```
#pandas: It used for data manipulation and analysis.
#numPy: It is a powerful Python library for numerical computing.
#seaborn: It provides a high-level interface for creating attractive and informative statistical graphics.
#matplotlib.pyplot: It provides a MATLAB-like interface for creating basic plots and visualizations.

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


#pd.read_csv is used to read the data from the csv files
#df is the variable in which the data is getting stored from the csv files

df=pd.read_csv("Agrofood_co2_emission.csv")


df
```

ıs ıg	Food Transport	Forestland	...	Manure Management	Fires in organic soils	Fires in humid tropical forests	On-farm energy use	Rural population	Urban population	Tota Populatio - Mal
;3	63.1152	-2388.8030	...	319.1763	0.0	0.0	NaN	9655167.0	2593947.0	5348387.
'3	61.2125	-2388.8030	...	342.3079	0.0	0.0	NaN	10230490.0	2763167.0	5372959.
'3	53.3170	-2388.8030	...	349.1224	0.0	0.0	NaN	10995568.0	2985663.0	6028494.
'3	54.3617	-2388.8030	...	352.2947	0.0	0.0	NaN	11858090.0	3237009.0	7003641.
'3	53.9874	-2388.8030	...	367.6784	0.0	0.0	NaN	12690115.0	3482604.0	7733458.
..	...	...	...	...	...	...	...	...	...	.
)0	251.1465	76500.2982	...	282.5994	0.0	0.0	417.3150	10934468.0	5215894.0	6796658.
)0	255.7975	76500.2982	...	255.5900	0.0	0.0	398.1644	11201138.0	5328766.0	6940631.
)0	327.0897	76500.2982	...	257.2735	0.0	0.0	465.7735	11465748.0	5447513.0	7086002.
)0	290.1893	76500.2982	...	267.5224	0.0	0.0	444.2335	11725970.0	5571525.0	7231989.
)0	238.7639	76500.2982	...	266.7316	0.0	0.0	444.2335	11980005.0	5700460.0	7385220.

```
#head() is a function used to read the starting 5 lines from the datasets

df.head()
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Food Transport	Forestl
0	Afghanistan	1990	14.7237	0.0557	205.6077	686.00	0.0	11.807483	63.1152	-2388.8
1	Afghanistan	1991	14.7237	0.0557	209.4971	678.16	0.0	11.712073	61.2125	-2388.8
2	Afghanistan	1992	14.7237	0.0557	196.5341	686.00	0.0	11.712073	53.3170	-2388.8
3	Afghanistan	1993	14.7237	0.0557	230.8175	686.00	0.0	11.712073	54.3617	-2388.8
4	Afghanistan	1994	14.7237	0.0557	242.0494	705.60	0.0	11.712073	53.9874	-2388.8

5 rows × 31 columns

```
#tail() function is used to read the ending 5 lines from the dataset

df.tail()
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Food Transport	For
6960	Zimbabwe	2016	1190.0089	232.5068	70.9451	7.4088	0.0	75.0	251.1465	76
6961	Zimbabwe	2017	1431.1407	131.1324	108.6262	7.9458	0.0	67.0	255.7975	76
6962	Zimbabwe	2018	1557.5830	221.6222	109.9835	8.1399	0.0	66.0	327.0897	76
6963	Zimbabwe	2019	1591.6049	171.0262	45.4574	7.8322	0.0	73.0	290.1893	76
6964	Zimbabwe	2020	481.9027	48.4197	108.3022	7.9733	0.0	73.0	238.7639	76

5 rows × 31 columns

```
#sample(8) function is used to get the sample of 8 rows from the given dataset

df.sample(8)
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Food Transport	F
824	Botswana	2016	1129.2992	1.0359	4.5453	246.407276	588.3829	1.000000	248.8509	
3958	Martinique	2016	0.0000	0.0000	NaN	764.892810	0.0000	12.437201	96.9546	
4920	Papua New Guinea	2016	62.0055	45.8275	0.8513	2.710000	21129.7403	5.000000	173.7833	
230	Antigua and Barbuda	2003	0.0000	0.0000	0.0028	246.407276	0.0000	0.000000	17.3753	
1722	Czechia	2020	0.0000	0.0000	476.7076	813.956265	264.6027	88.000000	1570.8256	·
73	Algeria	2001	43.3120	5.7166	219.7063	2.649900	0.0000	43.000000	1388.9576	·
1854	Dominica	1994	0.0000	0.0000	0.0110	448.357467	0.0000	11.481085	4.9292	
4226	Morocco	1991	2.5060	1.6759	633.9108	50.176000	0.0000	298.000000	776.0586	·

8 rows × 31 columns

```
# dtypes:It is used to get the datatype of the specifi ed columns from the given datasets
df.dtypes
```

```
Area                              object
Year                               int64
Savanna fires                    float64
Forest fires                     float64
Crop Residues                    float64
Rice Cultivation                 float64
Drained organic soils (CO2)      float64
Pesticides Manufacturing         float64
Food Transport                   float64
Forestland                       float64
Net Forest conversion            float64
Food Household Consumption       float64
Food Retail                      float64
On-farm Electricity Use          float64
Food Packaging                   float64
Agrifood Systems Waste Disposal  float64
Food Processing                  float64
Fertilizers Manufacturing        float64
IPPU                             float64
Manure applied to Soils          float64
Manure left on Pasture           float64
Manure Management                float64
Fires in organic soils           float64
Fires in humid tropical forests  float64
On-farm energy use               float64
Rural population                 float64
```

```
Urban population                  float64
Total Population - Male           float64
Total Population - Female         float64
total_emission                    float64
Average Temperature °C            float64
dtype: object
```

```
# info():It is used to get the Information of about each and every columns
# It will also tell about the no. of the cells which are non-empty by displaying their number

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6965 entries, 0 to 6964
Data columns (total 31 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Area                              6965 non-null   object 
 1   Year                              6965 non-null   int64  
 2   Savanna fires                     6934 non-null   float64
 3   Forest fires                      6872 non-null   float64
 4   Crop Residues                     5576 non-null   float64
 5   Rice Cultivation                  6965 non-null   float64
 6   Drained organic soils (CO2)       6965 non-null   float64
 7   Pesticides Manufacturing          6965 non-null   float64
 8   Food Transport                    6965 non-null   float64
 9   Forestland                        6472 non-null   float64
 10  Net Forest conversion             6472 non-null   float64
 11  Food Household Consumption        6492 non-null   float64
 12  Food Retail                       6965 non-null   float64
 13  On-farm Electricity Use           6965 non-null   float64
 14  Food Packaging                    6965 non-null   float64
 15  Agrifood Systems Waste Disposal   6965 non-null   float64
 16  Food Processing                   6965 non-null   float64
 17  Fertilizers Manufacturing         6965 non-null   float64
 18  IPPU                              6222 non-null   float64
 19  Manure applied to Soils           6037 non-null   float64
 20  Manure left on Pasture            6965 non-null   float64
 21  Manure Management                 6037 non-null   float64
 22  Fires in organic soils            6965 non-null   float64
 23  Fires in humid tropical forests   6810 non-null   float64
 24  On-farm energy use                6009 non-null   float64
 25  Rural population                  6965 non-null   float64
 26  Urban population                  6965 non-null   float64
 27  Total Population - Male           6965 non-null   float64
 28  Total Population - Female         6965 non-null   float64
 29  total_emission                    6965 non-null   float64
 30  Average Temperature °C            6965 non-null   float64
dtypes: float64(29), int64(1), object(1)
memory usage: 1.6+ MB
```

```
# count():It is used to count the non-empty cells of each column
# It will display the number of that sooo...

df.count()
```

```
Area                              6965
Year                              6965
Savanna fires                     6934
Forest fires                      6872
Crop Residues                     5576
Rice Cultivation                  6965
Drained organic soils (CO2)       6965
Pesticides Manufacturing          6965
Food Transport                    6965
Forestland                        6472
Net Forest conversion             6472
Food Household Consumption        6492
Food Retail                       6965
On-farm Electricity Use           6965
Food Packaging                    6965
Agrifood Systems Waste Disposal   6965
Food Processing                   6965
Fertilizers Manufacturing         6965
IPPU                              6222
Manure applied to Soils           6037
Manure left on Pasture            6965
Manure Management                 6037
Fires in organic soils            6965
Fires in humid tropical forests   6810
On-farm energy use                6009
Rural population                  6965
```

```
Urban population                  6965
Total Population - Male           6965
Total Population - Female         6965
total_emission                    6965
Average Temperature °C            6965
dtype: int64
```

```
# shape:It is used to determine the dimensions or the size of an array or DataFrame.

df.shape
```

```
(6965, 31)
```

```
# columns:It will display the column names

df.columns
```

```
Index(['Area', 'Year', 'Savanna fires', 'Forest fires', 'Crop Residues',
       'Rice Cultivation', 'Drained organic soils (CO2)',
       'Pesticides Manufacturing', 'Food Transport', 'Forestland',
       'Net Forest conversion', 'Food Household Consumption', 'Food Retail',
       'On-farm Electricity Use', 'Food Packaging',
       'Agrifood Systems Waste Disposal', 'Food Processing',
       'Fertilizers Manufacturing', 'IPPU', 'Manure applied to Soils',
       'Manure left on Pasture', 'Manure Management', 'Fires in organic soils',
       'Fires in humid tropical forests', 'On-farm energy use',
       'Rural population', 'Urban population', 'Total Population - Male',
       'Total Population - Female', 'total_emission',
       'Average Temperature °C'],
      dtype='object')
```

```
# mean(): It is used to fi nd the mean of the required column data

df['total_emission'].mean()
```

```
64091.24414739476
```

```
# median(): It is used to fi nd the median of the required column data

df['total_emission'].median()
```

```
12147.654319071926
```

```
# std(): It is used to fi nd the Standard Deviation of the required column data

df['total_emission'].std()
```

```
228312.95795654855
```

```
# mode(): It is used to fi nd the mode of the required column data

df['total_emission'].mode()
```

```
0    7251.256946
Name: total_emission, dtype: float64
```

```
# describe(): It is used to describe the dataset into the count,mean,std,min,25%,50%(median),75%,max
# It will only describe and fi nd the values of the dataset columns which are in float or integer

df.describe()
```

	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticide Manufacturin
count	6965.000000	6934.000000	6872.000000	5576.000000	6965.000000	6965.000000	6965.00000
mean	2005.124910	1188.390893	919.302167	998.706309	4259.666673	3503.228636	333.41839
std	8.894665	5246.287783	3720.078752	3700.345330	17613.825187	15861.445678	1429.15936

```
# value_counts(): It will count the no. of times the specfi c column name present in the dataset

df["Area"].value_counts()
```

```
Afghanistan                       31
Isle of Man                       31
Montserrat                        31
Morocco                           31
Mozambique                        31
                                  ..
Czechoslovakia                     3
Ethiopia PDR                       3
USSR                               2
Yugoslav SFR                       2
Pacific Islands Trust Territory    1
Name: Area, Length: 236, dtype: int64
```

```
# unique(): It is used to count the unique enteries in that column

df["Area"].unique()
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'American Samoa', 'Andorra',
       'Angola', 'Anguilla', 'Antigua and Barbuda', 'Argentina',
       'Armenia', 'Aruba', 'Australia', 'Austria', 'Azerbaijan',
       'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus',
       'Belgium', 'Belgium-Luxembourg', 'Belize', 'Benin', 'Bermuda',
       'Bhutan', 'Bolivia (Plurinational State of)',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei Darussalam', 'Bulgaria',
       'Burkina Faso', 'Burundi', 'Cabo Verde', 'Cambodia', 'Cameroon',
       'Canada', 'Cayman Islands', 'Central African Republic', 'Chad',
       'Channel Islands', 'Chile', 'China', 'China, Hong Kong SAR',
       'China, Macao SAR', 'China, mainland', 'China, Taiwan Province of',
       'Colombia', 'Comoros', 'Congo', 'Cook Islands', 'Costa Rica',
       'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Czechoslovakia',
       "Democratic People's Republic of Korea",
       'Democratic Republic of the Congo', 'Denmark', 'Djibouti',
       'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt',
       'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Eswatini', 'Ethiopia', 'Ethiopia PDR',
       'Falkland Islands (Malvinas)', 'Faroe Islands', 'Fiji', 'Finland',
       'France', 'French Polynesia', 'Gabon', 'Gambia', 'Georgia',
       'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland', 'Grenada',
       'Guadeloupe', 'Guam', 'Guatemala', 'Guinea', 'Guinea-Bissau',
       'Guyana', 'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland',
       'India', 'Indonesia', 'Iran (Islamic Republic of)', 'Iraq',
       'Ireland', 'Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan',
       'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kuwait',
       'Kyrgyzstan', "Lao People's Democratic Republic", 'Latvia',
       'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Liechtenstein',
       'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia',
       'Maldives', 'Mali', 'Malta', 'Marshall Islands', 'Martinique',
       'Mauritania', 'Mauritius', 'Mayotte', 'Mexico',
       'Micronesia (Federated States of)', 'Monaco', 'Mongolia',
       'Montenegro', 'Montserrat', 'Morocco', 'Mozambique', 'Myanmar',
       'Namibia', 'Nauru', 'Nepal', 'Netherlands (Kingdom of the)',
       'Netherlands Antilles (former)', 'New Caledonia', 'New Zealand',
       'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'North Macedonia',
       'Northern Mariana Islands', 'Norway', 'Oman',
       'Pacific Islands Trust Territory', 'Pakistan', 'Palau',
       'Palestine', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
       'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Qatar',
       'Republic of Korea', 'Republic of Moldova', 'Romania',
       'Russian Federation', 'Rwanda',
       'Saint Helena, Ascension and Tristan da Cunha',
       'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines',
       'Samoa', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia',
       'Senegal', 'Serbia', 'Serbia and Montenegro', 'Seychelles',
```

```
'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Sudan (former)', 'Suriname',
'Sweden', 'Switzerland', 'Syrian Arab Republic', 'Tajikistan',
'Thailand', 'Timor-Leste', 'Togo', 'Tokelau', 'Tonga',
'Trinidad and Tobago', 'Tunisia', 'Turkmenistan',
'Turks and Caicos Islands', 'Tuvalu', 'Uganda', 'Ukraine',
'United Arab Emirates',
'United Kingdom of Great Britain and Northern Ireland',
```

```
#sort_values(): It will sort the values and provides u the data in the sorting manner
# If ascending of descending is not given then by default it will give u the data in the ascending manner/order only

df.sort_values(["Year"])
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Food Transport
0	Afghanistan	1990	14.7237	0.0557	205.6077	686.000000	0.0000	11.807483	63.1152
3604	Libya	1990	0.5397	0.0000	28.5588	248.061497	0.0000	83.000000	692.9556
3635	Liechtenstein	1990	0.0000	0.0000	NaN	9239.011226	0.0000	11.481085	0.0001
3716	Madagascar	1990	2615.7130	659.3090	227.7026	5772.435200	4239.3751	3.000000	55.3162
3747	Malawi	1990	281.4975	410.4235	100.9752	56.922300	838.0748	2.000000	46.0458
...	...	...	...	...	...	...	...	...	...
123	American Samoa	2020	0.0000	0.0000	NaN	1410.445506	0.0000	11.481085	9.6998
3931	Marshall Islands	2020	0.0000	0.0000	NaN	1705.819818	0.0000	11.481085	9.9838
3901	Malta	2020	0.0000	0.0000	0.1029	246.407276	0.0000	2.000000	119.1870
3421	Kuwait	2020	0.0134	0.0000	1.2247	246.407276	0.0000	1.000000	1473.6719
6964	Zimbabwe	2020	481.9027	48.4197	108.3022	7.973300	0.0000	73.000000	238.7639

6965 rows × 31 columns

```
# isnull(): It will Return u the no. of the empty cells in the row

df.isnull().sum()
```

```
Area                                0
Year                                0
Savanna fires                      31
Forest fires                       93
Crop Residues                    1389
Rice Cultivation                    0
Drained organic soils (CO2)         0
Pesticides Manufacturing            0
Food Transport                      0
Forestland                        493
Net Forest conversion             493
Food Household Consumption        473
Food Retail                         0
On-farm Electricity Use             0
Food Packaging                      0
Agrifood Systems Waste Disposal     0
Food Processing                     0
Fertilizers Manufacturing           0
IPPU                              743
Manure applied to Soils           928
Manure left on Pasture              0
Manure Management                 928
Fires in organic soils              0
Fires in humid tropical forests   155
On-farm energy use                956
Rural population                    0
Urban population                    0
Total Population - Male             0
Total Population - Female           0
total_emission                      0
```

```
Average Temperature °C               0
dtype: int64
```

```
# drop():It is used to delete the specifi ed columns As the no. of the empty cells in the column anzsic_descriptor2 is 105 and
# category is 179 and this both the columns are not useful to predictthe data so deleting both the columns

df.drop(["Food Transport","Manure left on Pasture","IPPU"],axis=1,inplace=True)


# checking the deleted columns

df.head()
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Forestland	Net For convers
0	Afghanistan	1990	14.7237	0.0557	205.6077	686.00	0.0	11.807483	-2388.803	
1	Afghanistan	1991	14.7237	0.0557	209.4971	678.16	0.0	11.712073	-2388.803	
2	Afghanistan	1992	14.7237	0.0557	196.5341	686.00	0.0	11.712073	-2388.803	
3	Afghanistan	1993	14.7237	0.0557	230.8175	686.00	0.0	11.712073	-2388.803	
4	Afghanistan	1994	14.7237	0.0557	242.0494	705.60	0.0	11.712073	-2388.803	

5 rows × 28 columns

```
#dropna():It is used to delete the rows which have very less null values and then bring the data in the equal manner.

df.dropna(inplace=True)
df.isnull().sum()
```

```
Area                              0
Year                              0
Savanna fires                     0
Forest fires                      0
Crop Residues                     0
Rice Cultivation                  0
Drained organic soils (CO2)       0
Pesticides Manufacturing          0
Forestland                        0
Net Forest conversion             0
Food Household Consumption        0
Food Retail                       0
On-farm Electricity Use           0
Food Packaging                    0
Agrifood Systems Waste Disposal   0
Food Processing                   0
Fertilizers Manufacturing         0
Manure applied to Soils           0
Manure Management                 0
Fires in organic soils            0
Fires in humid tropical forests   0
On-farm energy use                0
Rural population                  0
Urban population                  0
Total Population - Male           0
Total Population - Female         0
total_emission                    0
Average Temperature °C            0
dtype: int64
```

```
df.info()
```

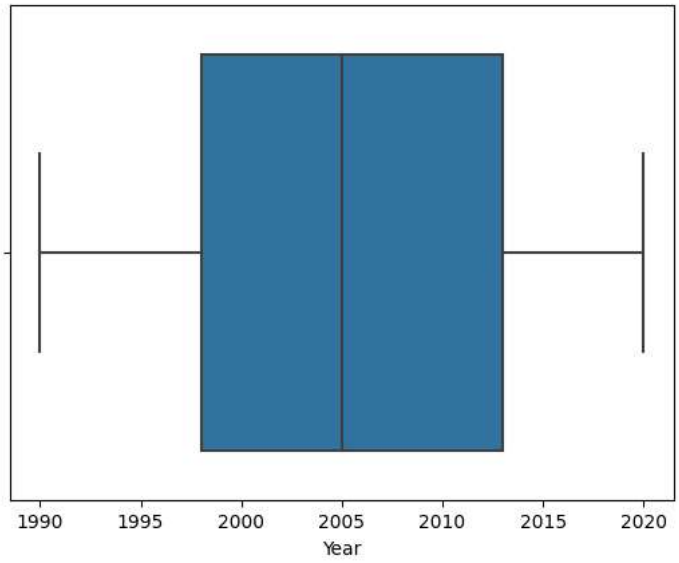
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4639 entries, 31 to 6964
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Area                            4639 non-null   object
 1   Year                            4639 non-null   int64
 2   Savanna fires                   4639 non-null   float64
 3   Forest fires                    4639 non-null   float64
 4   Crop Residues                   4639 non-null   float64
```

```
 5   Rice Cultivation                4639 non-null   float64
 6   Drained organic soils (CO2)     4639 non-null   float64
 7   Pesticides Manufacturing        4639 non-null   float64
 8   Forestland                      4639 non-null   float64
 9   Net Forest conversion           4639 non-null   float64
 10  Food Household Consumption      4639 non-null   float64
 11  Food Retail                     4639 non-null   float64
 12  On-farm Electricity Use         4639 non-null   float64
 13  Food Packaging                  4639 non-null   float64
 14  Agrifood Systems Waste Disposal 4639 non-null   float64
 15  Food Processing                 4639 non-null   float64
 16  Fertilizers Manufacturing       4639 non-null   float64
 17  Manure applied to Soils         4639 non-null   float64
 18  Manure Management               4639 non-null   float64
 19  Fires in organic soils          4639 non-null   float64
 20  Fires in humid tropical forests 4639 non-null   float64
 21  On-farm energy use              4639 non-null   float64
 22  Rural population                4639 non-null   float64
 23  Urban population                4639 non-null   float64
 24  Total Population - Male         4639 non-null   float64
 25  Total Population - Female       4639 non-null   float64
 26  total_emission                  4639 non-null   float64
 27  Average Temperature °C          4639 non-null   float64
dtypes: float64(26), int64(1), object(1)
memory usage: 1.0+ MB
```

```
# DETECTING OUTLIER
# Outlier: It is an extreme value that falls far outside the typical range of values in a dataset.
# It is a Univarient Analysis Because we are analyzing using the single variable

sns.boxplot(x=df['Year'])
```

```
<Axes: xlabel='Year'>
```




```
#There is no outlier in the year column

sns.boxplot(x=df['total_emission'])
```

```
<Axes: xlabel='total_emission'>
```

```
# There are outlier in the data_val column
```

```
# Calculates Interquartile Range (IQR)for each column in 25 % and 75 % and
#then it will substract to get the 50% of Interquartile Range (IQR)
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Year                              1.500000e+01
Savanna fires                     3.131433e+02
Forest fires                      2.082284e+02
Crop Residues                     4.719205e+02
Rice Cultivation                  1.189403e+03
Drained organic soils (CO2)       2.806704e+03
Pesticides Manufacturing          1.770000e+02
Forestland                        7.268415e+03
Net Forest conversion             7.068600e+03
Food Household Consumption        2.137858e+03
Food Retail                       1.297664e+03
On-farm Electricity Use           5.116611e+02
Food Packaging                    4.050632e+02
Agrifood Systems Waste Disposal   4.882189e+03
Food Processing                   1.623755e+03
Fertilizers Manufacturing         2.036627e+03
Manure applied to Soils           6.475016e+02
Manure Management                 1.556706e+03
Fires in organic soils            0.000000e+00
Fires in humid tropical forests   6.031430e+01
On-farm energy use                1.642417e+03
Rural population                  1.057895e+07
Urban population                  1.303215e+07
Total Population - Male           1.271566e+07
Total Population - Female         1.280551e+07
total_emission                    6.519072e+04
Average Temperature °C            7.291667e-01
dtype: float64
<ipython-input-29-c7d1a4625ffc>:3: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future ve
  Q1 = df.quantile(0.25)
<ipython-input-29-c7d1a4625ffc>:4: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future ve
  Q3 = df.quantile(0.75)
```

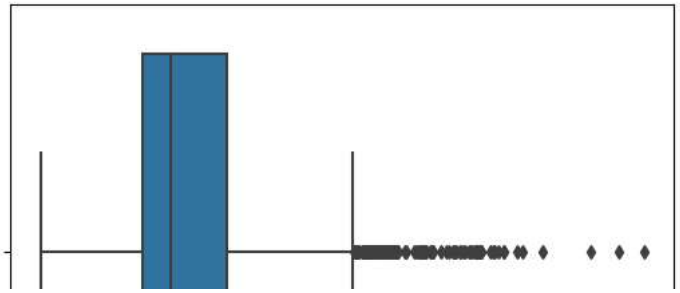
```
#(df < (Q1 - 1.5 * IQR)): This part of the expression checks if any value in the DataFrame islessthan thelower bound
#(Q1 - 1.5 * IQR).(df > (Q3 + 1.5 * IQR)): This part of the expression checks if any value in the DataFrame is greater
#than the upper bound(Q3 + 1.5 * IQR).The|operator is used to combine these two conditions with an OR operation.

df = df[~((df < (Q1 -1.5* IQR)) |(df > (Q3 +1.5* IQR))).any(axis=1)]
df.shape
```

```
<ipython-input-30-b93757ae4a09>:5: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will raise
  df = df[~((df < (Q1 -1.5* IQR)) |(df > (Q3 +1.5* IQR))).any(axis=1)]
(1919, 28)
```

```
sns.boxplot(x=df['total_emission'])
```

```
<Axes: xlabel='total_emission'>
```



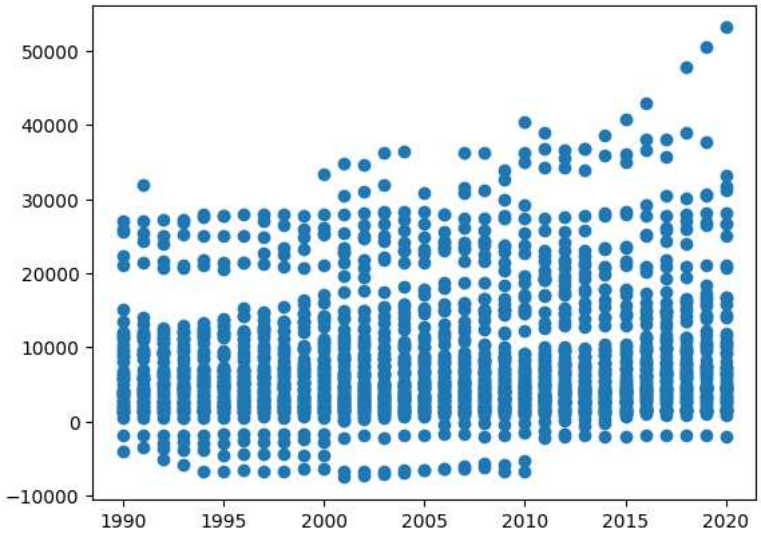
```
#outliners are removed successfully
```



```
# EDA (Exploratory Data Analysis)
# It is used to understand the main characteristics, patterns, and insights hidden in the data.

plt.scatter(df['Year'],df['total_emission'])
```

```
<matplotlib.collections.PathCollection at 0x7d70b3bf5f00>
```



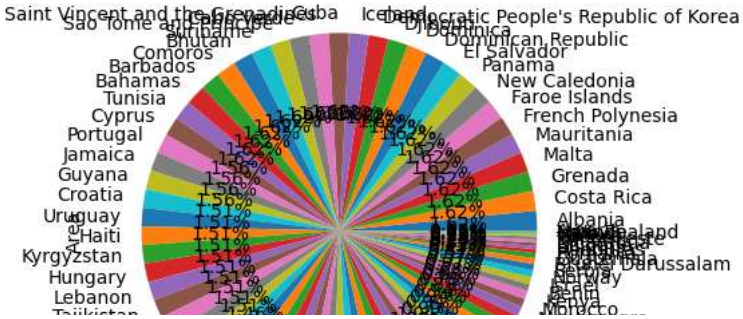

```
#Scatter PlotT
#It is Bivarent analysis because the 2 variables that is x and y are used.
#It represents data points as individual dots on a two-dimensional plane, with one
#variable on the x-axis and the other variable on the y-axis. Eachdot represents an observation or data point.
#In this also Year is the x axis whereas the total_emission is the y axis


pie=df['Area'].value_counts(12)
pie
```

```
Albania        0.016154
Costa Rica     0.016154
Grenada        0.016154
Malta          0.016154
Mauritania     0.016154
                  ...
Greece         0.002084
Malawi         0.001042
New Zealand    0.000521
Uganda         0.000521
Algeria        0.000521
Name: Area, Length: 83, dtype: float64
```

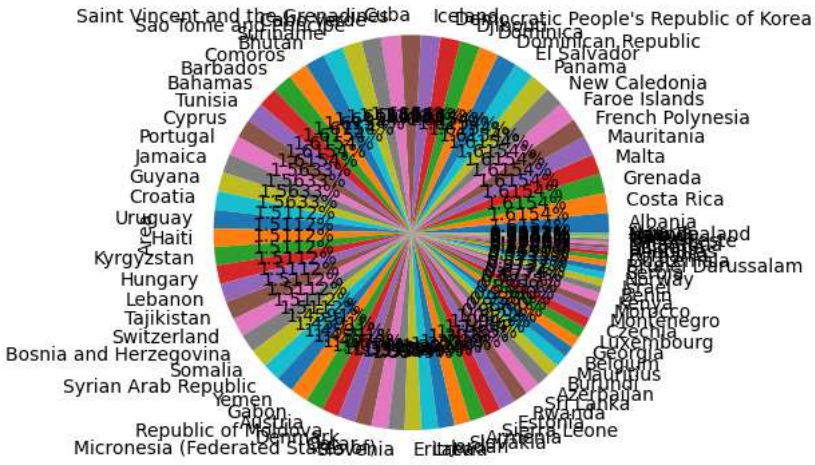
```
pie.plot(kind="pie",autopct="%.2f%%")
```

```
<Axes: ylabel='Area'>
```




```
df["Area"].value_counts().plot(kind="pie",autopct="%.4f%%")
```

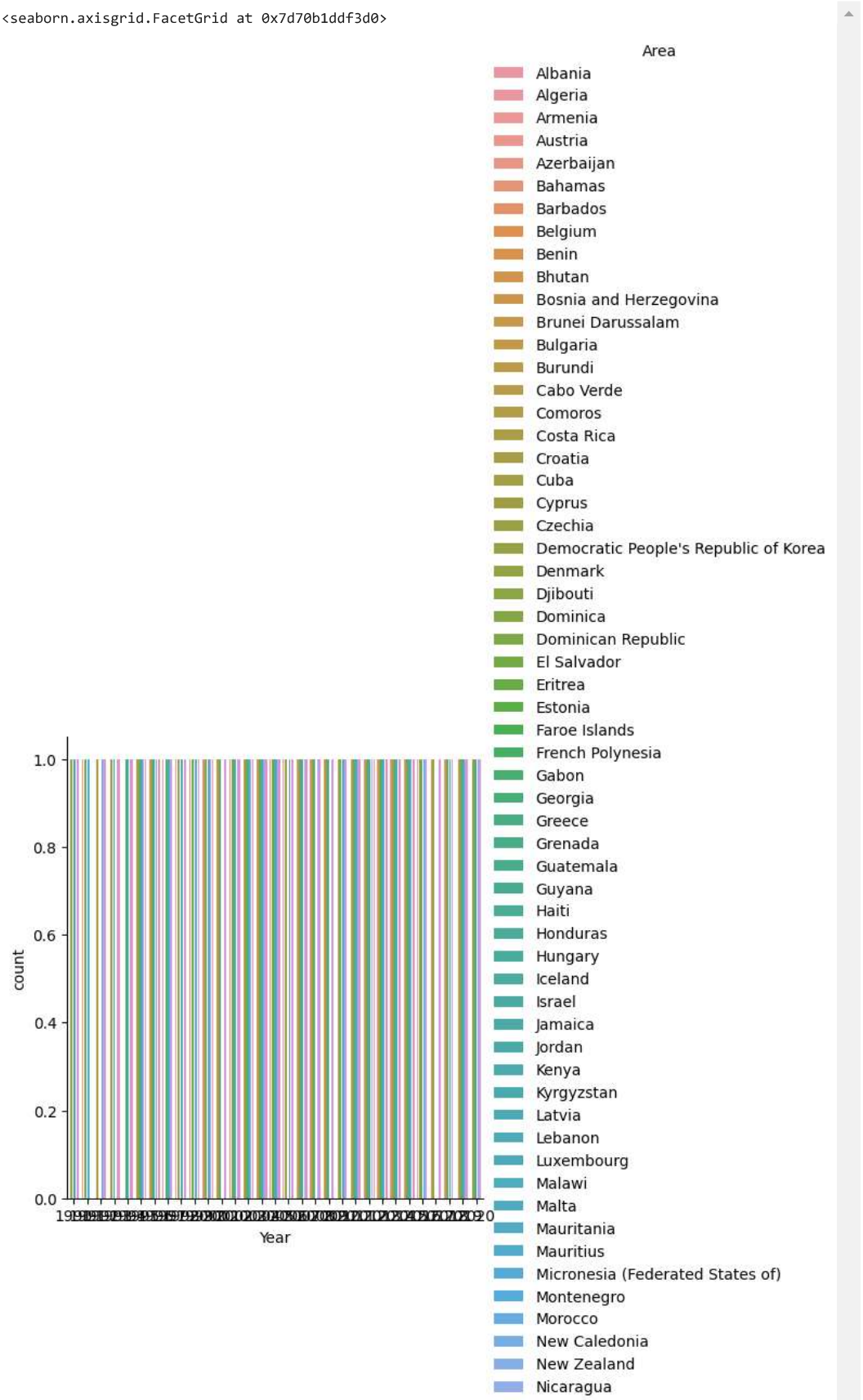
```
<Axes: ylabel='Area'>
```




```
# Pie Chart
# It is univarient analysis , as single variable is used.In this region and gas can show the region in the pie format.
# It is called a"pie chart"because the chart resembles a pie that is divided into slices, with each slice representing a
# particular category or datapoint.The size of each slice corresponds to the
# proportion or percentage of the whole that each category represents.


sns.catplot(x ='Year', hue ='Area',kind ='count', data = df)
```

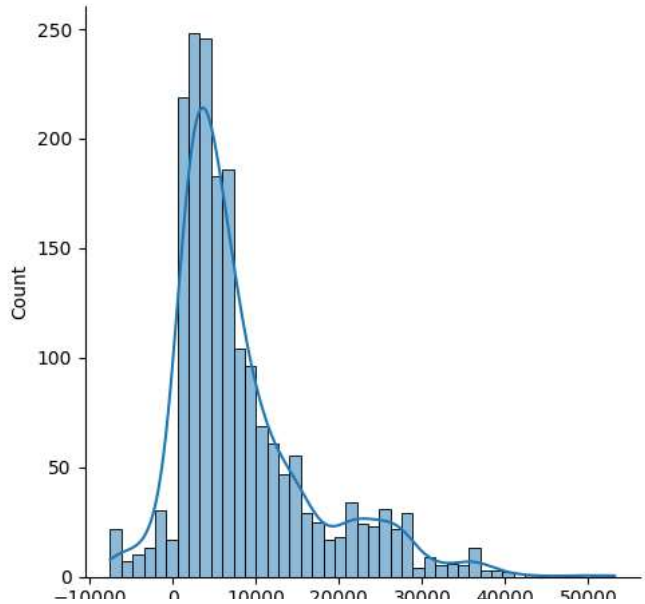
```
<seaborn.axisgrid.FacetGrid at 0x7d70b1ddf3d0>
```




```
#Count Plot
#In the Countplots height of each bar represents the number of occurrences of each category in the dataset. Countplots are particularly usefu
#In this the number of the gases released in each year.
```

```
sns.displot(df['total_emission'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d70b3c7f0a0>
```




```
# displot is used to create a histogram to visualize the distribution of a numerical variable.
#It is used to create a KDE plot to visualize the estimated probability density function of a numerical variable. KDE plots show the
#smoothedcontinuous representation of the data distribution.


#HEATMAPS
#Displays a 2D representation of data using colors to visualize the intensity or correlation between
#two variables in a matrix-like format.
plt.figure(figsize=(10,5))
m= df.corr()
sns.heatmap(m,cmap="crest",annot=True)
m
```

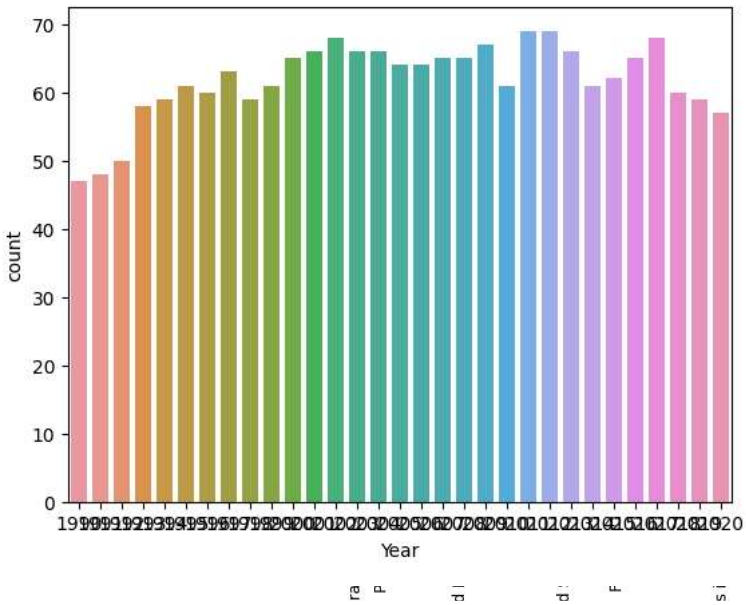
```
<ipython-input-44-98cb1e00c8d4>:5: FutureWarning: The default value of numeric_only in DataFrame.corr i
  m= df.corr()
```

	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Forestlan
Year	1.000000	0.032143	-0.061939	0.017639	0.008672	-0.020397	0.007087	0.09493
Savanna fires	0.032143	1.000000	0.265603	-0.008186	0.019222	-0.045424	-0.004106	0.07189
Forest fires	-0.061939	0.265603	1.000000	0.067989	0.325408	-0.048992	0.116353	-0.02613
Crop Residues	0.017639	-0.008186	0.067989	1.000000	0.349119	0.359830	0.452728	-0.27918
Rice Cultivation	0.008672	0.019222	0.325408	0.349119	1.000000	0.017748	0.144519	-0.17983
Drained organic soils (CO2)	-0.020397	-0.045424	-0.048992	0.359830	0.017748	1.000000	0.095052	-0.33146
Pesticides Manufacturing	0.007087	-0.004106	0.116353	0.452728	0.144519	0.095052	1.000000	-0.21549
Forestland	0.094936	0.071894	-0.026131	-0.279186	-0.179830	-0.331466	-0.215497	1.00000
Net Forest conversion	0.049618	0.313960	0.232217	0.016251	0.068036	0.010797	0.042096	0.10975
Food Household Consumption	0.035252	-0.090331	0.018705	0.602614	0.181549	0.069267	0.537004	-0.24801
Food Retail	0.211820	-0.088524	-0.035357	0.449596	0.223950	0.082708	0.384251	-0.14345
On-farm Electricity Use	-0.044669	-0.108735	-0.037286	0.415748	0.177741	0.104193	0.386870	-0.07708
Food Packaging	0.008418	-0.096523	-0.049688	0.459683	0.074929	0.055949	0.381650	-0.31255
Agrifood Systems Waste Disposal	0.031638	0.019299	0.314027	0.486832	0.476458	0.014967	0.423701	-0.13607
Food Processing	0.027951	-0.105398	-0.041656	0.590280	0.169261	0.258207	0.453871	-0.23331
Fertilizers Manufacturing	0.010296	-0.077340	-0.012881	-0.208898	-0.167865	0.015013	-0.218767	0.18342
Manure applied to Soils	0.004710	-0.064012	0.057011	0.675743	0.223431	0.261444	0.552804	-0.36535
Manure Management	-0.010712	-0.050883	0.023214	0.620635	0.213268	0.247360	0.447082	-0.28925
Fires in organic soils	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
Fires in humid tropical forests	-0.006295	0.427872	0.573403	-0.088410	0.063121	-0.023622	0.111501	-0.04602
On-farm energy use	-0.068973	-0.086395	0.156704	0.489323	0.369714	0.220655	0.339138	-0.31445
Rural population	-0.025703	0.173706	0.199757	0.366582	0.314537	0.027429	0.134963	0.00383
Urban population	0.080065	0.026982	0.289467	0.609766	0.502512	0.020289	0.511756	-0.22503
Total Population - Male	0.024454	0.121496	0.269073	0.526987	0.453329	0.020070	0.337846	-0.10494
Total Population - Female	0.023568	0.119015	0.276631	0.544720	0.459358	0.034917	0.346953	-0.11403
total_emission	0.126660	0.067399	0.192754	0.544090	0.313986	0.162609	0.425683	0.08312
Average Temperature	0.536415	0.002594	-0.033905	0.160504	0.074769	0.121917	0.064685	-0.09022

°C

27 rows × 27 columns

```
#COUNT PLOT
# Displays the count of occurrences of each category in a categorical variable using bars.
sns.countplot(x='Year',data=df)
plt.show()
```




```
#In x axis we r including all the columns except the targeted row which will be the y axis
x=df.iloc[:,0:5]
x
```

	Area	Year	Savanna fires	Forest fires	Crop Residues
31	Albania	1990	5.5561	7.0253	59.2391
32	Albania	1991	5.5561	7.0253	31.4625
33	Albania	1992	5.5561	7.0253	29.9373
34	Albania	1993	5.5561	7.0253	44.0550
35	Albania	1994	5.5561	7.0253	42.4253
...	...	...	...	...	...
6894	Yemen	2014	0.0734	0.0000	62.1553
6895	Yemen	2015	0.0000	0.0000	46.2019
6896	Yemen	2016	0.0000	0.0000	38.8552
6897	Yemen	2017	0.0000	0.0000	38.5935
6898	Yemen	2018	0.0000	0.0000	37.4579

1919 rows × 5 columns

```
#In y axis we r including the targeted row only i.e total_emission row
#Which will going to calculate and give us the value of the data i.e how much CO2 gass emmision would be produced by each region inthe year
y=df.iloc[:,5]
y
```

```
31       23.520000
32        6.272000
33        1.881600
34        1.097600
35        0.000000
           ...
6894    796.196122
6895    796.196122
6896    796.196122
6897    796.196122
6898    796.196122
Name: Rice Cultivation, Length: 1919, dtype: float64
```

```
#ENCODINGIn Encoding we will convert all the
#object: datatype of the column to the integer for the Machine Learning Process
#OriginalEncoder: It is usedfor encoding categorical features into ordinal integers
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
```

```
x[["Area","Year","Savanna fires"]]=oe.fit_transform(x[["Area","Year","Savanna fires"]])
x
```

	Area	Year	Savanna fires	Forest fires	Crop Residues
31	0.0	0.0	620.0	7.0253	59.2391
32	0.0	1.0	620.0	7.0253	31.4625
33	0.0	2.0	620.0	7.0253	29.9373
34	0.0	3.0	620.0	7.0253	44.0550
35	0.0	4.0	620.0	7.0253	42.4253
...	...	...	...	...	...
6894	82.0	24.0	101.0	0.0000	62.1553
6895	82.0	25.0	0.0	0.0000	46.2019
6896	82.0	26.0	0.0	0.0000	38.8552
6897	82.0	27.0	0.0	0.0000	38.5935
6898	82.0	28.0	0.0	0.0000	37.4579

1919 rows × 5 columns

```
df.head()
```

	Area	Year	Savanna fires	Forest fires	Crop Residues	Rice Cultivation	Drained organic soils (CO2)	Pesticides Manufacturing	Forestland	Net Fores conversic
31	Albania	1990	5.5561	7.0253	59.2391	23.5200	110.5705	2.0	72.8581	0
32	Albania	1991	5.5561	7.0253	31.4625	6.2720	110.5705	2.0	72.8581	0
33	Albania	1992	5.5561	7.0253	29.9373	1.8816	110.5705	2.0	72.8581	0
34	Albania	1993	5.5561	7.0253	44.0550	1.0976	110.5705	2.0	72.8581	0
35	Albania	1994	5.5561	7.0253	42.4253	0.0000	110.5705	3.0	72.8581	0

5 rows × 28 columns

```
# Model Building
#Model building is the process of creating and training a predictive or statistical model using data
#to make accurate predictions or provideinsights.
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets (80% training, 20% testing)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
#Linear Regression (LR) is a supervised machine learning algorithm used to model the
#relationship between a dependent variable and one or mmore independent variables by fitting a linear equation.
```

```
from sklearn.linear_model import LinearRegression
#scikit_learn
x = df[['Year']]
y = df['total_emission']
```

```
# Check for and handle non-numeric values in the 'total_emission' column
df['total_emission']= pd.to_numeric(df['total_emission'],errors='coerce')
# Convert non-numeric values to NaN
```

```
# Drop rows with NaN values in the 'total_emission' column
df = df.dropna(subset=['total_emission'])
```

```
<ipython-input-68-714f97f6da18>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  df['total_emission']= pd.to_numeric(df['total_emission'],errors='coerce')
```

```
# Create a LinearRegression model
model = LinearRegression()


# Train the model using the training data
model.fit(x_train,y_train)
# Make predictions on the test data
y_pred = model.predict(x_test)


from sklearn.metrics import mean_squared_error, r2_score
# Evaluate the model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
#Mean Squared Error (MSE) measures the average squared difference between predicted and actual values.
#R-squared (R²) indicates the proportion of variance explained by the model.
print("R-squared:", r2)
```

```
Mean Squared Error: 161517.173148036
R-squared: 0.13359425303774664
```

```
SUMMARY
From this Dataset of CO2-gas-emissions-by-Area-industry-and-household
We have found the CO2 emmisions releases from each Area
By Providing the Year the value of CO2 gas emissions by Area industry can be predicted
```

✓ 0s completed at 5:45 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.