Environmental Intensity Level Regressions and Dickey–Fuller test

In the notebook 'PredictingTimeSeries_&_PilotStock_CompDescription', we ran several linear regressions combining different features and we saw it yield a high R2 score. In this notebook, we will run the regressions but focus on their coefficients.

Also, we are going to do some Dickey-Fuller test

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import warnings
from sklearn import linear_model
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import requests
from bs4 import BeautifulSoup
import os
import statsmodels.api as sm
warnings.filterwarnings('ignore')
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is
deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [ ]:

```
df=pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datasets/Env
df.head()
```

Out[ ]:

	ISIN	Year	CompanyName	Country	Industry(Exiobase)	EnvironmentalIntensity(Sales)	EnvironmentalIntensity(OpInc)	TotalEnviror
0	DE0005545503	2016	1&1 DRILLISCH AG	Germany	Post and telecommunications (64)	-0.07%	-0.82%	
1	GB00B1YW4409	2010	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.12%	-0.11%	
2	GB00B1YW4409	2011	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.16%	-0.16%	
3	GB00B1YW4409	2012	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.15%	NaN	
4	US88579Y1010	2010	3M COMPANY	United States	Activities of membership organisation n.e.c. (91)	-7.90%	-35.45%	-2

5 rows × 39 columns

In [ ]:

```
years = [2016, 2017, 2018]
df_industry = df.groupby('Industry(Exiobase)').count()['CompanyName'].reset_index()
industries = df_industry[df_industry['CompanyName'] > 3]['Industry(Exiobase)']
df_industry_count4 = df[df['Industry(Exiobase)'].isin(industries)]
df_c = df_industry_count4.copy()
def predictiveModel(outcomeYear, pastYears, df_c):
    years.sort()
    for year in years:
        data = df_c[df_c['Year'] == year]
        data = data.loc[:,['CompanyName','Env_intensity','industry_avg_year']]
        data.rename(columns={'Env_intensity': f'Env_intensity_{year}','industry_avg_year':f'industry_avg_year_{
        if(year == min(years)):
            data1 = pd.DataFrame(data)
        else:
            data2 = pd.merge(data1, data, on=["CompanyName"])
            data1 = data2.copy()
    data3 = df_c[df_c['Year'] == outcomeYear]
    data3 = data3[['CompanyName','Env_intensity','industry_avg_year']]
    data3.rename(columns={'Env_intensity': f'Env_intensity_{outcomeYear}','industry_avg_year':f'industry_avg_ye
    data3 = pd.merge(data3, data2, on=["CompanyName"])

    filter_col = [col for col in data3 if ((col.startswith('Env_intensity') and not(col.endswith(f'{outcomeYear
    outcome_col = [col for col in data3 if (col.startswith('Env_intensity') and col.endswith(f'{outcomeYear}'))
    X=data3[filter_col]
    y=data3[outcome_col]

    x = sm.add_constant(X)
    print(sm.OLS(y, x).fit().summary())
```

In [ ]:

```
predictiveModel(2019, years, df_c)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Env_intensity_2019   R-squared:                       0.884
Model:                            OLS   Adj. R-squared:                  0.883
Method:                 Least Squares   F-statistic:                     1340.
Date:                Thu, 15 Jul 2021   Prob (F-statistic):               0.00
Time:                        22:30:25   Log-Likelihood:                 1179.1
No. Observations:                1065   AIC:                            -2344.
Df Residuals:                    1058   BIC:                            -2309.
Df Model:                           6
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                     -0.0015      0.003     -0.487      0.627      -0.007       0.004
Env_intensity_2016        -0.0512      0.034     -1.517      0.130      -0.118       0.015
industry_avg_year_2016    -0.1608      0.064     -2.503      0.012      -0.287      -0.035
Env_intensity_2017        -0.0646      0.047     -1.370      0.171      -0.157       0.028
industry_avg_year_2017    -0.0608      0.069     -0.884      0.377      -0.196       0.074
Env_intensity_2018         1.0234      0.038     27.134      0.000       0.949       1.097
industry_avg_year_2018     0.2641      0.087      3.021      0.003       0.093       0.436
==============================================================================
Omnibus:                     1463.812   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1055912.255
Skew:                           6.975   Prob(JB):                         0.00
Kurtosis:                     156.625   Cond. No.                         44.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
def predictiveModel_v2(outcomeYear, pastYears, df_c):
    years.sort()
    for year in years:
        data = df_c[df_c['Year'] == year]
        data = data.loc[:,['CompanyName','Env_intensity','industry_avg_year','Industry_indicator_year']]
        data.rename(columns={'Env_intensity': f'Env_intensity_{year}','industry_avg_year':f'industry_avg_year_{
        if(year == min(years)):
            data1 = pd.DataFrame(data)
        else:
            data2 = pd.merge(data1, data, on=["CompanyName"])
            data1 = data2.copy()
    data3 = df_c[df_c['Year'] == outcomeYear]
    data3 = data3[['CompanyName','Env_intensity']]
    data3.rename(columns={'Env_intensity': f'Env_intensity_{outcomeYear}'}, inplace=True)
    data3 = pd.merge(data3, data2, on=["CompanyName"])

    filter_col = [col for col in data3 if ((col.startswith('Env_intensity') and not(col.endswith(f'{outcomeYear
    outcome_col = [col for col in data3 if (col.startswith('Env_intensity') and col.endswith(f'{outcomeYear}'))
    X=data3[filter_col]
    y=data3[outcome_col]

    x = sm.add_constant(X)
    print(sm.OLS(y, x).fit().summary())
```

In [ ]:

```
predictiveModel_v2(2019, years, df_c)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Env_intensity_2019   R-squared:                       0.884
Model:                            OLS   Adj. R-squared:                  0.883
Method:                 Least Squares   F-statistic:                     893.7
Date:                Thu, 15 Jul 2021   Prob (F-statistic):               0.00
Time:                        22:31:46   Log-Likelihood:                 1180.6
No. Observations:                1065   AIC:                            -2341.
Df Residuals:                    1055   BIC:                            -2291.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           -0.0020      0.003     -0.576      0.565      -0.009       0.005
Env_intensity_2016              -0.0603      0.035     -1.737      0.083      -0.128       0.008
industry_avg_year_2016          -0.1463      0.066     -2.222      0.026      -0.276      -0.017
Industry_indicator_year_2016     0.0049      0.005      1.036      0.300      -0.004       0.014
Env_intensity_2017              -0.0686      0.048     -1.434      0.152      -0.163       0.025
industry_avg_year_2017          -0.0486      0.070     -0.692      0.489      -0.186       0.089
Industry_indicator_year_2017     0.0041      0.005      0.747      0.455      -0.007       0.015
Env_intensity_2018               1.0306      0.038     26.999      0.000       0.956       1.106
industry_avg_year_2018           0.2426      0.088      2.742      0.006       0.069       0.416
Industry_indicator_year_2018    -0.0067      0.005     -1.283      0.200      -0.017       0.004
==============================================================================
Omnibus:                     1463.792   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1057584.362
Skew:                           6.974   Prob(JB):                         0.00
Kurtosis:                     156.748   Cond. No.                         74.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
def predictiveModel_v3(outcomeYear, pastYears, df_c):
    years.sort()
    for year in years:
        data = df_c[df_c['Year'] == year]
        data = data.loc[:,['CompanyName','Env_intensity','industry_avg_year','Industry_indicator_year','Enviror
        data.rename(columns={'Env_intensity': f'Env_intensity_{year}','industry_avg_year':f'industry_avg_year_{
        if(year == min(years)):
            data1 = pd.DataFrame(data)
        else:
            data2 = pd.merge(data1, data, on=["CompanyName"])
            data1 = data2.copy()
    data2.dropna(inplace=True)
    data3 = df_c[df_c['Year'] == outcomeYear]
    data3 = data3[['CompanyName','Env_intensity']]
    data3.rename(columns={'Env_intensity': f'Env_intensity_{outcomeYear}'}, inplace=True)
    data3 = pd.merge(data3, data2, on=["CompanyName"])

    filter_col = [col for col in data3 if ((col.startswith('Env_intensity') and not(col.endswith(f'{outcomeYear
    outcome_col = [col for col in data3 if (col.startswith('Env_intensity') and col.endswith(f'{outcomeYear}'))
    X=data3[filter_col]
    y=data3[outcome_col]

    x = sm.add_constant(X)
    print(sm.OLS(y, x).fit().summary())
```

In [ ]:

```
predictiveModel_v3(2019, years, df_c)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Env_intensity_2019   R-squared:                       0.878
Model:                            OLS   Adj. R-squared:                  0.876
Method:                 Least Squares   F-statistic:                     596.2
Date:                Thu, 15 Jul 2021   Prob (F-statistic):               0.00
Time:                        22:36:03   Log-Likelihood:                 1109.2
No. Observations:                1010   AIC:                            -2192.
Df Residuals:                     997   BIC:                            -2128.
Df Model:                          12
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           -0.0034      0.004     -0.900      0.368      -0.011       0.004
Env_intensity_2016              -0.0087      0.045     -0.194      0.846      -0.097       0.080
industry_avg_year_2016          -0.2048      0.080     -2.562      0.011      -0.362      -0.048
Industry_indicator_year_2016     0.0047      0.005      0.930      0.353      -0.005       0.015
Environmental_Growth_2016        0.0002   6.12e-05      2.621      0.009    4.03e-05       0.000
Env_intensity_2017              -0.1571      0.059     -2.648      0.008      -0.274      -0.041
industry_avg_year_2017           0.0230      0.095      0.243      0.808      -0.163       0.209
Industry_indicator_year_2017     0.0037      0.006      0.647      0.518      -0.008       0.015
Environmental_Growth_2017    -1.794e-05   9.83e-06     -1.824      0.068   -3.72e-05    1.36e-06
Env_intensity_2018               1.0721      0.045     23.748      0.000       0.984       1.161
industry_avg_year_2018           0.2198      0.107      2.047      0.041       0.009       0.430
Industry_indicator_year_2018    -0.0056      0.006     -0.990      0.322      -0.017       0.005
Environmental_Growth_2018      9.94e-05   4.24e-05      2.344      0.019    1.62e-05       0.000
==============================================================================
Omnibus:                     1385.003   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           950049.204
Skew:                           6.930   Prob(JB):                         0.00
Kurtosis:                     152.611   Cond. No.                     1.59e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary: Using environmental intensity and industry average year have highest explanatory power. The coefficients for industry average of 2016 and 2018 and environmental intensity for 2018 are statistically significant. 2018 EI has highest(1.02) fixed effect on 2019 EI.

Dickey–Fuller test

In [27]:

```
from statsmodels.tsa.stattools import adfuller
df=pd.read_csv('Environmental_impact_cleaned.csv')
ind = df.copy()
y2018=list(ind[ind['Year'] == 2018]['Env_intensity'])
y2018=pd.Series(y2018)
X = y2018.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.129262, mean2=-0.101468
variance1=0.094017, variance2=0.052920
ADF Statistic: -24.980342
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2017=list(ind[ind['Year'] == 2017]['Env_intensity'])
y2017=pd.Series(y2017)
from statsmodels.tsa.stattools import adfuller
X = y2017.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.123985, mean2=-0.108138
variance1=0.089221, variance2=0.063000
ADF Statistic: -26.071015
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2016=list(ind[ind['Year'] == 2016]['Env_intensity'])
y2016=pd.Series(y2016)
from statsmodels.tsa.stattools import adfuller
X = y2016.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -38.998411
p-value: 0.000000
Critical Values:
mean1=-0.128953, mean2=-0.113759
variance1=0.084427, variance2=0.066553
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2015=list(ind[ind['Year'] == 2015]['Env_intensity'])
y2015=pd.Series(y2015)
from statsmodels.tsa.stattools import adfuller
X = y2015.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.128073, mean2=-0.112539
variance1=0.072975, variance2=0.066954
ADF Statistic: -25.147693
p-value: 0.000000
Critical Values:
        1%: -3.435
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2014=list(ind[ind['Year'] == 2014]['Env_intensity'])
y2014=pd.Series(y2014)
from statsmodels.tsa.stattools import adfuller
X = y2014.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.113915, mean2=-0.100781
variance1=0.062142, variance2=0.055520
ADF Statistic: -15.509563
p-value: 0.000000
Critical Values:
        1%: -3.435
        5%: -2.864
        10%: -2.568
```

Summary: The environmental intensity from 2014 to 2018 for each year the data is stational

Next,let's exam the industry average for each year to see whether they are stational

In [ ]:

```
ind.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14515 entries, 0 to 14514
Data columns (total 39 columns):
 #   Column                                                 Non-Null Count  Dtype
---  ------                                                 --------------  -----
 0   ISIN                                                   14515 non-null  object
 1   Year                                                   14515 non-null  int64
 2   CompanyName                                            14515 non-null  object
 3   Country                                                14515 non-null  object
 4   Industry(Exiobase)                                     14515 non-null  object
 5   EnvironmentalIntensity(Sales)                          14515 non-null  object
 6   EnvironmentalIntensity(OpInc)                          13700 non-null  object
 7   TotalEnvironmentalCost                                 14515 non-null  object
 8   WorkingCapacity                                        14515 non-null  object
 9   FishProductionCapacity                                 14515 non-null  object
 10  CropProductionCapacity                                 14515 non-null  object
 11  MeatProductionCapacity                                 14515 non-null  object
 12  Biodiversity                                           14515 non-null  object
 13  AbioticResources                                       14515 non-null  object
 14  Waterproductioncapacity(Drinkingwater&IrrigationWater)  14515 non-null  object
 15  WoodProductionCapacity                                 14515 non-null  object
 16  SDG1.5                                                 14515 non-null  object
 17  SDG2.1                                                 14515 non-null  object
 18  SDG2.2                                                 14515 non-null  object
 19  SDG2.3                                                 14515 non-null  object
 20  SDG2.4                                                 14515 non-null  object
 21  SDG3.3                                                 14515 non-null  object
 22  SDG3.4                                                 14515 non-null  object
 23  SDG3.9                                                 14515 non-null  object
 24  SDG6                                                   14515 non-null  object
 25  SDG12.2                                                14515 non-null  object
 26  SDG14.1                                                14515 non-null  object
 27  SDG14.2                                                14515 non-null  object
 28  SDG14.3                                                14515 non-null  object
 29  SDG14.c                                                14515 non-null  object
 30  SDG15.1                                                14515 non-null  object
 31  SDG15.2                                                14515 non-null  object
 32  SDG15.5                                                14515 non-null  object
 33  %Imputed                                               14515 non-null  object
 34  Env_intensity                                          14515 non-null  float64
 35  industry_avg                                           14515 non-null  float64
 36  industry_avg_year                                      14515 non-null  float64
 37  Industry_indicator_year                                14515 non-null  int64
 38  Environmental_Growth                                   11924 non-null  float64
dtypes: float64(4), int64(2), object(33)
memory usage: 4.3+ MB
```

Test whether the industry average for each year we used is stationary :

In [ ]:

```
y2018=list(ind[ind['Year'] == 2018]['industry_avg_year'])
y2018=pd.Series(y2018)
from statsmodels.tsa.stattools import adfuller
X = y2018.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.121867, mean2=-0.108862
variance1=0.032617, variance2=0.026903
ADF Statistic: -10.463661
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2017=list(ind[ind['Year'] == 2017]['industry_avg_year'])
y2017=pd.Series(y2017)
from statsmodels.tsa.stattools import adfuller
X = y2017.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.117744, mean2=-0.114378
variance1=0.030029, variance2=0.028428
ADF Statistic: -27.227921
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2016=list(ind[ind['Year'] == 2016]['industry_avg_year'])
y2016=pd.Series(y2016)
from statsmodels.tsa.stattools import adfuller
X = y2016.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.125713, mean2=-0.117000
variance1=0.035538, variance2=0.033986
ADF Statistic: -38.943780
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [ ]:

```
y2015=list(ind[ind['Year'] == 2015]['industry_avg_year'])
y2015=pd.Series(y2015)
from statsmodels.tsa.stattools import adfuller
X = y2015.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.125579, mean2=-0.115032
variance1=0.038191, variance2=0.033880
ADF Statistic: -20.655671
p-value: 0.000000
Critical Values:
        1%: -3.435
        5%: -2.863
        10%: -2.568
```

Summary:Each year from 2015 to 2018, the industry average is stational. And the model we used which includes the the industry average is predictable.

Let's take a look at environmental growth

In [28]:

```
ind['Environmental_Growth']=ind['Environmental_Growth'].fillna(ind['Environmental_Growth'].mean())
```

In [29]:

```
y2016=list(ind[ind['Year'] == 2016]['Environmental_Growth'])
y2016=pd.Series(y2016)
from statsmodels.tsa.stattools import adfuller
X = y2016.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=5.411575, mean2=4.908355
variance1=1923.017251, variance2=5805.189898
ADF Statistic: -13.302222
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [30]:

```
y2017=list(ind[ind['Year'] == 2017]['Environmental_Growth'])
y2017=pd.Series(y2017)
from statsmodels.tsa.stattools import adfuller
X = y2017.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=99.633708, mean2=-5.132693
variance1=5728066.542782, variance2=1756.053671
ADF Statistic: -41.477166
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [31]:

```
y2018=list(ind[ind['Year'] == 2018]['Environmental_Growth'])
y2018=pd.Series(y2018)
from statsmodels.tsa.stattools import adfuller
X = y2018.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=2.993927, mean2=0.514992
variance1=5296.795823, variance2=4757.639775
ADF Statistic: -40.731334
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

Summary: All of the environmental growths that we used is stational.

Let's take a look at top five industries

In [ ]:

```
ind.groupby('Industry(Exiobase)')['Env_intensity'].count().sort_values()
```

Out[ ]:

```
Industry(Exiobase)
Cultivation of cereal grains nec
          1
Forestry, logging and related service activities (02)
          5
Sea and coastal water transport
          6
Education (80)
          6
Production of electricity by petroleum and other oil derivatives
         12
Mining of coal and lignite; extraction of peat (10)
         15
Manufacture of tobacco products (16)
         22
Copper production
         22
Manufacture of furniture; manufacturing n.e.c. (36)
         22
Publishing, printing and reproduction of recorded media (22)
         32
Wholesale trade and commission trade, except of motor vehicles and motorcycles (51)
         37
Transport via pipelines
         39
Production of electricity by solar photovoltaic
         41
N-fertiliser
         42
Manufacture of basic iron and steel and of ferro-alloys and first products thereof
         45
Manufacture of wearing apparel; dressing and dyeing of fur (18)
         47
Transport via railways
         48
Manufacture of textiles (17)
         51
Health and social work (85)
         51
Collection, purification and distribution of water (41)
         53
Sale, maintenance, repair of motor vehicles, motor vehicles parts, motorcycles, motor cycles parts and accessoi
ries      62
Production of electricity by hydro
         71
Extraction of natural gas and services related to natural gas extraction, excluding surveying
         91
Other land transport
        117
Manufacture of rubber and plastic products (25)
        123
Insurance and pension funding, except compulsory social security (66)
        140
Research and development (73)
        147
Renting of machinery and equipment without operator and of personal and household goods (71)
        147
Petroleum Refinery
        192
Paper
        196
Air transport (62)
        200
Mining of other non-ferrous metal ores and concentrates
        233
Activities of membership organisation n.e.c. (91)
        235
Manufacture of beverages
        258
Activities auxiliary to financial intermediation (67)
        263
Recreational, cultural and sporting activities (92)
        272
Manufacture of radio, television and communication equipment and apparatus (32)
        274
Mining of chemical and fertilizer minerals, production of salt, other mining and quarrying n.e.c.
        303
Manufacture of motor vehicles, trailers and semi-trailers (34)
        341
Manufacture of machinery and equipment n.e.c. (29)
        364
Chemicals nec
        414
Manufacture of office machinery and computers (30)
        423
Manufacture of fabricated metal products, except machinery and equipment (28)
        433
Production of electricity nec
        444
Quarrying of sand and clay
        481
Extraction of crude petroleum and services related to crude oil extraction, excluding surveying
        496
Computer and related activities (72)
        507
Processing of Food products nec
        531
Other service activities (93)
        558
Post and telecommunications (64)
        599
Manufacture of medical, precision and optical instruments, watches and clocks (33)
        601
Retail trade, except of motor vehicles and motorcycles; repair of personal and household goods (52)
        682
Real estate activities (70)
        697
Construction (45)
        729
Manufacture of electrical machinery and apparatus n.e.c. (31)
        765
Financial intermediation, except insurance and pension funding (65)
       1529
Name: Env_intensity, dtype: int64
```

Top five industries:

Retail trade, except of motor vehicles and motorcycles; repair of personal and household goods (52)
Real estate activities(70)
Construction (45)
Manufacture of electrical machinery and apparatus n.e.c. (31)
Financial intermediation, except insurance and pension funding (65)

In [ ]:

```
listind=['Retail trade, except of motor vehicles and motorcycles; repair of personal and household goods (52)',
'Real estate activities(70)',
'Construction (45)',
'Manufacture of electrical machinery and apparatus n.e.c. (31)',
'Financial intermediation, except insurance and pension funding (65)']
num_order_new = ind[(ind['Industry(Exiobase)']=='Construction (45)')|(ind['Industry(Exiobase)'] == 'Financial i
|(ind['Industry(Exiobase)'] == 'Real estate activities(70)')|(ind['Industry(Exiobase)'] == 'Retail trade, excep
num_order_new
```

Out[ ]:

	ISIN	Year	CompanyName	Country	Industry(Exiobase)	EnvironmentalIntensity(Sales)	EnvironmentalIntensity(OpInc)	TotalEn
1	GB00B1YW4409	2010	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.12%	-0.11%	
2	GB00B1YW4409	2011	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.16%	-0.16%	
3	GB00B1YW4409	2012	3I GROUP PLC	United Kingdom	Financial intermediation, except insurance and...	-0.15%	NaN	
50	DE0005408116	2012	AAREAL BANK AG	Germany	Financial intermediation, except insurance and...	-0.10%	-0.69%	
51	DE0005408116	2013	AAREAL BANK AG	Germany	Financial intermediation, except insurance and...	-0.11%	-0.56%	
...	...	...	...	...	...	...	...	
14485	KYG989221000	2016	ZHEN DING TECHNOLOGY HOLDING LIMITED	Taiwan	Manufacture of electrical machinery and appara...	-8.73%	-155.74%	
14486	KYG989221000	2017	ZHEN DING TECHNOLOGY HOLDING LIMITED	Taiwan	Manufacture of electrical machinery and appara...	-6.03%	-76.11%	
14487	KYG989221000	2018	ZHEN DING TECHNOLOGY HOLDING LIMITED	Taiwan	Manufacture of electrical machinery and appara...	-6.15%	-48.85%	
14488	KYG989221000	2019	ZHEN DING TECHNOLOGY HOLDING LIMITED	Taiwan	Manufacture of electrical machinery and appara...	-6.02%	-49.99%	
14514	GB0006452857	2019	nmcn plc	United Kingdom	Construction (45)	-0.56%	-24.83%	

3705 rows × 39 columns

In [ ]:

```
y2018=list(num_order_new[num_order_new['Year'] == 2018]['Env_intensity'])
y2018=pd.Series(y2018)
from statsmodels.tsa.stattools import adfuller
X = y2018.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.041328, mean2=-0.039877
variance1=0.010222, variance2=0.005468
ADF Statistic: -21.045903
p-value: 0.000000
Critical Values:
        1%: -3.445
        5%: -2.868
        10%: -2.570
```

In [ ]:

```
y2017=list(num_order_new[num_order_new['Year'] == 2017]['Env_intensity'])
y2017=pd.Series(y2017)
from statsmodels.tsa.stattools import adfuller
X = y2017.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.041998, mean2=-0.035402
variance1=0.010698, variance2=0.002860
ADF Statistic: -5.443552
p-value: 0.000003
Critical Values:
        1%: -3.446
        5%: -2.868
        10%: -2.570
```

In [ ]:

```
y2016=list(num_order_new[num_order_new['Year'] == 2016]['Env_intensity'])
y2016=pd.Series(y2016)
from statsmodels.tsa.stattools import adfuller
X = y2016.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.044473, mean2=-0.037712
variance1=0.011978, variance2=0.002783
ADF Statistic: -20.528414
p-value: 0.000000
Critical Values:
        1%: -3.446
        5%: -2.868
        10%: -2.570
```

In [ ]:

```
y2015=list(num_order_new[num_order_new['Year'] == 2015]['Env_intensity'])
y2015=pd.Series(y2015)
from statsmodels.tsa.stattools import adfuller
X = y2015.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.043269, mean2=-0.028244
variance1=0.012533, variance2=0.019730
ADF Statistic: -18.508807
p-value: 0.000000
Critical Values:
        1%: -3.449
        5%: -2.870
        10%: -2.571
```

Let's check the top 3 industries

In [ ]:

```
num_order_new = ind[(ind['Industry(Exiobase)']=='Construction (45)')|(ind['Industry(Exiobase)'] == 'Financial i
```

In [ ]:

```
y2018=list(num_order_new[num_order_new['Year'] == 2018]['Env_intensity'])
y2018=pd.Series(y2018)
from statsmodels.tsa.stattools import adfuller
X = y2018.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.045647, mean2=-0.045448
variance1=0.012042, variance2=0.006497
ADF Statistic: -19.134100
p-value: 0.000000
Critical Values:
        1%: -3.449
        5%: -2.870
        10%: -2.571
```

In [ ]:

```
y2017=list(num_order_new[num_order_new['Year'] == 2017]['Env_intensity'])
y2017=pd.Series(y2017)
from statsmodels.tsa.stattools import adfuller
X = y2017.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.039995, mean2=-0.040480
variance1=0.003754, variance2=0.003352
ADF Statistic: -19.354295
p-value: 0.000000
Critical Values:
        1%: -3.448
        5%: -2.870
        10%: -2.571
```

In [ ]:

```
y2016=list(num_order_new[num_order_new['Year'] == 2016]['Env_intensity'])
y2016=pd.Series(y2016)
from statsmodels.tsa.stattools import adfuller
X = y2016.values
result = adfuller(X)
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
mean1=-0.048769, mean2=-0.043516
variance1=0.014048, variance2=0.003285
ADF Statistic: -19.089448
p-value: 0.000000
Critical Values:
        1%: -3.449
        5%: -2.870
        10%: -2.571
```

Conclusion

Summary: All of the data that we used is stational.

Next, we will continue our analysis in the 'DistilBERT_CompaniesDescription' notebook