
2009-01-01	211.143	0.000298
2009-02-01	212.193	0.002362
2009-03-01	212.709	-0.003836
2009-04-01	213.240	-0.007369
2009-05-01	213.856	-0.012814
...	...	...
2021-03-01	264.877	0.026198
2021-04-01	267.054	0.041597

```
for  d  in  [gold,palladium,nickel,brent,ng,wheat]:
    symbol  =  d.Symbol[0]
    d.columns  =  [symbol+'-'+i  for  i  in  d.columns]


commodity    =  pd.concat([
                                               gold.iloc[:,1:],
                                               palladium.iloc[:,1:],
                                               nickel.iloc[:,1:],
                                               brent.iloc[:,1:],
                                               ng.iloc[:,1:],
                                               wheat.iloc[:,1:]],axis=1)


commodity.fillna(method="ffill",inplace=True)
commodity.head()
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-Volume	Palladium-Open	Palladium-High
2009-01-02	881.5	881.5	868.9	878.8	46.0	NaN	NaN
2009-01-05	882.0	883.5	847.0	857.2	35.0	NaN	NaN
2009-01-06	855.1	867.6	840.0	865.4	113.0	190.0	192.0
2009-01-07	862.0	867.0	837.7	841.1	101.0	190.0	192.0
2009-	837.9	861.0	837.9	853.9	255.0	190.0	192.0

```
commodity.fillna(method="bfill",inplace=True)
commodity
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-Volume	Palladium-Open	Palladi H
2009-01-02	881.5	881.5	868.9	878.8	46.0	190.00	192
2009-01-05	882.0	883.5	847.0	857.2	35.0	190.00	192
2009-01-06	855.1	867.6	840.0	865.4	113.0	190.00	192
2009-01-07	862.0	867.0	837.7	841.1	101.0	190.00	192
2009-01-08	837.9	861.0	837.9	853.9	255.0	190.00	192
...	...	...	...	...	...	...	
2022-02-23	1901.2	1912.9	1891.1	1910.4	154843.0	2345.50	2485
2022-02-24	1911.9	1976.5	1878.6	1926.3	423048.0	2482.52	2712
2022-02-25	1906.5	1925.0	1884.4	1887.6	229780.0	2425.00	2482
2022-02-27	1906.5	1925.0	1884.4	1887.6	229780.0	2356.00	2553

```
commodity.isna().sum()
```

```
Gold-Open            0
Gold-High            0
Gold-Low             0
Gold-Close           0
Gold-Volume          0
Palladium-Open       0
Palladium-High       0
Palladium-Low        0
Palladium-Close      0
Palladium-Volume     0
Nickel-Open          0
Nickel-High          0
Nickel-Low           0
Nickel-Close         0
Nickel-Volume        0
Brent Oil-Open       0
Brent Oil-High       0
Brent Oil-Low        0
Brent Oil-Close      0
```

```
Brent Oil-Volume       0
Natural Gas-Open       0
Natural Gas-High       0
Natural Gas-Low        0
Natural Gas-Close      0
Natural Gas-Volume     0
US Wheat-Open          0
US Wheat-High          0
US Wheat-Low           0
US Wheat-Close         0
US Wheat-Volume        0
dtype: int64
```

Forecasting

First Attempt: Simple Average and OLS

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression


seg1 = '2016-12'
seg2 = '2017-01'


X = commodity.groupby(pd.PeriodIndex(commodity.index, freq="M")).mean()
X
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-V
2009-01	859.952381	868.404762	850.100000	860.500000	3725.5
2009-02	940.805000	952.760000	930.255000	942.225000	677.4
2009-03	926.672727	935.540909	916.804545	925.295455	3246.5
[illegible]	1793.688889	1802.974074	1786.355556	1795.774074	4527.5

```
X_train = X.loc['2009-01':seg1]
X_test = X.loc[seg2:]
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
scaler.fit(X_test)
X_test = scaler.transform(X_test)
```

```
y_train = inflation['year-on-year growth rate'].iloc[1:len(X_train)+1]
y_test = inflation['year-on-year growth rate'].iloc[len(X_train)+1:]
```

```
model = LinearRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_train)
```

```
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

```
MSE: 2.9104948516751224e-05
```

```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test, y_tpred)).mean())
```

```
MSE: 0.00017457832179893843
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```

Second Attempt: Simple Average and Penalized Regression

Lasso

```
from sklearn.linear_model import Lasso
```

```
model = Lasso(alpha=0.001)
model.fit(X_train, y_train)
y_predict = model.predict(X_train)
```

```
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

    MSE: 5.0932319589503475e-05

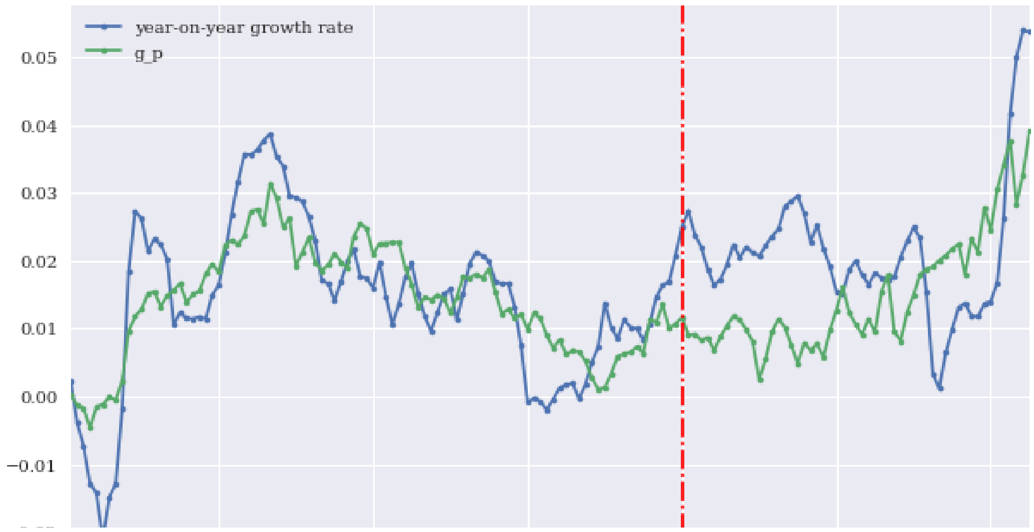
```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test, y_tpred)).mean())
```

    MSE: 0.00016667625435500033

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



LassoCV

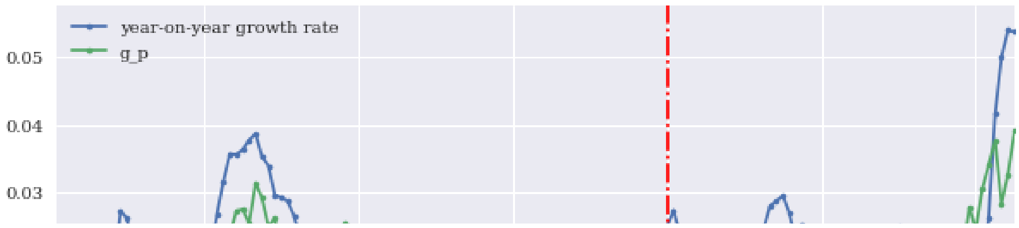
```
from sklearn.linear_model import LassoCV

model = LassoCV(alphas=[1e-4,1e-3,1e-2,1e-1])
model.fit(X_train, y_train)
y_predict = model.predict(X_train)

print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

```
MSE: 5.0932319589503475e-05
```

```
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

```
MSE: 0.0001666762543550033
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p

inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



Ridge

```
from sklearn.linear_model import Ridge
```

```
model = Ridge(alpha=0.001)
model.fit(X_train, y_train)
y_predict = model.predict(X_train)
```

```
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

    MSE: 3.0148834462553933e-05

```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

    MSE: 0.0001678178406496832

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```

RidgeCV

```
from sklearn.linear_model import RidgeCV
```

```
model = RidgeCV(alphas=[1e-4,1e-3,1e-2,1e-1])
model.fit(X_train, y_train)
y_predict = model.predict(X_train)
y_tpred = model.predict(X_test)
```

```
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

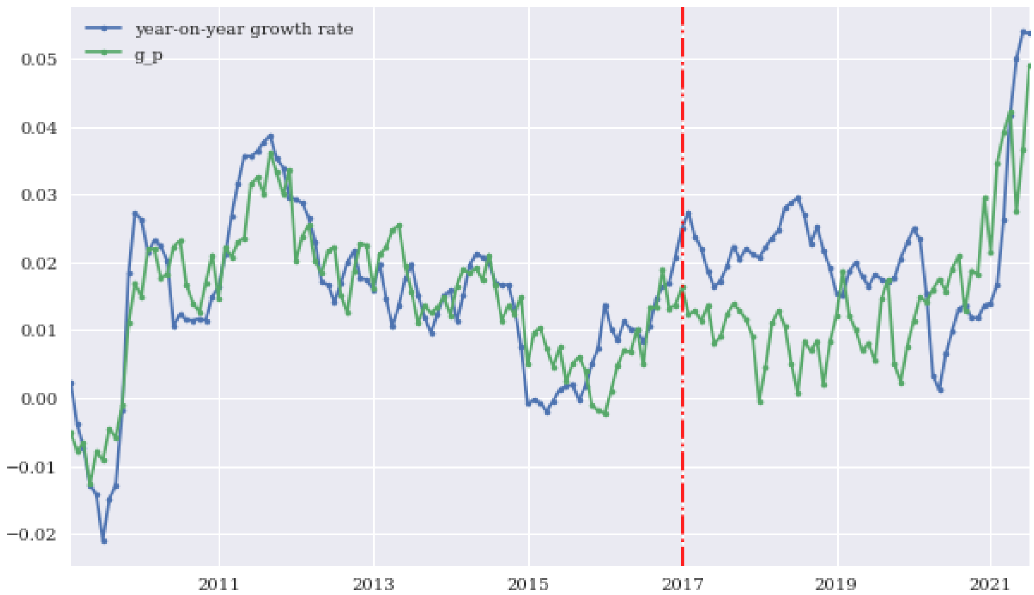
```
MSE: 3.68902460248577e-05
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

```
MSE: 0.0001681093680231343
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



Decision Tree

```
from sklearn.tree import DecisionTreeRegressor, export_text, plot_tree
```

```
model = DecisionTreeRegressor(max_depth=8, random_state=123)
model.fit(X_train, y_train)
```

    DecisionTreeRegressor(max_depth=8, random_state=123)

```
y_predict = model.predict(X_train)
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

    MSE: 1.8067843114297657e-07

```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

    MSE: 0.00017954714155981566

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```


```
print(export_text(model, feature_names=list(X.columns)))
```

```
    |   |   |   |   |   |   |   |--- value: [0.01]
    |   |   |   |   |   |   |--- US Wheat-Low >  -0.11
    |   |   |   |   |   |   |   |--- Gold-Low <= -0.06
    |   |   |   |   |   |   |   |   |--- value: [0.02]
```

```
|   |   |   |   |   |   |   |--- Gold-Low >  -0.06
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |--- Palladium-Open >  0.93
|   |   |   |   |--- US Wheat-Close <= 1.79
|   |   |   |   |   |--- Natural Gas-Open <= 0.82
|   |   |   |   |   |   |--- US Wheat-Low <= -0.45
|   |   |   |   |   |   |   |--- Natural Gas-High <= 0.36
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |   |--- Natural Gas-High >  0.36
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |--- US Wheat-Low >  -0.45
|   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |--- Natural Gas-Open >  0.82
|   |   |   |   |   |   |--- US Wheat-Close <= 0.30
|   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |--- US Wheat-Close >  0.30
|   |   |   |   |   |   |   |--- Gold-Open <= -0.17
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |   |--- Gold-Open >  -0.17
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |--- US Wheat-Close >  1.79
|   |   |   |   |   |--- value: [0.03]
|   |--- Gold-Open >  0.39
|   |   |--- Nickel-Low <= 0.39
|   |   |   |--- Gold-Volume <= 1.12
|   |   |   |   |--- Brent Oil-Volume <= 0.14
|   |   |   |   |   |--- US Wheat-Open <= 2.18
|   |   |   |   |   |   |--- Palladium-Close <= 0.46
|   |   |   |   |   |   |   |--- Palladium-Close <= 0.30
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |   |--- Palladium-Close >  0.30
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |--- Palladium-Close >  0.46
|   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |--- US Wheat-Open >  2.18
|   |   |   |   |   |   |--- Palladium-Volume <= -0.51
|   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |--- Palladium-Volume >  -0.51
|   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |--- Brent Oil-Volume >  0.14
|   |   |   |   |   |--- Brent Oil-Volume <= 0.16
|   |   |   |   |   |   |--- value: [0.01]
|   |   |   |   |   |--- Brent Oil-Volume >  0.16
|   |   |   |   |   |   |--- Gold-Volume <= -0.09
|   |   |   |   |   |   |   |--- Gold-High <= 1.27
|   |   |   |   |   |   |   |   |--- value: [0.01]
|   |   |   |   |   |   |   |--- Gold-High >  1.27
|   |   |   |   |   |   |   |   |--- value: [0.01]
|   |   |   |   |   |   |--- Gold-Volume >  -0.09
|   |   |   |   |   |   |   |--- US Wheat-Low <= 1.17
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |   |   |   |   |--- US Wheat-Low >  1.17
|   |   |   |   |   |   |   |   |--- value: [0.02]
|   |   |   |--- Gold-Volume >  1.12
|   |   |   |   |--- value: [0.0[illegible]]
```

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
```

```
model = RandomForestRegressor(n_estimators=500, max_features=int(X_train.shape[1]/3))
model.fit(X_train, y_train)
```

    RandomForestRegressor(max_features=10, n_estimators=500)

```
y_predict = model.predict(X_train)
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

    MSE: 3.879500618664648e-06

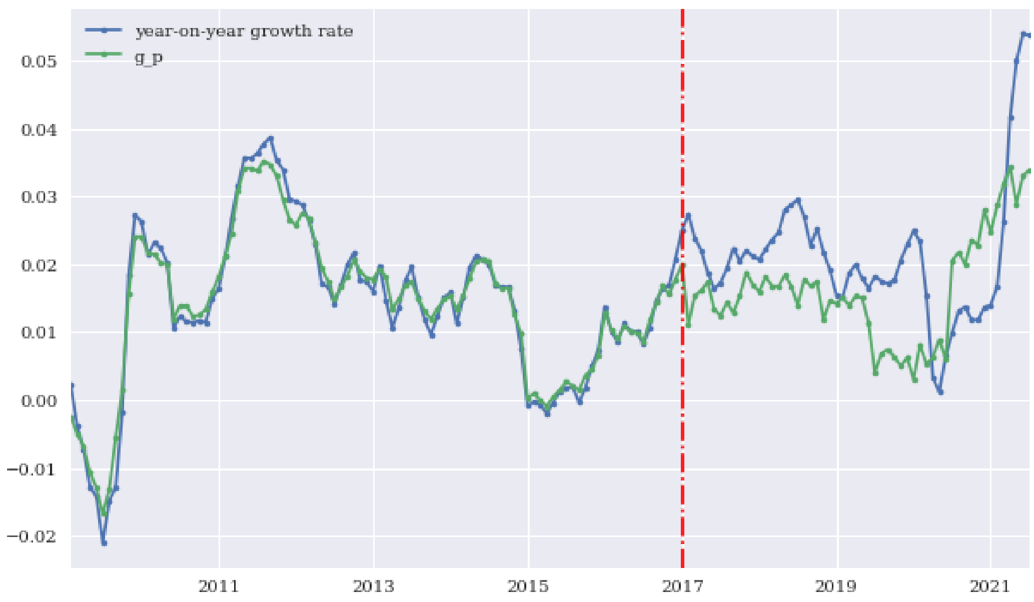
```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

    MSE: 0.00010771376556375286

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



XGBoost

```
import xgboost as xgb
```

```
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=300,max_depth=6,subsample=0.
```

```
model.fit(X_train,y_train)
```

```
XGBRegressor(colsample_bytree=0.8, max_depth=6, n_estimators=300,
             objective='reg:squarederror', subsample=0.6)
```

```
y_predict = model.predict(X_train)
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

```
MSE: 2.0120924594331378e-07
```

```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

```
MSE: 0.00011300860375473602
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```

MultiLayer Perceptron

```
from sklearn.neural_network import MLPRegressor

model = MLPRegressor(solver='lbfgs',hidden_layer_sizes=(10,), random_state=123,max_iter=10000)


model.fit(X_train,y_train)
```

```
MLPRegressor(hidden_layer_sizes=(10,), max_iter=10000, random_state=123,
             solver='lbfgs')
```

```
y_predict = model.predict(X_train)
print('MSE:', np.square(np.subtract(y_train,y_predict)).mean())
```

```
MSE: 2.383843843916805e-05
```

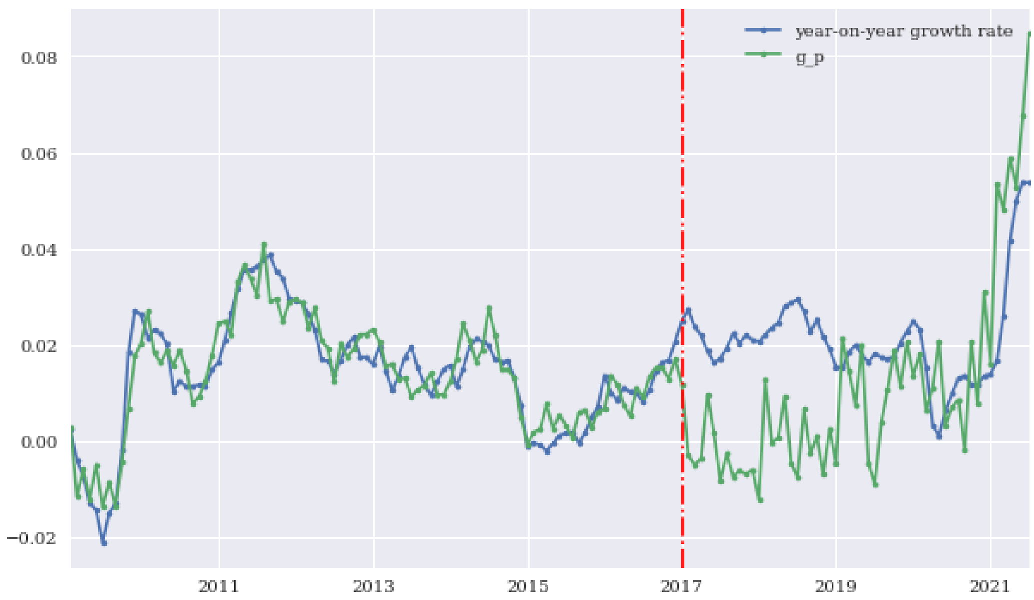
```
y_tpred = model.predict(X_test)
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

```
MSE: 0.00039807634526450175
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



LSTM

```python
import os
import random as rn
import tensorflow as tf
from tensorflow.keras import regularizers
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, LSTM, Activation, BatchNormalization
from keras.utils.np_utils import to_categorical


def set_my_seed():
    os.environ['PYTHONHASHSEED']='0'
    np.random.seed(1)
    rn.seed(12345)
    tf.random.set_seed(123)


set_my_seed()


X_train1 = np.reshape(X_train,(X_train.shape[0],1,X_train.shape[1]))
X_test1 = np.reshape(X_test,(X_test.shape[0],1,X_test.shape[1]))


model = Sequential()
model.add(LSTM(128,input_shape=(1,30)))
model.add(Dense(1))
model.add(Dropout(0.1))
model.add(Activation('tanh'))
model.compile(loss='mse',optimizer='adam')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               81408

 dense (Dense)               (None, 1)                 129

 dropout (Dropout)           (None, 1)                 0

 activation (Activation)     (None, 1)                 0

=================================================================
Total params: 81,537
Trainable params: 81,537
Non-trainable params: 0
_________________________________________________________________
```

```python
model.fit(X_train1,y_train,epochs=100,shuffle=False,verbose=2,batch_size=1)
```

```
Epoch 60/100
96/96 - 0s - loss: 7.9712e-05 - 216ms/epoch - 2ms/step
Epoch 61/100
96/96 - 0s - loss: 1.0534e-04 - 212ms/epoch - 2ms/step
Epoch 62/100
96/96 - 0s - loss: 1.1720e-04 - 199ms/epoch - 2ms/step
```

```
Epoch 63/100
96/96 - 0s - loss: 1.6353e-04 - 215ms/epoch - 2ms/step
Epoch 64/100
96/96 - 0s - loss: 9.7084e-05 - 213ms/epoch - 2ms/step
Epoch 65/100
96/96 - 0s - loss: 1.3059e-04 - 204ms/epoch - 2ms/step
Epoch 66/100
96/96 - 0s - loss: 1.0995e-04 - 212ms/epoch - 2ms/step
Epoch 67/100
96/96 - 0s - loss: 9.5347e-05 - 208ms/epoch - 2ms/step
Epoch 68/100
96/96 - 0s - loss: 8.2335e-05 - 217ms/epoch - 2ms/step
Epoch 69/100
96/96 - 0s - loss: 7.9245e-05 - 212ms/epoch - 2ms/step
Epoch 70/100
96/96 - 0s - loss: 8.9683e-05 - 214ms/epoch - 2ms/step
Epoch 71/100
96/96 - 0s - loss: 7.3396e-05 - 207ms/epoch - 2ms/step
Epoch 72/100
96/96 - 0s - loss: 1.3886e-04 - 209ms/epoch - 2ms/step
Epoch 73/100
96/96 - 0s - loss: 5.7962e-05 - 215ms/epoch - 2ms/step
Epoch 74/100
96/96 - 0s - loss: 1.0722e-04 - 206ms/epoch - 2ms/step
Epoch 75/100
96/96 - 0s - loss: 9.9224e-05 - 203ms/epoch - 2ms/step
Epoch 76/100
96/96 - 0s - loss: 1.1950e-04 - 225ms/epoch - 2ms/step
Epoch 77/100
96/96 - 0s - loss: 1.2760e-04 - 232ms/epoch - 2ms/step
Epoch 78/100
96/96 - 0s - loss: 9.3308e-05 - 241ms/epoch - 3ms/step
Epoch 79/100
96/96 - 0s - loss: 9.4987e-05 - 210ms/epoch - 2ms/step
Epoch 80/100
96/96 - 0s - loss: 1.3281e-04 - 206ms/epoch - 2ms/step
Epoch 81/100
96/96 - 0s - loss: 7.0694e-05 - 210ms/epoch - 2ms/step
Epoch 82/100
96/96 - 0s - loss: 8.1724e-05 - 219ms/epoch - 2ms/step
Epoch 83/100
96/96 - 0s - loss: 1.1215e-04 - 216ms/epoch - 2ms/step
Epoch 84/100
96/96 - 0s - loss: 9.8251e-05 - 218ms/epoch - 2ms/step
Epoch 85/100
96/96 - 0s - loss: 9.5162e-05 - 207ms/epoch - 2ms/step
Epoch 86/100
96/96 - 0s - loss: 1.2553e-04 - 215ms/epoch - 2ms/step
Epoch 87/100
96/96 - 0s - loss: 8.5607e-05 - 217ms/epoch - 2ms/step
Epoch 88/100
96/96 - 0s - loss: 5.6651e-05 - 212ms/epoch - 2ms/step
Epoch 89/100
```

```
y_predict  =  np.squeeze(model.predict(X_train1))
print('MSE:',  np.square(np.subtract(y_train,y_predict)).mean())
```

```
MSE: 6.365179970335274e-05
```

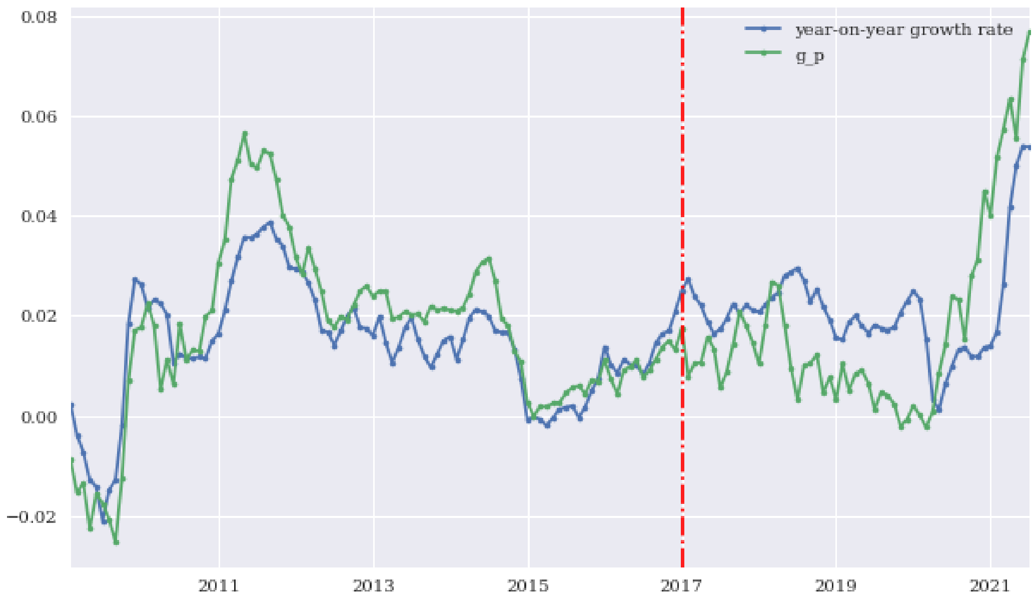
```
y_tpred = np.squeeze(model.predict(X_test1))
```

```
y_tpred = y_tpred[-y_test.shape[0]:] # 2022.05.01
print('MSE:', np.square(np.subtract(y_test,y_tpred)).mean())
```

```
MSE: 0.00025180806669138154
```

```
y_p = np.concatenate([[np.nan],y_predict, y_tpred])
inflation['g_p'] = y_p
```

```
inflation[['year-on-year growth rate','g_p']].loc['2009-02-01':].plot(figsize=(10,6),marker='.')
plt.axvline(seg2, color='r', linestyle='-.');
```



Conclusion

LSTM has the lowest MSE

Nowcasting

Average and OLS

SMA

```
Xn = commodity.rolling(22).mean()
Xn.dropna(inplace=True)
Xn
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-V
2009-02-02	863.090909	871.159091	852.468182	862.600000	3702.5
2009-02-03	864.077273	872.604545	853.418182	863.200000	3766.4
2009-02-04	864.900000	873.772727	855.690909	865.218182	3786.0
2009-02-05	867.227273	876.354545	858.618182	867.409091	3815.8
2009-02-06	869.613636	878.695455	861.881818	870.718182	3826.4
...	...	...	...	...	
2022-02-23	1842.409091	1856.102273	1832.772727	1848.143182	171331.7
2022-02-24	1847.513636	1864.111364	1837.227273	1854.493182	180690.8
2022-02-25	1852.681818	1869.752273	1841.709091	1858.638636	184798.0
2022-	1857.572727	1874.993182	1845.718182	1862.552273	189384.4

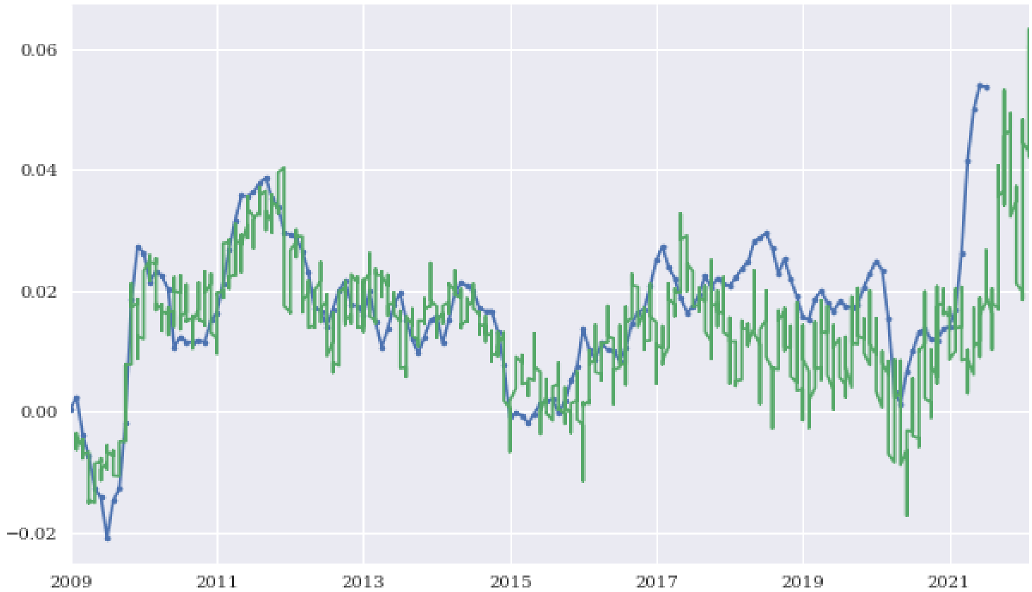
```
Xn_train = Xn.loc['2009-01':seg1]
Xn_test = Xn.loc[seg2:]

scaler = StandardScaler()
scaler.fit(Xn_train)
Xn_train = scaler.transform(Xn_train)
scaler.fit(Xn_test)
Xn_test = scaler.transform(Xn_test)

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
nowcasting_p = model.predict(Xn_train)
nowcasting_tp = model.predict(Xn_test)
```

```
nowcasting = np.concatenate([nowcasting_p, nowcasting_tp])
Xn['nowcasting'] = nowcasting
```

```
inflation['year-on-year growth rate'].plot(figsize=(10,6), marker='.')
Xn['nowcasting'].plot(figsize=(10,6));
```



EMA

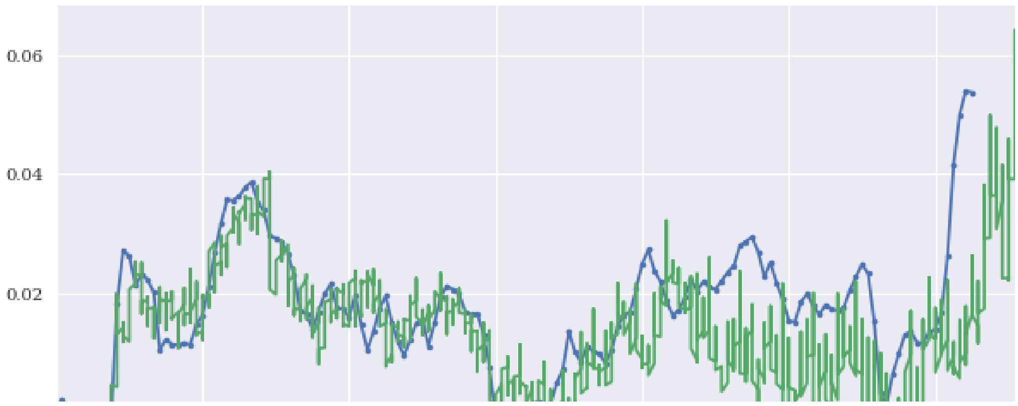
```
Xn = commodity.ewm(alpha=0.06,adjust=True,min_periods=22).mean()
Xn.dropna(inplace=True)
Xn
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-V
2009-02-02	870.805558	879.625465	860.724307	871.778602	5843.2
2009-02-03	873.366240	882.287335	863.022652	873.377043	5496.0
2009-02-04	875.439965	884.374937	865.658258	875.566283	5105.7
2009-02-05	877.792431	887.426055	868.611548	878.465600	4775.2
2009-02-06	880.545965	889.756995	871.678702	881.123631	4442.1
...	...	...	...	...	
2022-02-24	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
Xn_train = Xn.loc['2009-01':seg1]
Xn_test = Xn.loc[seg2:]

scaler = StandardScaler()
scaler.fit(Xn_train)
Xn_train = scaler.transform(Xn_train)
scaler.fit(Xn_test)
Xn_test = scaler.transform(Xn_test)


model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
nowcasting_p = model.predict(Xn_train)
nowcasting_tp = model.predict(Xn_test)


nowcasting = np.concatenate([nowcasting_p, nowcasting_tp])
Xn['nowcasting'] = nowcasting


inflation['year-on-year growth rate'].plot(figsize=(10,6), marker='.')
Xn['nowcasting'].plot(figsize=(10,6));
```



LSTM

LSTM Regression

```
model  =  Sequential()
model.add(LSTM(128,input_shape=(1,30)))
model.add(Dense(1))
model.add(Dropout(0.1))
model.add(Activation('tanh'))
model.compile(loss='mse',optimizer='adam')
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 128)               81408

 dense_1 (Dense)             (None, 1)                 129

 dropout_1 (Dropout)         (None, 1)                 0

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 81,537
Trainable params: 81,537
Non-trainable params: 0
_________________________________________________________________
```

```
set_my_seed()
```

```
model.fit(X_train1,y_train,epochs=100,shuffle=False,verbose=2,batch_size=1)
```

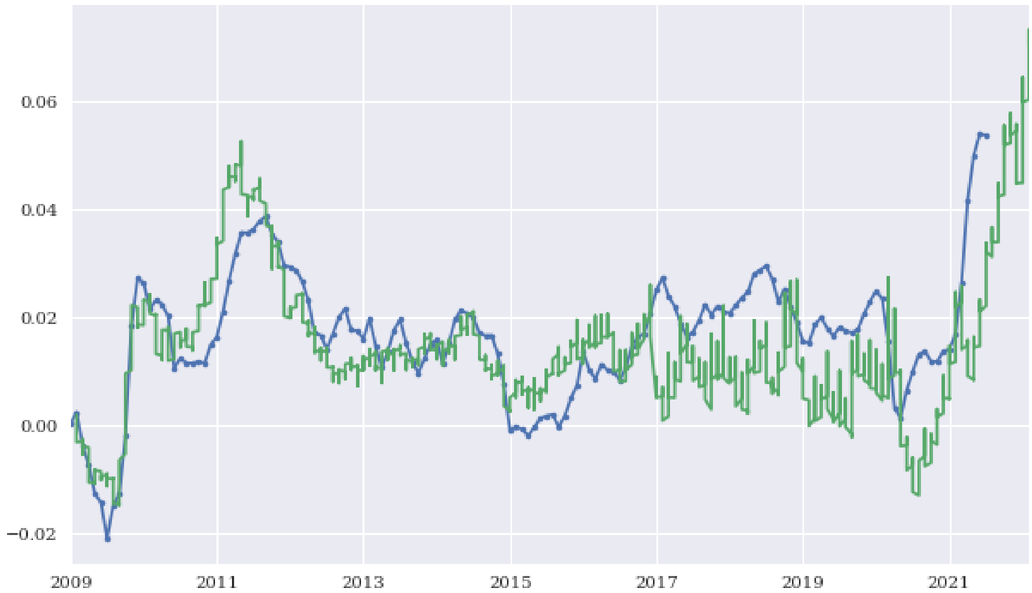
```
Epoch 60/100
96/96 - 0s - loss: 4.4707e-05 - 206ms/epoch - 2ms/step
Epoch 61/100
96/96 - 0s - loss: 6.2710e-05 - 196ms/epoch - 2ms/step
Epoch 62/100
96/96 - 0s - loss: 6.3311e-05 - 212ms/epoch - 2ms/step
Epoch 63/100
```

```
Epoch 63/100
96/96 - 0s - loss: 1.0005e-04 - 210ms/epoch - 2ms/step
Epoch 64/100
96/96 - 0s - loss: 8.6898e-05 - 203ms/epoch - 2ms/step
Epoch 65/100
96/96 - 0s - loss: 1.8295e-04 - 200ms/epoch - 2ms/step
Epoch 66/100
96/96 - 0s - loss: 1.8520e-04 - 210ms/epoch - 2ms/step
Epoch 67/100
96/96 - 0s - loss: 1.1860e-04 - 211ms/epoch - 2ms/step
Epoch 68/100
96/96 - 0s - loss: 1.4626e-04 - 205ms/epoch - 2ms/step
Epoch 69/100
96/96 - 0s - loss: 1.5099e-04 - 219ms/epoch - 2ms/step
Epoch 70/100
96/96 - 0s - loss: 9.4913e-05 - 204ms/epoch - 2ms/step
Epoch 71/100
96/96 - 0s - loss: 7.4791e-05 - 219ms/epoch - 2ms/step
Epoch 72/100
96/96 - 0s - loss: 1.1166e-04 - 210ms/epoch - 2ms/step
Epoch 73/100
96/96 - 0s - loss: 7.9723e-05 - 205ms/epoch - 2ms/step
Epoch 74/100
96/96 - 0s - loss: 9.8271e-05 - 205ms/epoch - 2ms/step
Epoch 75/100
96/96 - 0s - loss: 1.0441e-04 - 197ms/epoch - 2ms/step
Epoch 76/100
96/96 - 0s - loss: 1.1900e-04 - 206ms/epoch - 2ms/step
Epoch 77/100
96/96 - 0s - loss: 1.2196e-04 - 212ms/epoch - 2ms/step
Epoch 78/100
96/96 - 0s - loss: 5.2227e-05 - 202ms/epoch - 2ms/step
Epoch 79/100
96/96 - 0s - loss: 9.1618e-05 - 210ms/epoch - 2ms/step
Epoch 80/100
96/96 - 0s - loss: 1.3583e-04 - 206ms/epoch - 2ms/step
Epoch 81/100
96/96 - 0s - loss: 8.8135e-05 - 212ms/epoch - 2ms/step
Epoch 82/100
96/96 - 0s - loss: 1.2264e-04 - 203ms/epoch - 2ms/step
Epoch 83/100
96/96 - 0s - loss: 1.1490e-04 - 216ms/epoch - 2ms/step
Epoch 84/100
96/96 - 0s - loss: 9.9005e-05 - 201ms/epoch - 2ms/step
Epoch 85/100
96/96 - 0s - loss: 1.1500e-04 - 201ms/epoch - 2ms/step
Epoch 86/100
96/96 - 0s - loss: 1.0400e-04 - 213ms/epoch - 2ms/step
Epoch 87/100
96/96 - 0s - loss: 9.3426e-05 - 205ms/epoch - 2ms/step
Epoch 88/100
96/96 - 0s - loss: 7.5010e-05 - 209ms/epoch - 2ms/step
```

```
Xn_train1 = np.reshape(Xn_train,(Xn_train.shape[0],1,Xn_train.shape[1]))
Xn_test1 = np.reshape(Xn_test,(Xn_test.shape[0],1,Xn_test.shape[1]))


nowcasting_p = model.predict(Xn_train1)
nowcasting_tp = model.predict(Xn_test1)
```

```
nowcasting = np.concatenate([nowcasting_p, nowcasting_tp])
Xn['nowcasting'] = nowcasting
```

```
inflation['year-on-year growth rate'].plot(figsize=(10,6), marker='.')
Xn['nowcasting'].plot(figsize=(10,6));
```



Window Method

The results of the previous methods are too volatile.

New Section

```
commodity
```

	Gold-Open	Gold-High	Gold-Low	Gold-Close	Gold-Volume	Palladium-Open	Palladi H
2009-01-02	881.5	881.5	868.9	878.8	46.0	190.00	192
2009-01-05	882.0	883.5	847.0	857.2	35.0	190.00	192
2009-01-06	855.1	867.6	840.0	865.4	113.0	190.00	192
2009-01-07	862.0	867.0	837.7	841.1	101.0	190.00	192
2009-01-08	837.9	861.0	837.9	853.9	255.0	190.00	192
...	...	...	...	...	...	...	

```
inflation
```

	CPI	year-on-year growth rate	g_p
2009-01-01	211.143	0.000298	NaN
2009-02-01	212.193	0.002362	-0.008765
2009-03-01	212.709	-0.003836	-0.015415
2009-04-01	213.240	-0.007369	-0.013436
2009-05-01	213.856	-0.012814	-0.022605
...	...	...	...
2021-03-01	264.877	0.026198	0.057146
2021-04-01	267.054	0.041597	0.063369
2021-05-01	269.195	0.049927	0.055432
2021-06-01	271.696	0.053915	0.071097

0 秒　完成时间：21:44