Problem1

Pejman

Nov 28th 2018

1 Introduction

1.1 Question:

Use freely available data from the web to predict/explain macroeconomic indicators. Financial/Economic/Fundamentals data are not allowed.

1.2 Description of the Approach and Steps

Description.- In the following we have gathered data from the internet, from an open source. These data includes GDP historical data from 1941 and a few other time series (historical data) of (potentially) indicators such as the US population, durables, and 3-month LIBOR.

Objective.- The objective is to find which of indicators above, or combination of them is a good proxy to the GDP.

As a second goal and second phase we may attempt to forecast the GDP using these indicators.

Steps.- Steps taken are as follows:

- Using https://fred.stlouisfed.org/ as the source, we gathered data of the following indicators: Population, Libor, Durables, GDP
- Since the GDP is quarterly reported an appropriate averaging over every three months of the other data is made.
- The etart and end dates are different and filtering was required.
- A roll of one day is performed to math the dates with the GDP report dates
- Illustration of the normalized data. Normalization is done by subtracting the mean and division be the max-min in the time series window.
- (although visually observable) The r^2-squared is performed by fitting a line in the scatter plot of the indicator-GDP.
- We found **the population** to be in the best agreement with the GDP.
- Extension...(if time allows) forward propagation of the indicators and prediction of the GDP.

2 Packages

In [9]:
```python
'''
Source of data : https://fred.stlouisfed.org/
Author : Pej
'''

import pandas as pd
import datetime
import matplotlib.pyplot as plt

### other packs
from  sklearn import linear_model
```

3 Cleaning and Acquiring Data

In [15]:
```python
### reading data and cleaning (less needed here though)
###############################################################################
##### Actual GDP
###############################################################################
gdp_data = pd.read_csv('./GDPC1.csv', index_col=0, parse_dates=[0])
gdp_data.columns = ['Actual GDP']
gdp_data = gdp_data.reset_index()
gdp_data['DATE']= pd.to_datetime(gdp_data['DATE'])
gdp_data = gdp_data.set_index('DATE')


###############################################################################
##### Oil Production (Finished_Motor_Gasoline)
###############################################################################
###### using the data on oil production from 1991 to 2018
oil_production =\
    pd.read_csv('./Weekly_US_Product_Supplied_of_Finished_Motor_Gasoline.csv',
                                 skiprows=[0,1,2, 3])
oil_production.columns=['Date', 'Productions']
oil_production.loc[:, 'Date'] = pd.to_datetime(oil_production['Date'])
oil_production_cleaned = oil_production.set_index('Date')
### resampling Quarterly
oil_prod_quarterly = oil_production_cleaned.resample('Q').mean()


###############################################################################
##### 3-MONTH LIBOR  (some cleaning...)
###############################################################################
libor3M = pd.read_csv('./USD3MTD156N.csv')
libor3M['DATE'] = pd.to_datetime(libor3M['DATE'])
libor3M =libor3M.set_index('DATE')
```

```
libor3M['USD3MTD156N'] = pd.to_numeric(libor3M['USD3MTD156N'].values[:],
       errors='coerce')
null_col =  libor3M.columns[libor3M.isnull().any()] # obvious !
num_nulls = len(libor3M[libor3M.isnull().any(axis=1)][null_col]) ## ne 0
libor3M = libor3M.dropna()  # num_nulls = 0

### This is a three month libor but in case the daycount is different...
libor3M = libor3M.resample('Q').mean()



################################################################################
##### DURABLE GOODS ORDERS
################################################################################
durable = pd.read_csv('./DGORDER.csv')
durable = durable.dropna(how='any')
durable['DATE'] = pd.to_datetime(durable['DATE'])
durable =durable.set_index('DATE')
### resampling
durable = durable.resample('Q').mean()

################################################################################
##### US Population
################################################################################
population = pd.read_csv('./POPTHM.csv')
population = population.dropna(how='any')
population['DATE'] = pd.to_datetime(population['DATE'])
population =population.set_index('DATE')
#### resampling
population = population.resample('Q').mean()
```

4 Illustration

In [16]:
```
################################################################################
### finding the row number of the most recent date
### there might be better ways ...
################################################################################

t0_oil   = datetime.datetime.strptime(str(oil_prod_quarterly.index[0]), "%Y-%m-%d %H:%M
t0_gdp   = datetime.datetime.strptime(str(gdp_data.index[0]), "%Y-%m-%d %H:%M:%S")
t0_libor = datetime.datetime.strptime(str(libor3M.index[0]), "%Y-%m-%d %H:%M:%S")
t0_pop   = datetime.datetime.strptime(str(population.index[0]), "%Y-%m-%d %H:%M:%S")
t0_durb  = datetime.datetime.strptime(str(durable.index[0]), "%Y-%m-%d %H:%M:%S")


tf_oil   = datetime.datetime.strptime(str(oil_prod_quarterly.index[-1]), "%Y-%m-%d %H:%
tf_gdp   = datetime.datetime.strptime(str(gdp_data.index[-1]), "%Y-%m-%d %H:%M:%S")
```

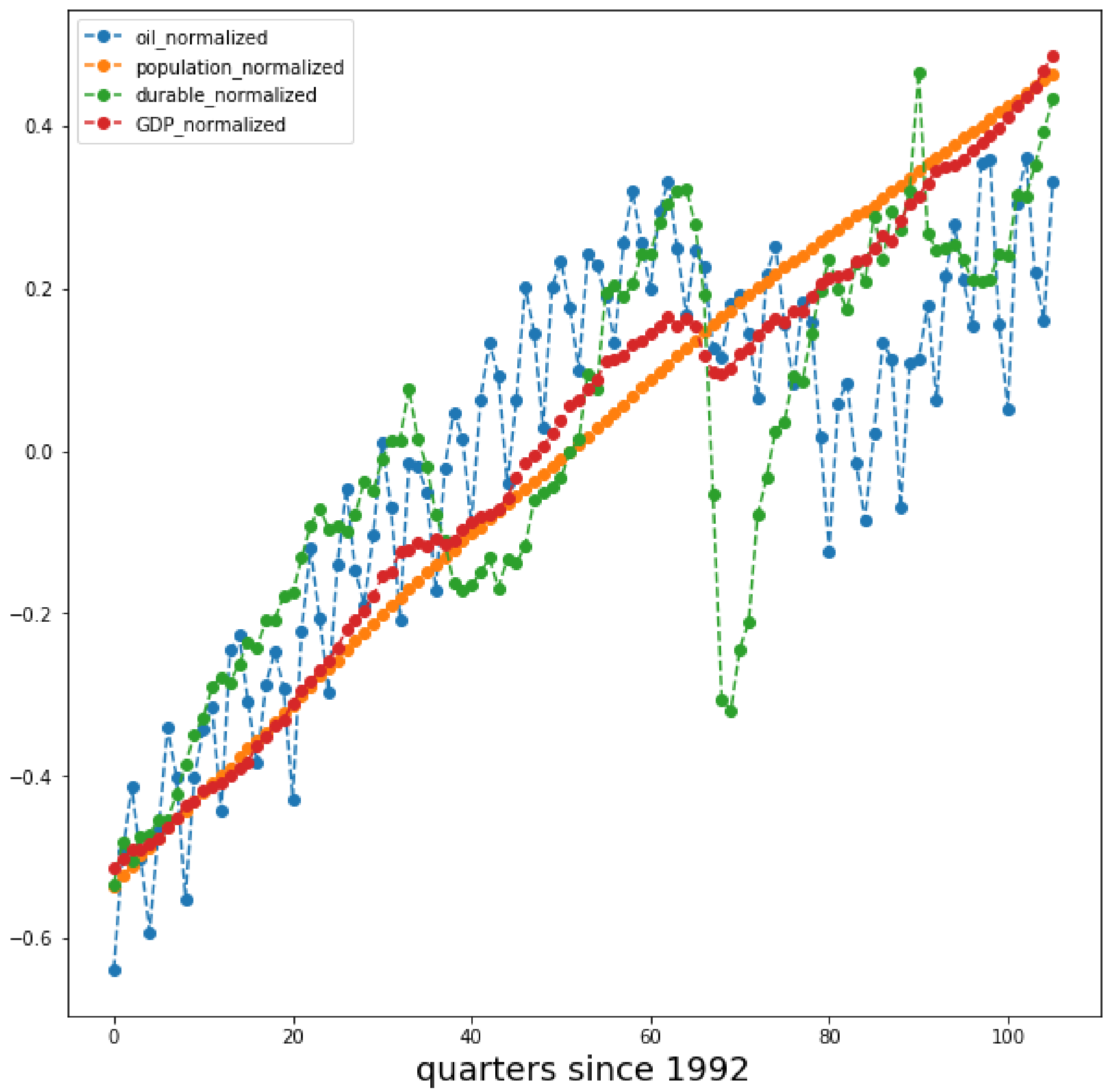
```
tf_libor = datetime.datetime.strptime(str(libor3M.index[-1]), "%Y-%m-%d %H:%M:%S")
tf_pop   = datetime.datetime.strptime(str(population.index[-1]), "%Y-%m-%d %H:%M:%S")
tf_durb  = datetime.datetime.strptime(str(durable.index[-1]), "%Y-%m-%d %H:%M:%S")

start_date =  max([t0_oil, t0_gdp, t0_libor, t0_pop, t0_durb ])
end_date   =  min([tf_oil, tf_gdp, tf_libor, tf_pop, tf_durb ])

##### filter and adjust the date
libor3M = libor3M.loc[start_date:end_date]
libor3M = libor3M.reset_index()
libor3M['DATE'] =\
        libor3M['DATE'].apply(lambda x: x + datetime.timedelta(days=1))
libor3M = libor3M.set_index('DATE')

oil_prod_quarterly = oil_prod_quarterly.loc[start_date:end_date]
oil_prod_quarterly = oil_prod_quarterly.reset_index()
oil_prod_quarterly['Date'] =\
            oil_prod_quarterly['Date'].apply(lambda x: x + datetime.timedelta(days=1))
oil_prod_quarterly = oil_prod_quarterly.set_index('Date')


population = population.loc[start_date:end_date]
population = population.reset_index()
population['DATE'] =\
        population['DATE'].apply(lambda x: x + datetime.timedelta(days=1))
population = population.set_index('DATE')


durable = durable.loc[start_date:end_date]
durable = durable.reset_index()
durable['DATE'] =\
        durable['DATE'].apply(lambda x: x + datetime.timedelta(days=1))
durable = durable.set_index('DATE')
```

5 Analysis

5.1 R2-Squared

In [17]:
```
#### visualization:
final_dt = pd.concat([oil_prod_quarterly,
                      libor3M,
                      population,
                      durable,
                      gdp_data], axis=1).dropna()

final_dt.columns=['oil',
                  'libor',
```

```
                    'population',
                    'durable',
                    'GDP',
                    ]

## normalize for illustration
final_dt_norm = (final_dt - final_dt.mean())/(final_dt.max() - final_dt.min())

### seems like libor is not a good fit unless we consider some non-linear relation
### between Libor and GDP ... so let's drop it for now.
fig, ax = plt.subplots(figsize=[10,10])
for col in range(final_dt_norm.shape[1]):
    if final_dt_norm.columns[col]!= 'libor':
        ax.plot(final_dt_norm.values[:, col],
                label=str(final_dt_norm.columns[col])+'_normalized')
ax.set_xlabel('quarters since 1992', **{'fontsize':18})
plt.legend()
plt.show()

###### let's now the correlation variace around the fit
###### with the gdp or not.
###### From the plot they are not perfactly in shape ... but anyway...

y_data = final_dt.iloc[:, -1]       ##  gdp the target

for i in range(4):  ### since we have chosen four indicators
    X_data = final_dt.iloc[:, [i]] ##  the indexs we chose
    ##
    lin_mod = linear_model.LinearRegression()
    _ = lin_mod.fit(X_data, y_data)
    print ' ==========================   ======================== '
    print 'The indecator chosen is : %s ' %(final_dt.columns[i])
    print('R-Squared of Linear Regression Model:', lin_mod.score(X_data, y_data))
    print ' ==========================   ======================== \n\n'
```




```
 ===========================   =========================
The indecator chosen is : oil
('R-Squared of Linear Regression Model:', 0.7433326914980255)
 ===========================   =========================


 ===========================   =========================
The indecator chosen is : libor
('R-Squared of Linear Regression Model:', 0.44945552780276155)
 ===========================   =========================


 ===========================   =========================
```

```
The indecator chosen is : population
('R-Squared of Linear Regression Model:', 0.98341604629017587)
 ==========================   ========================


 ==========================   ========================
The indecator chosen is : durable
('R-Squared of Linear Regression Model:', 0.78012867601848335)
 ==========================   ========================
```

5.2 Others