dataexplorationn

March 13, 2024

[6]:
```
import pandas as pd

# Read the dataset
df = pd.read_csv('dataset.csv', header=None)
```

[7]:
```
df.head()
```

[7]:

	0	1	2	3	4	5	6	7	8	9	…	26	27	28	29	30	31	32	33	34
0	ObjectId	Country	ISO2	ISO3	Indicator	Unit	Source	CTS Code	CTS Name	CTS Full Descriptor	…	2009.000	2010.000	2011.000	2012.000	2013.000	2014.000	2015.000	2016.000	2017.000
1	1	Argentina	AR	ARG	CO2 emissions	Millions of Metric tons of CO2	OECD (2021), OECD Inter-Country Input-Output D…	ECNC	CO2 Emissions	Environment, Climate Change, Economic Activity…	…	0.284	0.319							
2	2	Argentina	AR	ARG	CO2 emissions	Millions of Metric tons of CO2	OECD (2021), OECD Inter-Country Input-Output D…	ECNC	CO2 Emissions	Environment, Climate Change, Economic Activity…	…	0.000	0.000							
3	3	Argentina	AR	ARG	CO2 emissions	Millions of Metric tons of CO2	OECD (2021), OECD Inter-Country Input-Output D…	ECNC	CO2 Emissions	Environment, Climate Change, Economic Activity…	…	0.725	0.738							
4	4	Argentina	AR	ARG	CO2 emissions	Millions of Metric tons of CO2	OECD (2021), OECD Inter-Country Input-Output D…	ECNC	CO2 Emissions	Environment, Climate Change, Economic Activity…	…	13.804	14.110							

```
1     0.338     0.377     0.378     0.369     0.371     0.303     0.358
2     0.000     0.000     0.000     0.000     0.000     0.000     0.000
3     0.765     0.777     0.840     0.794     0.852     0.789     0.846
4    14.616    14.120    15.358    13.728    14.273    13.838    14.121

          35
0   2018.000
1      0.344
2      0.000
3      0.857
4     13.067

[5 rows x 36 columns]
```

[8]:
```
df.tail()
```

[8]:
```
        0        1   2    3                          4   \
8906  8906  Vietnam  VN  VNM  CO2 emissions multipliers
8907  8907  Vietnam  VN  VNM  CO2 emissions multipliers
8908  8908  Vietnam  VN  VNM  CO2 emissions multipliers
8909  8909  Vietnam  VN  VNM  CO2 emissions multipliers
8910  8910  Vietnam  VN  VNM  CO2 emissions multipliers

                                                    5   \
8906  Metric Tons of CO2 Emissions per $1million USD…
8907  Metric Tons of CO2 Emissions per $1million USD…
8908  Metric Tons of CO2 Emissions per $1million USD…
8909  Metric Tons of CO2 Emissions per $1million USD…
8910  Metric Tons of CO2 Emissions per $1million USD…

                                                    6     7   \
8906  OECD (2021), OECD Inter-Country Input-Output D…  ECNM
8907  OECD (2021), OECD Inter-Country Input-Output D…  ECNM
8908  OECD (2021), OECD Inter-Country Input-Output D…  ECNM
8909  OECD (2021), OECD Inter-Country Input-Output D…  ECNM
8910  OECD (2021), OECD Inter-Country Input-Output D…  ECNM

                            8   \
8906  CO2 Emissions Multipliers
8907  CO2 Emissions Multipliers
8908  CO2 Emissions Multipliers
8909  CO2 Emissions Multipliers
8910  CO2 Emissions Multipliers

                                                    9   …           26  \
8906  Environment, Climate Change, Economic Activity…  …   571.409871
8907  Environment, Climate Change, Economic Activity…  …   906.449860
```

```
8908  Environment, Climate Change, Economic Activity…  …  3807.684853
8909  Environment, Climate Change, Economic Activity…  …   449.066931
8910  Environment, Climate Change, Economic Activity…  …   564.308833

              27           28           29           30           31  \
8906   460.836156   347.350194   299.661077   282.529754   297.871935
8907   889.629032   725.459644   659.111915   573.819271   585.155874
8908  3020.134010  2253.246177  1917.872096  1908.984147  2093.765490
8909   440.446520   378.152049   357.297470   350.369327   377.126416
8910   558.348722   446.045111   389.419367   335.815256   340.664071

              32           33           34           35
8906   365.437022   333.755635   303.371701   326.056753
8907   693.572818   702.200536   613.915080   694.331871
8908  2982.160528  3974.738515  3664.875652  3295.556416
8909   479.038527   506.595843   451.921218   508.214053
8910   570.908659   460.042792   398.818037   441.426688

[5 rows x 36 columns]
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8911 entries, 0 to 8910
Data columns (total 36 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       8911 non-null   object
 1   1       8911 non-null   object
 2   2       8911 non-null   object
 3   3       8911 non-null   object
 4   4       8911 non-null   object
 5   5       8911 non-null   object
 6   6       8911 non-null   object
 7   7       8911 non-null   object
 8   8       8911 non-null   object
 9   9       8911 non-null   object
 10  10      8911 non-null   object
 11  11      8911 non-null   object
 12  12      8911 non-null   float64
 13  13      8911 non-null   float64
 14  14      8911 non-null   float64
 15  15      8911 non-null   float64
 16  16      8911 non-null   float64
 17  17      8911 non-null   float64
 18  18      8911 non-null   float64
 19  19      8911 non-null   float64
```

```
 20  20      8911 non-null   float64
 21  21      8911 non-null   float64
 22  22      8911 non-null   float64
 23  23      8911 non-null   float64
 24  24      8911 non-null   float64
 25  25      8911 non-null   float64
 26  26      8911 non-null   float64
 27  27      8911 non-null   float64
 28  28      8911 non-null   float64
 29  29      8911 non-null   float64
 30  30      8911 non-null   float64
 31  31      8911 non-null   float64
 32  32      8911 non-null   float64
 33  33      8911 non-null   float64
 34  34      8911 non-null   float64
 35  35      8911 non-null   float64
dtypes: float64(24), object(12)
memory usage: 2.4+ MB
```

[10]:
```
df.describe()
```

[10]:

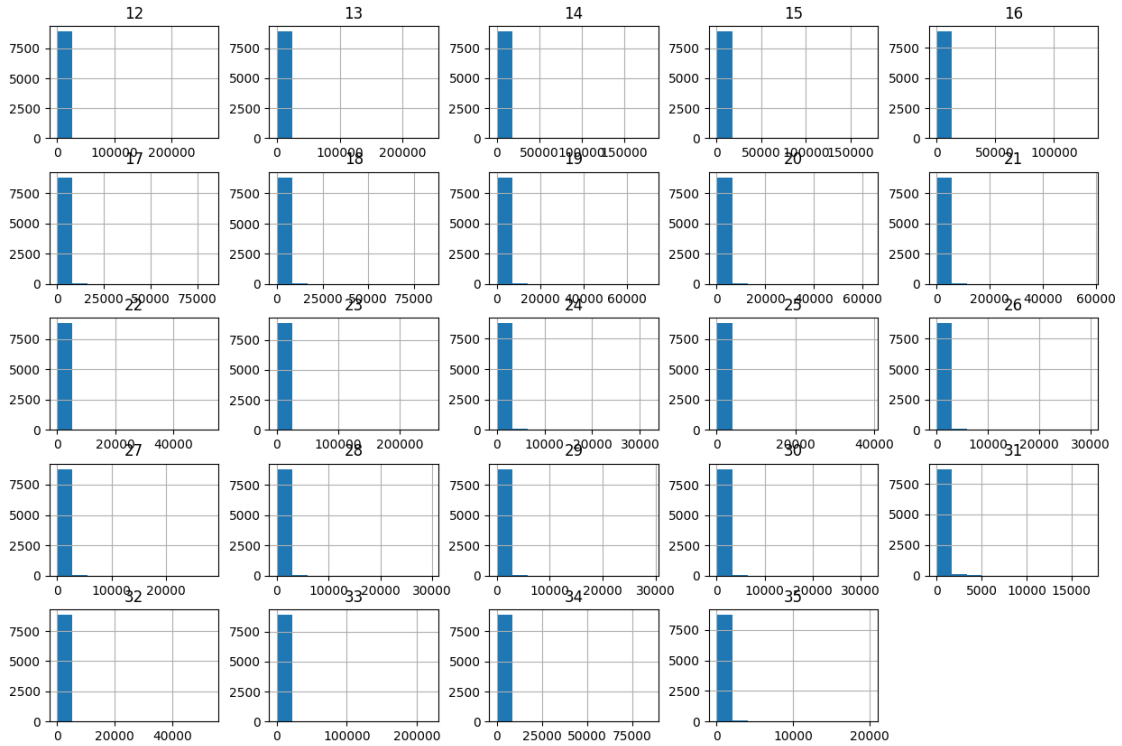
	12	13	14	15	\
count	8911.000000	8911.000000	8911.000000	8911.000000	
mean	642.842524	626.401664	598.046073	634.676427	
std	4630.516299	4370.165976	3440.427988	3472.603771	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.552500	0.580000	0.574500	0.583500	
50%	51.546392	51.699397	52.323204	54.192531	
75%	342.607969	340.100350	349.717484	359.159509	
max	268619.614800	244816.005900	180811.599800	172174.706800	

	16	17	18	19	20	\
count	8911.000000	8911.000000	8911.000000	8911.000000	8911.000000	
mean	628.686490	584.973942	581.599920	546.066537	490.382253	
std	3072.350175	2403.399359	2405.242879	2168.166807	2003.733249	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.606500	0.672000	0.677000	0.703500	0.747000	
50%	54.685426	57.688482	59.341376	56.864809	50.775632	
75%	357.752062	362.333765	363.807009	341.158113	302.449951	
max	131047.244200	82288.011300	84204.541120	70922.998900	63207.730870	

	21	…	26	27	28	\
count	8911.000000	…	8911.000000	8911.000000	8911.000000	
mean	422.529533	…	277.746523	255.795930	228.619174	
std	1735.852556	…	1073.245490	968.081649	885.445898	
min	0.000000	…	0.000000	0.000000	0.000000	
25%	0.767000	…	0.831500	0.865500	0.870500	

```
50%       44.730721  …     31.517643     29.970462     26.790116
75%      259.425137  …    181.731933    170.212573    149.830403
max    57965.874820  …  29960.060010  28251.186870  29678.748310

                29           30           31           32           33  \
count   8911.000000  8911.000000  8911.000000  8911.000000  8911.000000
mean     226.871328   220.108604   217.332860   255.144850   300.788955
std      913.272158   911.338240   800.815609  1154.602752  3413.627056
min        0.000000     0.000000     0.000000     0.000000     0.000000
25%        0.895000     0.907500     0.889500     0.905000     0.898500
50%       27.460969    26.488330    25.461823    27.391488    27.463739
75%      150.806335   146.300718   148.550918   167.713508   167.144784
max    29089.169930  32033.948990  17089.568230  53393.751770  220076.705300

                34           35
count   8911.000000  8911.000000
mean     256.377863   224.673274
std     1516.851816   794.117554
min        0.000000     0.000000
25%        0.962500     0.976000
50%       26.602324    25.706207
75%      158.129264   150.401056
max    85065.783690  20139.395610

[8 rows x 24 columns]
```

[11]:
```
import matplotlib.pyplot as plt

df.hist(bins=10, figsize=(15, 10))
plt.show()
```



[12]:
```
df.isnull().sum()
```

[12]:
```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
```

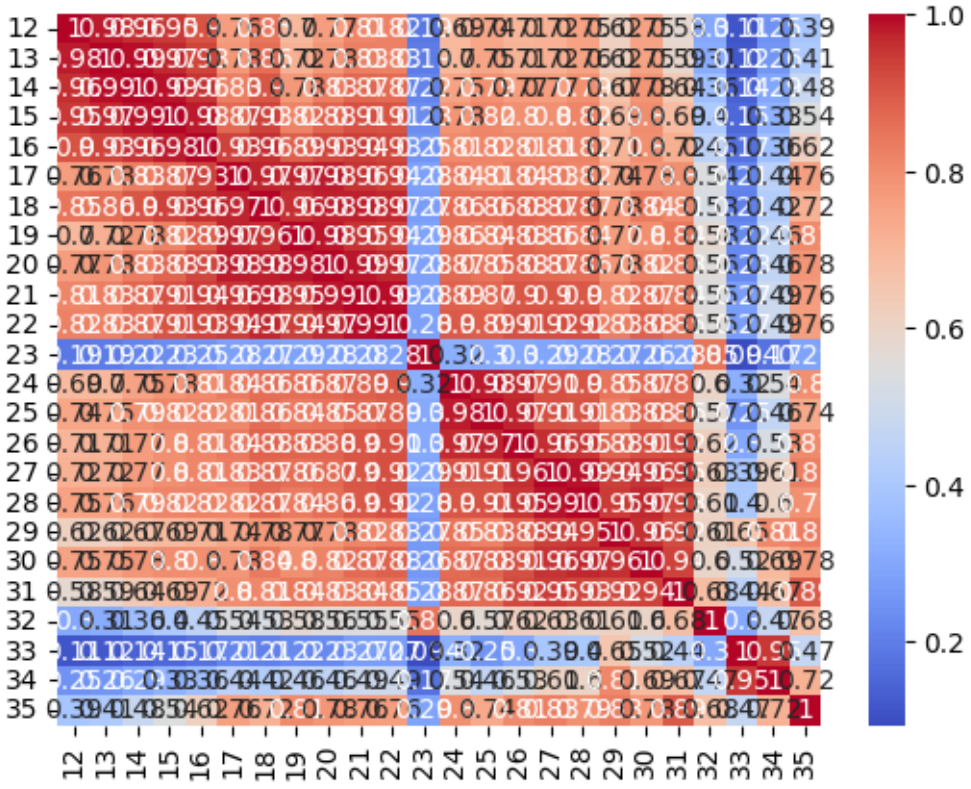
```
21    0
22    0
23    0
24    0
25    0
26    0
27    0
28    0
29    0
30    0
31    0
32    0
33    0
34    0
35    0
dtype: int64
```

[13]:
```
# Drop rows with missing values
df.dropna(inplace=True)
```

[15]:
```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

```
<ipython-input-15-038fc9e360ab>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

[15]: <Axes: >


12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
1.0
0.8
0.6
0.4
0.2