# Final Project - Big Data


***Uriel bender***

***Dor ben-zvi***

***Ohad haviv***


## Link to Exercises

- Make the Pollutant dfs
- run linear reg
- Create the features Dataframe
- cllasification
- linear reg predict on next year

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt  # visualization
import seaborn as sns # visualization
```

In [2]:

```
df = pd.read_csv("AIR_EMISSIONS.csv")
```

In [3]:

```
df.head()
```

Out[3]:

| | COU | Country | POL | Pollutant | VAR | Variable | YEA | Year | Unit Code | Unit | PowerCode Code | PowerCode | Reference Period Code | Reference Period | Va |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AUS | Australia | SOX | Sulphur Oxides | TOT | Total man-made emissions | 1990 | 1990 | TONNE | Tonnes | 3 | Thousands | NaN | NaN | 1585. |
| 1 | AUS | Australia | SOX | Sulphur Oxides | TOT | Total man-made emissions | 1991 | 1991 | TONNE | Tonnes | 3 | Thousands | NaN | NaN | 1570. |
| 2 | AUS | Australia | SOX | Sulphur Oxides | TOT | Total man-made emissions | 1992 | 1992 | TONNE | Tonnes | 3 | Thousands | NaN | NaN | 1652. |
| 3 | AUS | Australia | SOX | Sulphur Oxides | TOT | Total man-made emissions | 1993 | 1993 | TONNE | Tonnes | 3 | Thousands | NaN | NaN | 1743. |
| 4 | AUS | Australia | SOX | Sulphur Oxides | TOT | Total man-made emissions | 1994 | 1994 | TONNE | Tonnes | 3 | Thousands | NaN | NaN | 1764. |

In [4]:

```
df.drop(labels=['COU','POL','VAR','YEA','Unit Code','Reference Period Code','Reference
Period','Flag Codes','Flags','PowerCode','PowerCode Code','Variable','Unit'],axis=1,inplace=True)
df.head()
```

Out[4]:

Out[4]:

| | Country | Pollutant | Year | Value |
|---|---|---|---|---|
| 0 | Australia | Sulphur Oxides | 1990 | 1585.754 |
| 1 | Australia | Sulphur Oxides | 1991 | 1570.777 |
| 2 | Australia | Sulphur Oxides | 1992 | 1652.946 |
| 3 | Australia | Sulphur Oxides | 1993 | 1743.161 |
| 4 | Australia | Sulphur Oxides | 1994 | 1764.906 |

In [5]:

```
NitrogenOxides=df[df['Pollutant']=='Nitrogen Oxides']
Particulates=df[df['Pollutant']=='Particulates (PM2.5)']
CarbonMonoxide=df[df['Pollutant']=='Carbon Monoxide']
SulphurOxides=df[df['Pollutant']=='Sulphur Oxides']
```

### Make the Pollutant df

In [6]:

```
NitrogenOxides.rename(columns={'Value':'Nitrogen Oxides'},inplace=True)
NitrogenOxides.drop(labels=['Pollutant'],axis=1,inplace=True)

Particulates.rename(columns={'Value':'Particulates'},inplace=True)
Particulates.drop(labels=['Pollutant'],axis=1,inplace=True)

CarbonMonoxide.rename(columns={'Value':'Carbon Monoxide'},inplace=True)
CarbonMonoxide.drop(labels=['Pollutant'],axis=1,inplace=True)

SulphurOxides.rename(columns={'Value':'Sulphur Oxides'},inplace=True)
SulphurOxides.drop(labels=['Pollutant'],axis=1,inplace=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:4223: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(**kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:4102: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

In [7]:

```
CarbonMonoxide.head(2)
```

Out[7]:

| | Country | Year | Carbon Monoxide |
|---|---|---|---|
| 56 | Australia | 1990 | 5728.460 |
| 57 | Australia | 1991 | 5747.308 |

### Create the features Dataframe

In [8]:

```
# Green growth pd
Green_Growth_features = pd.read_csv("Green_Growth.csv")
Renewable=Green_Growth_features[Green_Growth_features['Variable']=='Renewable energy public RD&D b
udget, % total energy public RD&D']
gdp=Green_Growth_features[Green_Growth_features['Variable']=='Energy public RD&D budget, % GDP']
fossilFules=Green_Growth_features[Green_Growth_features['Variable']=='Fossil fuel public RD&D budg
et (excluding CCS), % total energy public RD&D']

Renewable.drop(labels=['COU','VAR','YEA','PowerCode','Unit Code' ,'PowerCode Code','Reference Perio
d Code','Reference Period','Flag Codes','Flags','Unit','Variable'],axis=1,inplace=True)
Renewable.rename(columns={'Value':'Renewable energy public RD&D budget, % total energy public RD&D
'},inplace=True)

gdp.drop(labels=['COU','VAR','YEA','PowerCode','Unit Code' ,'PowerCode Code','Reference Period Code
','Reference Period','Flag Codes','Flags','Unit','Variable'],axis=1,inplace=True)
gdp.rename(columns={'Value':'Energy public RD&D budget, % GDP'},inplace=True)

fossilFules.drop(labels=['COU','VAR','YEA','PowerCode','Unit Code' ,'PowerCode Code','Reference Per
iod Code','Reference Period','Flag Codes','Flags','Unit','Variable'],axis=1,inplace=True)
fossilFules.rename(columns={'Value':'Fossil fuel public RD&D budget (excluding CCS), % total
energy public RD&D'},inplace=True)

# Gdp per capita db
gdp_per_capita = pd.read_csv("gdp_per_capita.csv")
gdp_df=gdp_per_capita[gdp_per_capita['Variable']=='Real GDP per capita']
gdp_df.drop(labels=['COU','VAR','YEA','PowerCode','Unit Code' ,'PowerCode Code','Reference Period C
ode','Reference Period','Flag Codes','Flags','Unit','Variable'],axis=1,inplace=True)
gdp_df.rename(columns={'Value':'GDP per capita'},inplace=True)

# Population pd
Population_df = pd.read_csv("population.csv")
Population_df.drop(columns=['LOCATION','SUBJECT','Subject','SEX','Sex','FREQUENCY','Frequency','Tim
e','Unit Code','Unit','PowerCode Code','PowerCode','Reference Period Code','Reference
Period','Flag Codes','Flags'],inplace=True)
Population_df.rename(columns={'TIME':'Year','Value':'Population'},inplace=True)

# MotorVehicle pd
motorVehicle = pd.read_csv("Motor_Vehicle.csv")
motorVehicle.drop(labels=['COUNTRY','INDICATOR','Indicator','YEAR','Flag
Codes','Flags'],axis=1,inplace=True)
motorVehicle.rename(columns={'Value':'Road traffic in thousand vehicle-km per road motor vehicle'}
,inplace=True)

# Share cars pd
ShareCars = pd.read_csv("Share_cars.csv")
ShareCars.drop(labels=['COUNTRY','INDICATOR','Indicator','YEAR','Flag
Codes','Flags'],axis=1,inplace=True)
ShareCars.rename(columns={'Value':'Share of passenger cars in total road motor vehicle'},inplace=T
rue)

# Support Measures for Fossil Fuels : Natural_Gas and Petroleum

fossil_fuels=pd.read_csv("fossil_fuels.csv")

Natural_Gas=fossil_fuels[fossil_fuels['MEASURE']=='NATGAS']
Petroleum=fossil_fuels[fossil_fuels['MEASURE']=='PETROLEUM']



Natural_Gas.drop(labels=['LOCATION','MEASURE','TIME','Measure' ,'Flag Codes'
,'Flags'],axis=1,inplace=True)
Natural_Gas.rename(columns={'Value':'Natural_Gas'},inplace=True)


Petroleum.drop(labels=['LOCATION','MEASURE','TIME','Flag Codes','Measure' ,'Flags'],axis=1,inplace
=True)
Petroleum.rename(columns={'Value':'Petroleum'},inplace=True)
```

### Create the NitrogenOxides Dataframe

In [9]:

```
NitrogenOxides2=NitrogenOxides.copy()
NitrogenOxides2["Year"]=NitrogenOxides2["Year"]-1
```

```
NitrogenOxides2.rename(columns={'Nitrogen Oxides':'Nitrogen_Oxides_next_year'
                                       },inplace=True)
```

In [10]:

```
NitrogenOxides2.head(1)
```

Out[10]:

| | Country | Year | Nitrogen_Oxides_next_year |
|---|---|---|---|
| 28 | Australia | 1989 | 1620.749 |

In [11]:

```
NitrogenOxides=pd.merge(NitrogenOxides,Renewable ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,gdp ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,fossilFules ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,gdp_df ,how='left',on=['Country','Year'])

NitrogenOxides=pd.merge(NitrogenOxides,motorVehicle ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,ShareCars ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,Population_df ,how='left',on=['Country','Year'])
NitrogenOxides=pd.merge(NitrogenOxides,NitrogenOxides2 ,how='left',on=['Country','Year'])
# NitrogenOxides=pd.merge(NitrogenOxides,Natural_Gas ,how='left',on=['Country','Year'])
# NitrogenOxides=pd.merge(NitrogenOxides,Petroleum ,how='left',on=['Country','Year'])
```

### Create the Particulates Dataframe

In [12]:

```
Particulates=pd.merge(Particulates,Renewable ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,gdp ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,fossilFules ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,gdp_df ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,Population_df ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,motorVehicle ,how='left',on=['Country','Year'])
Particulates=pd.merge(Particulates,ShareCars ,how='left',on=['Country','Year'])
# Particulates=pd.merge(Particulates,Natural_Gas ,how='left',on=['Country','Year'])
# Particulates=pd.merge(Particulates,Petroleum ,how='left',on=['Country','Year'])
```

### Create the CarbonMonoxide Dataframe

In [13]:

```
CarbonMonoxide=pd.merge(CarbonMonoxide,Renewable ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,gdp ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,fossilFules ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,gdp_df ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,Population_df ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,motorVehicle ,how='left',on=['Country','Year'])
CarbonMonoxide=pd.merge(CarbonMonoxide,ShareCars ,how='left',on=['Country','Year'])
# CarbonMonoxide=pd.merge(CarbonMonoxide,Natural_Gas ,how='left',on=['Country','Year'])
# CarbonMonoxide=pd.merge(CarbonMonoxide,Petroleum ,how='left',on=['Country','Year'])
```

### Create the SulphurOxides Dataframe

In [14]:

```
SulphurOxides=pd.merge(SulphurOxides,Renewable ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,gdp ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,fossilFules ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,gdp_df ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,Population_df ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,motorVehicle ,how='left',on=['Country','Year'])
SulphurOxides=pd.merge(SulphurOxides,ShareCars ,how='left',on=['Country','Year'])
# SulphurOxides=pd.merge(SulphurOxides,Natural_Gas ,how='left',on=['Country','Year'])
```

```
# SulphurOxides=pd.merge(SulphurOxides,Petroleum ,how='left',on=['Country','Year'])
```

In [15]:

```
NitrogenOxides.shape
```

Out[15]:

```
(1014, 11)
```

In [ ]:

In [16]:

```
Particulates.shape
```

Out[16]:

```
(790, 10)
```

In [17]:

```
CarbonMonoxide.shape
```

Out[17]:

```
(1012, 10)
```

In [18]:

```
SulphurOxides.shape
```

Out[18]:

```
(1014, 10)
```

## Drop all the NaN rows

In [19]:

```
NitrogenOxides.dropna(inplace=True)
Particulates.dropna(inplace=True)
CarbonMonoxide.dropna(inplace=True)
SulphurOxides.dropna(inplace=True)
```

In [ ]:

In [20]:

```
NitrogenOxides.tail(3)
```

Out[20]:

| | Country | Year | Nitrogen Oxides | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Population | Nitrogen_Oxides_next_yea |
|---|---|---|---|---|---|---|---|---|---|---|---|

| | Country | Year | Nitrogen Oxides | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Population | Nitrogen_Oxides_next_year |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 840 | Estonia | 2011 | 41.304 | 55.238560 | 0.049419 | [illegible] | 23287.18 | 12.366686 | 83.836988 | 1334947.0 | 38.30 |
| 841 | Estonia | 2012 | 38.303 | 7.748184 | 0.011514 | [illegible] | 24373.75 | [illegible] | [illegible] | 1329301.0 | 37.16 |
| 842 | Estonia | 2013 | [illegible] | 3.540916 | 0.209584 | [illegible] | [illegible] | [illegible] | [illegible] | 1320174.0 | 36.66 |

In [21]:

```
NitrogenOxides.shape
```

Out[21]:

```
(262, 11)
```

In [22]:

```
NitrogenOxides.corr()
```

Out[22]:

| | Year | Nitrogen Oxides | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Population | Nitrogen_( |
|---|---|---|---|---|---|---|---|---|---|---|
| Year | 1.000000 | -0.105119 | 0.246503 | 0.334394 | -0.182649 | 0.151053 | -0.085796 | 0.032979 | -0.052120 | |
| Nitrogen Oxides | -0.105119 | 1.000000 | -0.217935 | -0.069361 | 0.057545 | 0.192230 | 0.384326 | -0.530529 | 0.895075 | |
| Renewable energy public RD&D budget, % total energy public RD&D | 0.246503 | -0.217935 | 1.000000 | -0.315277 | -0.238163 | -0.070929 | 0.000481 | 0.234655 | -0.250452 | |
| Energy public RD&D budget, % GDP | 0.334394 | -0.069361 | -0.315277 | 1.000000 | -0.095947 | 0.205640 | -0.091063 | -0.090397 | -0.022940 | |
| Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | -0.182649 | 0.057545 | -0.238163 | -0.095947 | 1.000000 | 0.120068 | 0.073082 | -0.140860 | -0.032750 | |
| GDP per capita | 0.151053 | 0.192230 | -0.070929 | 0.205640 | 0.120068 | 1.000000 | 0.173940 | -0.123840 | 0.076553 | |
| Road traffic in thousand vehicle-km per road motor vehicle | -0.085796 | 0.384326 | 0.000481 | -0.091063 | 0.073082 | 0.173940 | 1.000000 | -0.090658 | 0.233104 | |
| Share of passenger cars in total road motor vehicle | 0.032979 | -0.530529 | 0.234655 | -0.090397 | -0.140860 | -0.123840 | -0.090658 | 1.000000 | -0.636985 | |
| Population | -0.052120 | 0.895075 | -0.250452 | -0.022940 | -0.032750 | 0.076553 | 0.233104 | -0.636985 | 1.000000 | |
| Nitrogen_Oxides_next_year | -0.109498 | 0.998481 | -0.217770 | -0.070055 | 0.060538 | 0.189657 | 0.384878 | -0.527754 | 0.890981 | |

## print every head of data fram

In [23]:

```
NitrogenOxides.head()
```

Out[23]:

| | Country | Year | Nitrogen Oxides | Renewable energy public RD&D budget, % total energy public | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public | GDP per capita | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Population | Nitrogen_Oxides_next_yea |
|---|---|---|---|---|---|---|---|---|---|---|---|

| | Country | Year | Nitrogen Oxides | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Population | Nitrogen_Oxides_next_year |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | Australia | 2001 | 1959.907 | 9.947913 | 0.020618 | 60.81924 | 37666.54 | 15.240487 | 78.833595 | 19275000.0 | 2039.96 |
| 13 | Australia | 2003 | 2121.206 | 10.604360 | 0.017622 | 54.48494 | 39481.01 | 15.955329 | 78.751045 | 19721000.0 | 2174.57 |
| 14 | Australia | 2004 | 2174.572 | 12.396670 | 0.034020 | 17.25762 | 40222.41 | 16.017882 | 78.541343 | 19933000.0 | 2194.33 |
| 15 | Australia | 2005 | 2194.335 | 12.423920 | 0.023397 | 22.94425 | 40734.01 | 15.764583 | 78.275862 | 20177000.0 | 2204.75 |
| 16 | Australia | 2006 | 2204.751 | 14.630740 | 0.024532 | 24.45360 | 41583.29 | 15.308378 | 77.923254 | 20451000.0 | 2223.92 |

In [24]:

```
Particulates.head(5)
```

Out[24]:

| | Country | Year | Particulates | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Population | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Austria | 2000 | 24.428 | 28.02075 | 0.010918 | 1.908070 | 37382.84 | 8011566.0 | 11.906964 | 73.295558 |
| 3 | Austria | 2001 | 24.713 | 26.53321 | 0.013546 | 2.119041 | 37723.44 | 8042293.0 | 11.881742 | 73.409783 |
| 5 | Austria | 2003 | 23.778 | 39.68057 | 0.010774 | 1.845329 | 38337.71 | 8118245.0 | 12.763979 | 73.574460 |
| 6 | Austria | 2004 | 23.273 | 28.46067 | 0.013837 | 1.344904 | 39190.00 | 8169441.0 | 12.740828 | 73.628769 |
| 7 | Austria | 2005 | 21.979 | 36.03274 | 0.013224 | 0.616071 | 39890.56 | 8225278.0 | 12.825143 | 73.690818 |

In [25]:

```
CarbonMonoxide.head(3)
```

Out[25]:

| | Country | Year | Carbon Monoxide | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Population | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | Australia | 2001 | 5121.711 | 9.947913 | 0.020618 | 60.81924 | 37666.54 | 19275000.0 | 15.240487 | 78.833595 |
| 13 | Australia | 2003 | 4499.161 | 10.604360 | 0.017622 | 54.48494 | 39481.01 | 19721000.0 | 15.955329 | 78.751045 |
| 14 | Australia | 2004 | 4593.240 | 12.396670 | 0.034020 | 17.25762 | 40222.41 | 19933000.0 | 16.017882 | 78.541343 |

In [26]:

```
SulphurOxides.head(3)
```

Out[26]:

| | Country | Year | Sulphur Oxides | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Population | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | Australia | 2001 | 2585.104 | 9.947913 | 0.020618 | 60.81924 | 37666.54 | 19275000.0 | 15.240487 | 78.833595 |
| 13 | Australia | 2003 | 2775.016 | 10.604360 | 0.017622 | 54.48494 | 39481.01 | 19721000.0 | 15.955329 | 78.751045 |
| 14 | Australia | 2004 | 2512.628 | 12.396670 | 0.034020 | 17.25762 | 40222.41 | 19933000.0 | 16.017882 | 78.541343 |

## Scale the data

In [27]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

#scaled NitrogenOxides
scaler.fit(NitrogenOxides.drop(columns=['Country']))
NitrogenOxides_scaled = pd.DataFrame(scaler.transform(NitrogenOxides.drop(columns=['Country'])), c
olumns=NitrogenOxides.drop(columns=['Country']).columns)
NitrogenOxides.reset_index(drop=True,inplace=True)
NitrogenOxides_scaled['Country']=NitrogenOxides['Country']
NitrogenOxides_scaled = NitrogenOxides_scaled[['Country','Year','Renewable energy public RD&D budg
et, % total energy public RD&D','Energy public RD&D budget, % GDP','Fossil fuel public RD&D budget
(excluding CCS), % total energy public RD&D','GDP per capita','Population','Road traffic in thousa
nd vehicle-km per road motor vehicle','Share of passenger cars in total road motor vehicle','Nitro
gen Oxides','Nitrogen_Oxides_next_year']]

#scaled Particulates
scaler.fit(Particulates.drop(columns=['Country']))
Particulates_scaled = pd.DataFrame(scaler.transform(Particulates.drop(columns=['Country'])),
columns=Particulates.drop(columns=['Country']).columns)
Particulates.reset_index(drop=True,inplace=True)
Particulates_scaled['Country']=Particulates['Country']
Particulates_scaled = Particulates_scaled[['Country','Year','Renewable energy public RD&D budget,
% total energy public RD&D','Energy public RD&D budget, % GDP','Fossil fuel public RD&D budget (ex
cluding CCS), % total energy public RD&D','GDP per capita','Population','Road traffic in thousand
vehicle-km per road motor vehicle','Share of passenger cars in total road motor
vehicle','Particulates']]

#scaled CarbonMonoxide
scaler.fit(CarbonMonoxide.drop(columns=['Country']))
CarbonMonoxide_scaled = pd.DataFrame(scaler.transform(CarbonMonoxide.drop(columns=['Country'])), c
olumns=CarbonMonoxide.drop(columns=['Country']).columns)
CarbonMonoxide.reset_index(drop=True,inplace=True)
CarbonMonoxide_scaled['Country']=CarbonMonoxide['Country']
CarbonMonoxide_scaled = CarbonMonoxide_scaled[['Country','Year','Renewable energy public RD&D budg
et, % total energy public RD&D','Energy public RD&D budget, % GDP','Fossil fuel public RD&D budget
(excluding CCS), % total energy public RD&D','GDP per capita','Population','Road traffic in thousa
nd vehicle-km per road motor vehicle','Share of passenger cars in total road motor vehicle','Carbo
n Monoxide']]

#scaled CarbonMonoxide
scaler.fit(SulphurOxides.drop(columns=['Country']))
SulphurOxides_scaled = pd.DataFrame(scaler.transform(SulphurOxides.drop(columns=['Country'])), col
umns=SulphurOxides.drop(columns=['Country']).columns)
SulphurOxides_scaled.reset_index(drop=True,inplace=True)
SulphurOxides_scaled['Country']=SulphurOxides['Country']
SulphurOxides_scaled = SulphurOxides_scaled[['Country','Year','Renewable energy public RD&D budget
, % total energy public RD&D','Energy public RD&D budget, % GDP','Fossil fuel public RD&D budget (
excluding CCS), % total energy public RD&D','GDP per capita','Population','Road traffic in thousan
d vehicle-km per road motor vehicle','Share of passenger cars in total road motor vehicle','Sulphu
r Oxides']]
```

# Remane the columns

In [28]:

```
NitrogenOxides_scaled.rename(columns={'Renewable energy public RD&D budget, % total energy public
RD&D':'Renewable_energy',
                                       'Energy public RD&D budget, % GDP':'Energy_GDP',
                                       'Fossil fuel public RD&D budget (excluding CCS), % total ene
y public RD&D':'Fossil_Energy',
                                       'GDP per capita':'GDP_per_capita',
                                       'Road traffic in thousand vehicle-km per road motor vehicle':
Road_Traffic',
                                       'Share of passenger cars in total road motor vehicle':'Coope
tive_Vehicles',
                                       'Nitrogen Oxides':'Nitrogen_Oxides'},inplace=True)

Particulates_scaled.rename(columns={'Renewable energy public RD&D budget, % total energy public RD
&D':'Renewable_energy',
                                     'Energy public RD&D budget, % GDP':'Energy_GDP',
                                     'Fossil fuel public RD&D budget (excluding CCS), % total ene
y public RD&D':'Fossil_Energy',
```

```
                                        'GDP per capita':'GDP_per_capita',
                                        'Road traffic in thousand vehicle-km per road motor vehicle':
Road_Traffic',
                                        'Share of passenger cars in total road motor vehicle':'Coope
tive_Vehicles'
                                         },inplace=True)

CarbonMonoxide_scaled.rename(columns={'Renewable energy public RD&D budget, % total energy public
RD&D':'Renewable_energy',
                                        'Energy public RD&D budget, % GDP':'Energy_GDP',
                                        'Fossil fuel public RD&D budget (excluding CCS), % total ene
y public RD&D':'Fossil_Energy',
                                        'GDP per capita':'GDP_per_capita',
                                        'Road traffic in thousand vehicle-km per road motor vehicle':
Road_Traffic',
                                        'Share of passenger cars in total road motor vehicle':'Coope
tive_Vehicles',
                                        'Carbon Monoxide':'Carbon_Monoxide'},inplace=True)
SulphurOxides_scaled.rename(columns={'Renewable energy public RD&D budget, % total energy public R
D&D':'Renewable_energy',
                                        'Energy public RD&D budget, % GDP':'Energy_GDP',
                                        'Fossil fuel public RD&D budget (excluding CCS), % total ene
y public RD&D':'Fossil_Energy',
                                        'GDP per capita':'GDP_per_capita',
                                        'Road traffic in thousand vehicle-km per road motor vehicle':
Road_Traffic',
                                        'Share of passenger cars in total road motor vehicle':'Coope
tive_Vehicles',
                                        'Sulphur Oxides':'Sulphur_Oxides'},inplace=True)
```

# Run Linear regresion

## Population dofek hakol

In [ ]:

In [29]:

```
NitrogenOxides_scaled.head(1)
```

Out[29]:

| | Country | Year | Renewable_energy | Energy_GDP | Fossil_Energy | GDP_per_capita | Population | Road_Traffic | Cooperative_Vehicles |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Australia | -1.53002 | -0.865897 | -0.578173 | 4.333778 | -0.072309 | -0.411762 | 0.70873 | 0.578403 |

In [30]:

```
import statsmodels.formula.api as smf
linereg_Nitorgen =
smf.ols('Nitrogen_Oxides~Renewable_energy+Road_Traffic+Cooperative_Vehicles+Energy_GDP+GDP_per_capi
,data=NitrogenOxides_scaled).fit()
linereg_Nitorgen.summary()
```

Out[30]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Nitrogen_Oxides | **R-squared:** | 0.430 |
| **Model:** | OLS | **Adj. R-squared:** | 0.419 |
| **Method:** | Least Squares | **F-statistic:** | 38.66 |
| **Date:** | Sun, 26 Jan 2020 | **Prob (F-statistic):** | 1.69e-29 |

| | | | |
|---|---|---|---|
| **Time:** | 13:36:17 | **Log-Likelihood:** | -298.07 |
| **No. Observations:** | 262 | **AIC:** | 608.1 |
| **Df Residuals:** | 256 | **BIC:** | 629.5 |
| **Df Model:** | 5 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -8.63e-17 | 0.047 | -1.83e-15 | 1.000 | -0.093 | 0.093 |
| **Renewable_energy** | -0.1486 | 0.051 | -2.916 | 0.004 | -0.249 | -0.048 |
| **Road_Traffic** | 0.3106 | 0.048 | 6.407 | 0.000 | 0.215 | 0.406 |
| **Cooperative_Vehicles** | -0.4687 | 0.049 | -9.570 | 0.000 | -0.565 | -0.372 |
| **Energy_GDP** | -0.1510 | 0.051 | -2.953 | 0.003 | -0.252 | -0.050 |
| **GDP_per_capita** | 0.1007 | 0.049 | 2.038 | 0.043 | 0.003 | 0.198 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 110.127 | **Durbin-Watson:** | 0.223 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 476.224 |
| **Skew:** | 1.711 | **Prob(JB):** | 3.88e-104 |
| **Kurtosis:** | 8.649 | **Cond. No.** | 1.62 |

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

In [31]:

```
# X=NitrogenOxides_scaled.drop(columns=
['Country','Year','Nitrogen_Oxides','Energy_GDP','Cooperative_Vehicles','Fossil_Energy','Nitrogen_(
s_next_year'])
X=NitrogenOxides_scaled[['Renewable_energy','Road_Traffic','Cooperative_Vehicles','GDP_per_capita'
,'Energy_GDP']]

y=NitrogenOxides_scaled['Nitrogen_Oxides']
```

In [32]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()  # define our model using least square method
lm.fit(X_train,y_train)
y_predict = lm.predict(X_test)
test = pd.DataFrame({'Actual':y_test,'Predict':y_predict})
test.head(2)
```

Out[32]:

| | Actual | Predict |
|---|---|---|
| 90 | 0.063275 | -0.291638 |
| 185 | -0.396404 | 0.908928 |

In [33]:

```
from sklearn.metrics import r2_score
r2_score(y_test, y_predict)
```

Out[33]:

```
-5.988320928173019
```

In [34]:

```
from sklearn import metrics
print("MSE:",metrics.mean_squared_error(test.Actual, test.Predict))
print("RMSE:",np.sqrt(metrics.mean_squared_error(test.Actual, test.Predict)))
print("MAE:",metrics.mean_absolute_error(test.Actual, test.Predict))
```

```
MSE: 0.37480072364994754
RMSE: 0.6122097056156065
MAE: 0.4984510591029204
```

## linear Regression for Particulates

In [35]:

```
Particulates_scaled.head(1)
```

Out[35]:

| | Country | Year | Renewable_energy | Energy_GDP | Fossil_Energy | GDP_per_capita | Population | Road_Traffic | Cooperative_Vehicles |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Austria | -<br>1.747735 | 0.411291 | -0.831544 | -0.626339 | -0.15062 | -0.51589 | 0.066942 | -0.169422 |

In [36]:

```
linereg_Particulates =
smf.ols('Particulates~Renewable_energy+Road_Traffic',data=Particulates_scaled).fit()
linereg_Particulates.summary()
```

Out[36]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Particulates | **R-squared:** | 0.236 |
| **Model:** | OLS | **Adj. R-squared:** | 0.229 |
| **Method:** | Least Squares | **F-statistic:** | 33.73 |
| **Date:** | Sun, 26 Jan 2020 | **Prob (F-statistic):** | 1.70e-13 |
| **Time:** | 13:36:18 | **Log-Likelihood:** | -285.20 |
| **No. Observations:** | 222 | **AIC:** | 576.4 |
| **Df Residuals:** | 219 | **BIC:** | 586.6 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 1.006e-16 | 0.059 | 1.7e-15 | 1.000 | -0.116 | 0.116 |
| **Renewable_energy** | -0.2413 | 0.059 | -4.084 | 0.000 | -0.358 | -0.125 |
| **Road_Traffic** | 0.4211 | 0.059 | 7.127 | 0.000 | 0.305 | 0.538 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 115.190 | **Durbin-Watson:** | 0.304 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 463.192 |
| **Skew:** | 2.186 | **Prob(JB):** | 2.62e-101 |
| **Kurtosis:** | 8.564 | **Cond. No.** | 1.00 |

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

In [37]:

```
X=Particulates_scaled.drop(columns=['Country','Particulates','GDP_per_capita'
,'Population','Energy_GDP','Cooperative_Vehicles'])
```

```
y=Particulates_scaled['Particulates']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()  # define our model using least square method
lm.fit(X_train,y_train)
y_predict = lm.predict(X_test)
test = pd.DataFrame({'Actual':y_test,'Predict':y_predict})
test.head(2)
```

Out[37]:

| | Actual | Predict |
|---|---|---|
| 144 | -0.368126 | 0.465646 |
| 114 | -0.387279 | 0.323158 |

In [38]:

```
from sklearn import metrics
print("MSE:",metrics.mean_squared_error(test.Actual, test.Predict))
print("RMSE:",np.sqrt(metrics.mean_squared_error(test.Actual, test.Predict)))
print("MAE:",metrics.mean_absolute_error(test.Actual, test.Predict))
```

```
MSE: 0.6351292917582799
RMSE: 0.7969499932607315
MAE: 0.566447944618793
```

## linear Regression for CarbonMonoxide

In [39]:

```
linereg_CarbonMonoxide =
smf.ols('Carbon_Monoxide~Renewable_energy+Energy_GDP+Fossil_Energy+GDP_per_capita+Road_Traffic+Coop
ive_Vehicles',data=CarbonMonoxide_scaled).fit()
linereg_CarbonMonoxide.summary()
```

Out[39]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Carbon_Monoxide | **R-squared:** | 0.423 |
| **Model:** | OLS | **Adj. R-squared:** | 0.409 |
| **Method:** | Least Squares | **F-statistic:** | 31.15 |
| **Date:** | Sun, 26 Jan 2020 | **Prob (F-statistic):** | 5.44e-28 |
| **Time:** | 13:36:18 | **Log-Likelihood:** | -299.73 |
| **No. Observations:** | 262 | **AIC:** | 613.5 |
| **Df Residuals:** | 255 | **BIC:** | 638.4 |
| **Df Model:** | 6 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 7.286e-17 | 0.048 | 1.53e-15 | 1.000 | -0.094 | 0.094 |
| **Renewable_energy** | -0.1678 | 0.053 | -3.150 | 0.002 | -0.273 | -0.063 |
| **Energy_GDP** | -0.1811 | 0.053 | -3.436 | 0.001 | -0.285 | -0.077 |
| **Fossil_Energy** | -0.1182 | 0.051 | -2.336 | 0.020 | -0.218 | -0.019 |
| **GDP_per_capita** | 0.1318 | 0.050 | 2.623 | 0.009 | 0.033 | 0.231 |
| **Road_Traffic** | 0.2891 | 0.049 | 5.910 | 0.000 | 0.193 | 0.385 |
| **Cooperative_Vehicles** | -0.4754 | 0.050 | -9.598 | 0.000 | -0.573 | -0.378 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 118.331 | **Durbin-Watson:** | 0.259 |

| | | | |
|---|---|---|---|
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 566.352 |
| **Skew:** | 1.817 | **Prob(JB):** | 1.04e-123 |
| **Kurtosis:** | 9.218 | **Cond. No.** | 1.80 |

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

In [40]:

```
X=CarbonMonoxide_scaled.drop(columns=['Country','Carbon_Monoxide'])
y=CarbonMonoxide_scaled['Carbon_Monoxide']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()  # define our model using least square method
lm.fit(X_train,y_train)
y_predict = lm.predict(X_test)
test = pd.DataFrame({'Actual':y_test,'Predict':y_predict})
test.head(3)
```

Out[40]:

| | Actual | Predict |
|---|---|---|
| 90 | -0.057522 | 0.648027 |
| 185 | -0.329689 | -0.216553 |
| 171 | -0.326522 | -0.266224 |

In [41]:

```
from sklearn import metrics
print("MSE:",metrics.mean_squared_error(test.Actual, test.Predict))
print("RMSE:",np.sqrt(metrics.mean_squared_error(test.Actual, test.Predict)))
print("MAE:",metrics.mean_absolute_error(test.Actual, test.Predict))
```

```
MSE: 0.18178895121855232
RMSE: 0.4263671554171971
MAE: 0.31084645308675196
```

## linear Regression for SulphurOxides

In [42]:

```
linereg_Sulphur =
smf.ols('Sulphur_Oxides~Renewable_energy+Energy_GDP+Fossil_Energy+GDP_per_capita+Road_Traffic+Coope
ve_Vehicles',data=SulphurOxides_scaled).fit()
linereg_Sulphur.summary()
```

Out[42]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Sulphur_Oxides | **R-squared:** | 0.397 |
| **Model:** | OLS | **Adj. R-squared:** | 0.383 |
| **Method:** | Least Squares | **F-statistic:** | 27.98 |
| **Date:** | Sun, 26 Jan 2020 | **Prob (F-statistic):** | 1.31e-25 |
| **Time:** | 13:36:18 | **Log-Likelihood:** | -305.50 |
| **No. Observations:** | 262 | **AIC:** | 625.0 |
| **Df Residuals:** | 255 | **BIC:** | 650.0 |
| **Df Model:** | 6 | | |
| **Covariance Type:** | nonrobust | | |

| Covariance Type: | nonrobust | | |
|---|---|---|---|

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -5.551e-17 | 0.049 | -1.14e-15 | 1.000 | -0.096 | 0.096 |
| **Renewable_energy** | -0.1588 | 0.054 | -2.916 | 0.004 | -0.266 | -0.052 |
| **Energy_GDP** | -0.1792 | 0.054 | -3.326 | 0.001 | -0.285 | -0.073 |
| **Fossil_Energy** | -0.0023 | 0.052 | -0.044 | 0.965 | -0.104 | 0.100 |
| **GDP_per_capita** | 0.0460 | 0.051 | 0.895 | 0.372 | -0.055 | 0.147 |
| **Road_Traffic** | 0.3148 | 0.050 | 6.296 | 0.000 | 0.216 | 0.413 |
| **Cooperative_Vehicles** | -0.4419 | 0.051 | -8.728 | 0.000 | -0.542 | -0.342 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 144.451 | **Durbin-Watson:** | 0.276 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 965.204 |
| **Skew:** | 2.163 | **Prob(JB):** | 2.56e-210 |
| **Kurtosis:** | 11.349 | **Cond. No.** | 1.80 |

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

In [43]:

```
X=SulphurOxides_scaled.drop(columns=['Country','Sulphur_Oxides'])
y=SulphurOxides_scaled['Sulphur_Oxides']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()  # define our model using least square method
lm.fit(X_train,y_train)
y_predict = lm.predict(X_test)
test = pd.DataFrame({'Actual':y_test,'Predict':y_predict})
test.head(3)
```

Out[43]:

| | Actual | Predict |
|---|---|---|
| 90 | -0.166708 | 0.627932 |
| 185 | -0.420933 | 0.020005 |
| 171 | -0.415018 | -0.333384 |

In [44]:

```
from sklearn import metrics
print("MSE:",metrics.mean_squared_error(test.Actual, test.Predict))
print("RMSE:",np.sqrt(metrics.mean_squared_error(test.Actual, test.Predict)))
print("MAE:",metrics.mean_absolute_error(test.Actual, test.Predict))
```

```
MSE: 0.18499703125746222
RMSE: 0.43011281224518555
MAE: 0.35606527827135565
```

# classifiction by median

In [45]:

```
Particulates["Particulates_per_capita"]=(Particulates["Particulates"]*1000)/Particulates["Populatic
n"]
med = Particulates["Particulates_per_capita"].median()
```

In [46]:

```
Particulates.head(1)
```

Out[46]:

| | Country | Year | Particulates | Renewable energy public RD&D budget, % total energy public RD&D | Energy public RD&D budget, % GDP | Fossil fuel public RD&D budget (excluding CCS), % total energy public RD&D | GDP per capita | Population | Road traffic in thousand vehicle-km per road motor vehicle | Share of passenger cars in total road motor vehicle | Particulates_per_capita |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Austria | 2000 | 24.428 | 28.02075 | 0.010918 | 1.90807 | 37382.84 | 8011566.0 | 11.906964 | 73.295558 | 0.003049 |

In [47]:

```
Particulates["enrgy_gdp"]=Particulates["Energy public RD&D budget, % GDP"]*Particulates["GDP per c
apita"]
Particulates["High_Emissions"]=np.where(Particulates["Particulates_per_capita"]>med,1,0)
X=Particulates.drop(columns=['Country','Year','Particulates'
,'Particulates_per_capita','Population','High_Emissions','Energy public RD&D budget, % GDP','Road
traffic in thousand vehicle-km per road motor vehicle','GDP per capita'])
y=Particulates['High_Emissions']
```

In [48]:

```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# train a logistic regression model on the training set
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X_train, y_train)

# make class predictions for the testing set
y_pred_test = logreg.predict(X_test)

# calculate accuracy
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred_test))
```

```
0.6785714285714286
```

In [49]:

```
# print the first 25 true and predicted responses
print('Actual    :', y_test.values[0:25])
print('Predicted:', y_pred_test[0:25])
```

```
Actual    : [0 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0]
Predicted: [1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1]
```

In [50]:

```
print(metrics.confusion_matrix(y_test, y_pred_test))
```

```
[[24  7]
 [11 14]]
```

In [51]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

           0       0.69      0.77      0.73        31
           1       0.67      0.56      0.61        25

    accuracy                           0.68        56
   macro avg       0.68      0.67      0.67        56
weighted avg       0.68      0.68      0.67        56
```

# linear reg predict on next year

In [52]:

```
NitrogenOxides_scaled.head()
```

Out[52]:

| | Country | Year | Renewable_energy | Energy_GDP | Fossil_Energy | GDP_per_capita | Population | Road_Traffic | Cooperative_Vehicles |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Australia | -1.530020 | -0.865897 | -0.578173 | 4.333778 | -0.072309 | -0.411762 | 0.708730 | 0.578403 |
| 1 | Australia | -1.037859 | -0.819696 | -0.676524 | 3.795664 | 0.103205 | -0.405455 | 0.855484 | 0.568913 |
| 2 | Australia | -0.791778 | -0.693553 | -0.138231 | 0.633113 | 0.174921 | -0.402457 | 0.868326 | 0.544805 |
| 3 | Australia | -0.545698 | -0.691635 | -0.486924 | 1.116206 | 0.224408 | -0.399007 | 0.816325 | 0.514284 |
| 4 | Australia | -0.299617 | -0.536318 | -0.449682 | 1.244429 | 0.306559 | -0.395132 | 0.722668 | 0.473747 |

In [53]:

```
linereg_Nitorgen =
smf.ols('Nitrogen_Oxides_next_year~Renewable_energy+Road_Traffic+Cooperative_Vehicles+Energy_GDP+GI
r_capita',data=NitrogenOxides_scaled).fit()
linereg_Nitorgen.summary()
```

Out[53]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Nitrogen_Oxides_next_year | **R-squared:** | 0.428 |
| **Model:** | OLS | **Adj. R-squared:** | 0.416 |
| **Method:** | Least Squares | **F-statistic:** | 38.25 |
| **Date:** | Sun, 26 Jan 2020 | **Prob (F-statistic):** | 3.01e-29 |
| **Time:** | 13:36:18 | **Log-Likelihood:** | -298.67 |
| **No. Observations:** | 262 | **AIC:** | 609.3 |
| **Df Residuals:** | 256 | **BIC:** | 630.7 |
| **Df Model:** | 5 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | -3.903e-17 | 0.047 | -8.25e-16 | 1.000 | -0.093 | 0.093 |
| **Renewable_energy** | -0.1492 | 0.051 | -2.922 | 0.004 | -0.250 | -0.049 |
| **Road_Traffic** | 0.3119 | 0.049 | 6.418 | 0.000 | 0.216 | 0.408 |
| **Cooperative_Vehicles** | -0.4660 | 0.049 | -9.493 | 0.000 | -0.563 | -0.369 |
| **Energy_GDP** | -0.1510 | 0.051 | -2.947 | 0.004 | -0.252 | -0.050 |
| **GDP_per_capita** | 0.0982 | 0.050 | 1.983 | 0.048 | 0.001 | 0.196 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 118.719 | **Durbin-Watson:** | 0.221 |

| | | | |
|---|---|---|---|
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 588.445 |
| **Skew:** | 1.809 | **Prob(JB):** | 1.66e-128 |
| **Kurtosis:** | 9.389 | **Cond. No.** | 1.62 |

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

In [54]:

```
X=NitrogenOxides_scaled[['Renewable_energy','Road_Traffic','Cooperative_Vehicles','GDP_per_capita'
,'Energy_GDP','Population']]
y=NitrogenOxides_scaled['Nitrogen_Oxides_next_year']
```

In [55]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()  # define our model using least square method
lm.fit(X_train,y_train)
y_predict = lm.predict(X_test)
test = pd.DataFrame({'Actual':y_test,'Predict':y_predict})
test.head(2)
```

Out[55]:

| | Actual | Predict |
|---|---|---|
| 90 | 0.059321 | 0.530067 |
| 185 | -0.397975 | -0.318939 |

In [ ]:

In [56]:

```
from sklearn import metrics
print("MSE:",metrics.mean_squared_error(test.Actual, test.Predict))
print("RMSE:",np.sqrt(metrics.mean_squared_error(test.Actual, test.Predict)))
print("MAE:",metrics.mean_absolute_error(test.Actual, test.Predict))
```

```
MSE: 0.125999437901221
RMSE: 0.3549639952181362
MAE: 0.2716689137851464
```

In [ ]:

In [ ]: