In [98]:
```
# import packages

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

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

In [4]:
```
# upload air quality concentration dataset
conc = pd.read_csv('annual_conc_by_monitor_2019.zip')
conc.head()
```

Out[4]:

	State Code	County Code	Site Num	Parameter Code	POC	Latitude	Longitude	Datum	Parameter Name	Samp Durati
0	1	3	10	44201	1	30.497478	-87.880258	NAD83	Ozone	1 HOU
1	1	3	10	44201	1	30.497478	-87.880258	NAD83	Ozone	8-H RU AV BEG HOU
2	1	3	10	44201	1	30.497478	-87.880258	NAD83	Ozone	8-H RU AV BEG HOU
3	1	3	10	44201	1	30.497478	-87.880258	NAD83	Ozone	8-H RU AV BEG HOU
4	1	3	10	88101	1	30.497478	-87.880258	NAD83	PM2.5 - Local Conditions	HOU

5 rows × 55 columns

In [99]:
```
# upload chronic disease indicator data
cdi = pd.read_csv('U.S._Chronic_Disease_Indicators__CDI_.csv')
cdi.head()
```

```
C:\Users\Amulya Cherian\AppData\Local\Temp\ipykernel_8348\2270999370.py:
2: DtypeWarning: Columns (10) have mixed types. Specify dtype option on
import or set low_memory=False.
  cdi = pd.read_csv('U.S._Chronic_Disease_Indicators__CDI_.csv')
```

Out[99]:

	YearStart	YearEnd	LocationAbbr	LocationDesc	DataSource	Topic	Question	R
0	2014	2014	AR	Arkansas	SEDD; SID	Asthma	Hospitalizations for asthma	
1	2018	2018	CO	Colorado	SEDD; SID	Asthma	Hospitalizations for asthma	
2	2018	2018	DC	District of Columbia	SEDD; SID	Asthma	Hospitalizations for asthma	
3	2017	2017	GA	Georgia	SEDD; SID	Asthma	Hospitalizations for asthma	
4	2010	2010	MI	Michigan	SEDD; SID	Asthma	Hospitalizations for asthma	

5 rows × 34 columns

In [4]:
```
# check for unique values
conc['Parameter Name'].unique()
conc['Sample Duration'].unique()
conc['Units of Measure'].unique()
```

```
       'Silver PM2.5 LC', 'Zinc PM2.5 LC', 'Strontium PM2.5 LC',
       'Sulfur PM2.5 LC', 'Rubidium PM2.5 LC', 'Potassium PM2.5 LC',
       'Sodium PM2.5 LC', 'Zirconium PM2.5 LC', 'Chloride PM2.5 LC',
       'Ammonium Ion PM2.5 LC', 'Sodium Ion Pm2.5 LC',
       'Potassium Ion PM2.5 LC', 'Total Nitrate PM2.5 LC',
       'OC PM2.5 LC TOR', 'EC PM2.5 LC TOR', 'OC1 PM2.5 LC',
       'OC2 PM2.5 LC', 'OC3 PM2.5 LC', 'OC4 PM2.5 LC', 'OP PM2.5 LC T
OR',
       'EC1 PM2.5 LC', 'EC2 PM2.5 LC', 'EC3 PM2.5 LC', 'Nitrite PM2.5
LC',
       'Ammonium Sulfate PM2.5 LC', 'Ammonium Nitrate PM2.5 LC',
       'Soil PM2.5 LC', 'Organic Carbon Mass PM2.5 LC',
       'OC CSN_Rev Unadjusted PM2.5 LC TOT',
       'EC CSN_Rev Unadjusted PM2.5 LC TOT',
       'OC CSN_Rev Unadjusted PM2.5 LC TOR',
       'OC1 CSN_Rev Unadjusted PM2.5 LC',
       'OC2 CSN_Rev Unadjusted PM2.5 LC',
       'OC3 CSN_Rev Unadjusted PM2.5 LC',
       'OC4 CSN_Rev Unadjusted PM2.5 LC',
```

In [91]:
```
# ensure CDI dataset only includes values from 2019
cdi = cdi[cdi['YearEnd']==2019]

# check unique values
cdi['Topic'].unique()
```

Out[91]:
```
array(['Asthma', 'Cardiovascular Disease', 'Chronic Kidney Disease',
       'Chronic Obstructive Pulmonary Disease', 'Diabetes'], dtype=objec
t)
```

In [6]:

```
# frequency table for CONC dataset: site # and sample duration
sam_site_table = pd.crosstab(index=conc['Site Num'], columns=conc['Sample
sam_site_table

# determine which sample durations to examine
np.sum(sam_site_table == 0, axis=0)

# filter CONC dataset for certain sample durations
x = '1 HOUR'
y = '24-HR BLK AVG'
conc = conc[(conc['Sample Duration']==x) | (conc['Sample Duration']==y)]
conc

# frequency table for CONC dataset: site # and parameter
par_site_table = pd.crosstab(index=conc['Site Num'], columns=conc['Paramet
par_site_table

# determine which parameters to examine
np.sum(par_site_table > 0, axis = 0)

# filter CONC dataset for certain air pollutants
parameters = ['Ozone', 'PM2.5 - Local Conditions', 'PM10 - LC', 'Black Car
conc = conc[conc['Parameter Name'].isin(parameters)]

# view filtered CONC dataset
conc
```

Out[6]:

Sample Duration	1 HOUR	15 MINUTE	2 HOUR	24 HOUR	24-HR BLK AVG	3 HOURS	3-HR BLK AVG	5 MINUTE	8 HOUR	8-HR RUN AVG BEGIN HOUR
Site Num										
1	434	0	0	1519	213	0	16	10	0	198
2	547	0	0	2145	330	74	22	12	6	231
3	522	0	0	1666	231	0	19	13	0	201
4	458	0	0	1143	263	86	20	11	0	183
5	434	0	0	1525	226	0	23	17	0	147
...	...	...	...	...	...	...	...	...	...	...
9702	7	0	0	0	0	0	0	0	0	9

In [92]:
```
# filter CDI dataset for certain health topics
topics = ['Asthma', 'Diabetes', 'Cardiovascular Disease', 'Chronic Obstruc
cdi = cdi[cdi['Topic'].isin(topics)]
cdi
```

Out[92]:

	YearStart	YearEnd	LocationAbbr	LocationDesc	DataSource	Topic	
23	2019	2019	MS	Mississippi	NVSS	Asthma	n
139	2019	2019	UT	Utah	CMS Part A Claims Data	Cardiovascular Disease	Ho for
30969	2019	2019	MS	Mississippi	NVSS	Asthma	n
30988	2019	2019	CO	Colorado	NVSS	Asthma	n
30994	2019	2019	MA	Massachusetts	NVSS	Asthma	n
...	...	...	...	...	...	...	
1182292	2019	2019	WY	Wyoming	BRFSS	Cardiovascular Disease	ar ag
1182333	2019	2019	WY	Wyoming	BRFSS	Diabetes	Pn noni
1182395	2019	2019	WY	Wyoming	BRFSS	Asthma	Pn noni
1182414	2019	2019	WY	Wyoming	BRFSS	Chronic Obstructive Pulmonary Disease	Pr pul
1182456	2019	2019	WY	Wyoming	BRFSS	Chronic Obstructive Pulmonary Disease	Pr curr ar

45909 rows × 34 columns

In [8]:
```
# filter CONC for one-hour values only
one = conc[(conc['Sample Duration']=='1 HOUR')]

# view units
unit = one.groupby(['Parameter Name', 'Units of Measure'])['Units of Measu
unit

# convert units into microgram/m^3 for each pollutant:
# carbon monoxide
CM = (one['Parameter Name'] == 'Carbon monoxide')
one.loc[CM, 'Arithmetic Mean'] = one.loc[CM, 'Arithmetic Mean']*(2.31/2)

# nitric oxide
NO = (one['Parameter Name'] == 'Nitric oxide (NO)')
one.loc[NO, 'Arithmetic Mean'] = one.loc[NO, 'Arithmetic Mean']*(122.42/10

# nitrogen dioxide
NO2 = (one['Parameter Name'] == 'Nitrogen dioxide (NO2)')
one.loc[NO2, 'Arithmetic Mean'] = one.loc[NO2, 'Arithmetic Mean']*(94.35/5

# ozone
OZ = (one['Parameter Name'] == 'Ozone')
one.loc[OZ, 'Arithmetic Mean'] = one.loc[OZ, 'Arithmetic Mean']*(0.194/0.1

# sulfur dioxide
SU = (one['Parameter Name'] == 'Sulfur dioxide')
one.loc[SU, 'Arithmetic Mean'] = one.loc[SU, 'Arithmetic Mean']*(261.36/10

# average parameter mean values by parameter name
para_mean = one.groupby('Parameter Name')['Arithmetic Mean'].mean()

# visualize relationship between parameter name & mean values
paramean_plot = plt.bar(para_mean.index, para_mean.values,color='lightcora
plt.xticks(rotation=45, ha='right')
plt.xlabel('Pollutant'), plt.ylabel('Mean Concentration (µg/m³)'), plt.tit

# create a boxplot
conc_dist = sns.boxplot(data=one, x='Parameter Name', y='Arithmetic Mean',
plt.xticks(rotation=45, ha='right')
plt.xlabel('Pollutant'), plt.ylabel('Concentration (µg/m³)'), plt.title('U
```

Out[8]:
```
Parameter Name                Units of Measure
Black Carbon PM2.5 at 880 nm  Micrograms/cubic meter (LC)      38
Carbon monoxide               Parts per million               281
Nitric oxide (NO)             Parts per billion               510
Nitrogen dioxide (NO2)        Parts per billion               946
Ozone                         Parts per million              1295
PM10 - LC                     Micrograms/cubic meter (LC)     175
PM2.5 - Local Conditions      Micrograms/cubic meter (LC)     794
Sulfur dioxide                Parts per billion               974
Name: Units of Measure, dtype: int64
```

Out[8]:
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
```

In [88]:
```
# examine topic questions in CDI dataset
cdi['Topic'].unique()
to_qu = cdi.groupby(['Topic', 'Question'])['Question'].count()
to_qu
```

Out[88]:
```
array(['Asthma', 'Cardiovascular Disease', 'Chronic Kidney Disease',
       'Chronic Obstructive Pulmonary Disease', 'Diabetes'], dtype=objec
t)
```

Out[88]:

Topic	Question
Asthma	Asthma mortality rate
1227	
	Asthma prevalence among women age
d 18-44 years	
325	
	Current asthma prevalence among a
dults aged >= 18 years	
870	
	Influenza vaccination among nonin
stitutionalized adults aged 18-64 years with asthma	
870	
	Influenza vaccination among nonin
stitutionalized adults aged >= 65 years with asthma	
870	
	Pneumococcal vaccination among no
ninstitutionalized adults aged 18-64 years with asthma	
870	
	Pneumococcal vaccination among no
ninstitutionalized adults aged >= 65 years with asthma	
870	
Cardiovascular Disease	Awareness of high blood pressure
among adults aged >= 18 years	
870	
	Awareness of high blood pressure
among women aged 18-44 years	
325	
	Cholesterol screening among adult
s aged >= 18 years	
870	
	High cholesterol prevalence among
adults aged >= 18 years	
870	
	Hospitalization for heart failure
among Medicare-eligible persons aged >= 65 years	
1248	
	Influenza vaccination among nonin
stitutionalized adults aged 18-64 years with a history of coronary heart	
disease or stroke	870
	Influenza vaccination among nonin
stitutionalized adults aged >= 65 years with a history of coronary heart	
disease or stroke	870
	Mortality from cerebrovascular di
sease (stroke)	
1227	
	Mortality from coronary heart dis
ease	
1227	
	Mortality from diseases of the he
art	
1227	
	Mortality from heart failure
1086	
	Mortality from total cardiovascul
ar diseases	
1227	
	Pneumococcal vaccination among no

```
ninstitutionalized adults aged 18-64 years with a history of coronary he
art disease                      870
                                       Pneumococcal vaccination among no
ninstitutionalized adults aged >= 65 years with a history of coronary he
art disease                      870
                                       Taking medicine for high blood pr
essure control among adults aged >= 18 years with high blood pressure
870
Chronic Kidney Disease                 Mortality with end-stage renal di
sease
1227
                                       Prevalence of chronic kidney dise
ase among adults aged >= 18 years
870
Chronic Obstructive Pulmonary Disease  Hospitalization for chronic obstr
uctive pulmonary disease as any diagnosis among Medicare-eligible person
s aged >= 65 years             1248
                                       Hospitalization for chronic obstr
uctive pulmonary disease as first-listed diagnosis among Medicare-eligib
le persons aged >= 65 years    1248
                                       Influenza vaccination among nonin
stitutionalized adults aged >= 45 years with chronic obstructive pulmona
ry disease                       870
                                       Mortality with chronic obstructiv
e pulmonary disease as underlying cause among adults aged >= 45 years
1227
                                       Mortality with chronic obstructiv
e pulmonary disease as underlying or contributing cause among adults age
d >= 45 years                  1227
                                       Pneumococcal vaccination among no
ninstitutionalized adults aged >= 45 years with chronic obstructive pulm
onary disease                    870
                                       Prevalence of chronic obstructive
pulmonary disease among adults >= 18
870
                                       Prevalence of chronic obstructive
pulmonary disease among adults >= 45 years
870
                                       Prevalence of current smoking amo
ng adults >= 18 with diagnosed chronic obstructive pulmonary disease
870
                                       Prevalence of current smoking amo
ng adults >= 45 years with diagnosed chronic obstructive pulmonary disea
se                               870
Diabetes                               Adults with diagnosed diabetes ag
ed >= 18 years who have taken a diabetes self-management course
870
                                       Diabetes prevalence among women a
ged 18-44 years
325
                                       Dilated eye examination among adu
lts aged >= 18 years with diagnosed diabetes
870
                                       Foot examination among adults age
d >= 18 years with diagnosed diabetes
870
                                       Glycosylated hemoglobin measureme
```

```
nt among adults aged >= 18 years with diagnosed diabetes
870
                                        Influenza vaccination among nonin
stitutionalized adults aged 18-64 years with diagnosed diabetes
870
                                        Influenza vaccination among nonin
stitutionalized adults aged >= 65 years with diagnosed diabetes
870
                                        Mortality due to diabetes reporte
d as any listed cause of death
1227
                                        Mortality with diabetic ketoacido
sis reported as any listed cause of death
1221
                                        Pneumococcal vaccination among no
ninstitutionalized adults aged 18-64 years with diagnosed diabetes
870
                                        Pneumococcal vaccination among no
ninstitutionalized adults aged >= 65 years with diagnosed diabetes
870
                                        Prevalence of depressive disorder
s among adults aged >= 18 years with diagnosed diabetes
870
                                        Prevalence of diagnosed diabetes
among adults aged >= 18 years
870
                                        Prevalence of high blood pressure
among adults aged >= 18 years with diagnosed diabetes
870
                                        Prevalence of high cholesterol am
ong adults aged >= 18 years with diagnosed diabetes
870
Name: Question, dtype: int64
```

In [89]:
```
# filter for certain conditions in the CDI dataset
cdi_adultprev = cdi[cdi['Question'].str.contains('>= 18') & cdi['Question
cdi_adultprev = cdi_adultprev[cdi_adultprev['Question'].str.contains('asth

# view the unique values in cdi_adultprev dataset
cdi_adultprev['Question'].unique()
cdi_adultprev['Topic'].unique()
cdi_adultprev
```

Out[89]:
```
array(['Current asthma prevalence among adults aged >= 18 years',
       'Prevalence of chronic obstructive pulmonary disease among adults
>= 18',
       'Prevalence of chronic kidney disease among adults aged >= 18 yea
rs',
       'Prevalence of diagnosed diabetes among adults aged >= 18 years',
       'High cholesterol prevalence among adults aged >= 18 years'],
      dtype=object)
```

Out[89]:
```
array(['Asthma', 'Chronic Obstructive Pulmonary Disease',
       'Chronic Kidney Disease', 'Diabetes', 'Cardiovascular Disease'],
      dtype=object)
```

Out[89]:

	YearStart	YearEnd	LocationAbbr	LocationDesc	DataSource	Topic	Qu
529351	2019	2019	GU	Guam	BRFSS	Asthma	C a prev a age
529428	2019	2019	AK	Alaska	BRFSS	Asthma	C a prev a age
529446	2019	2019	AK	Alaska	BRFSS	Chronic Obstructive Pulmonary Disease	Prev of c obst puln
529539	2019	2019	AK	Alaska	BRFSS	Asthma	C a prev a age
529832	2019	2019	AK	Alaska	BRFSS	Chronic Kidney Disease	Prev of c d a
...	...	...	...	...	...	...	
1181450	2019	2019	WY	Wyoming	BRFSS	Chronic Obstructive Pulmonary Disease	Prev of c obst puln
1181526	2019	2019	WY	Wyoming	BRFSS	Cardiovascular Disease	chol prev a
1181911	2019	2019	WY	Wyoming	BRFSS	Chronic Obstructive Pulmonary Disease	a Prev of c obst puln
1181929	2019	2019	WY	Wyoming	BRFSS	Chronic Kidney Disease	Prev of c d a
1182038	2019	2019	WY	Wyoming	BRFSS	Diabetes	Prev diag di a ac

4350 rows × 34 columns

In [15]:
```
import geopandas as gpd
# load shapefile
US_states = gpd.read_file('tl_2022_us_state.zip')
US_states.head()
```

Out[15]:

	REGION	DIVISION	STATEFP	STATENS	GEOID	STUSPS	NAME	LSAD	MTFCC	FI
0	3	5	54	01779805	54	WV	West Virginia	00	G4000	
1	3	5	12	00294478	12	FL	Florida	00	G4000	
2	2	3	17	01779784	17	IL	Illinois	00	G4000	
3	2	4	27	00662849	27	MN	Minnesota	00	G4000	
4	3	5	24	01714934	24	MD	Maryland	00	G4000	

In [33]:
```
asthma = cdi_adultprev[cdi_adultprev['Topic'] == 'Asthma']

# merge US states shapefile with the asthma dataset
merge_asthma = US_states.merge(asthma, left_on='NAME',right_on='LocationDe
merge_asthma = merge_asthma.dropna(subset=['DataValue'])
merge_asthma['DataValue'] = merge_asthma['DataValue'].astype(float)

# create heat map
col = 'DataValue'
merge_asthma['DataValue'].dtypes
vmin = merge_asthma['DataValue'].min()
vmax = merge_asthma['DataValue'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_asthma.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap=cma
plt.xlim(-170,-60)
plt.ylim(20,80)
plt.title('Prevalence of Asthma in 2019', fontsize=20)

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

Out[33]: dtype('float64')

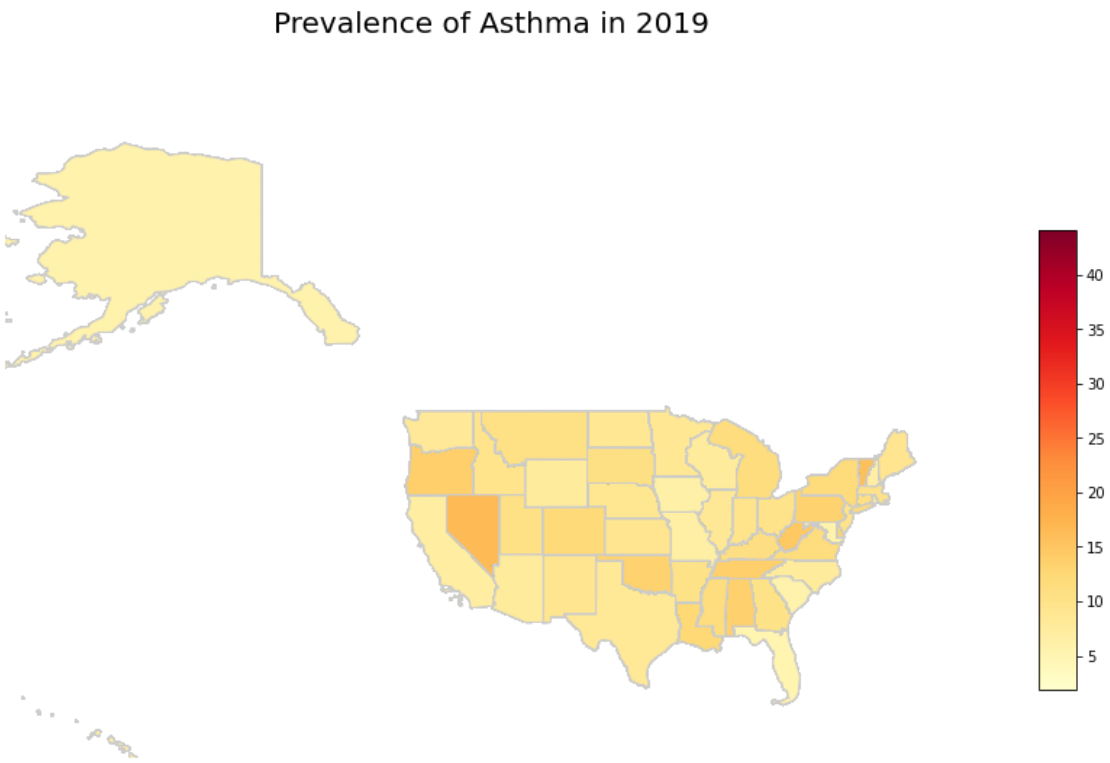
Out[33]: (0.0, 1.0, 0.0, 1.0)

Out[33]: <AxesSubplot:>

Out[33]: (-170.0, -60.0)

Out[33]: (20.0, 80.0)

Out[33]: Text(0.5, 1.0, 'Prevalence of Asthma in 2019')


Prevalence of Asthma in 2019
40
35
30
25
20
15
10
5

In [34]:
```
cardio = cdi_adultprev[cdi_adultprev['Topic'] == 'Cardiovascular Disease']

# merge US states shapefile with the cardio dataset
merge_cardio = US_states.merge(cardio, left_on='NAME',right_on='LocationDe
merge_cardio = merge_cardio.dropna(subset=['DataValue'])
merge_cardio['DataValue'] = merge_cardio['DataValue'].astype(float)

# create heatmap
col = 'DataValue'
merge_cardio['DataValue'].dtypes
vmin = merge_cardio['DataValue'].min()
vmax = merge_cardio['DataValue'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_cardio.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap=cma
plt.xlim(-170,-60)
plt.ylim(20,80)
plt.title('Prevalence of Cardiovascular Disease in 2019', fontsize=20)

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

Out[34]:
```
dtype('float64')
```

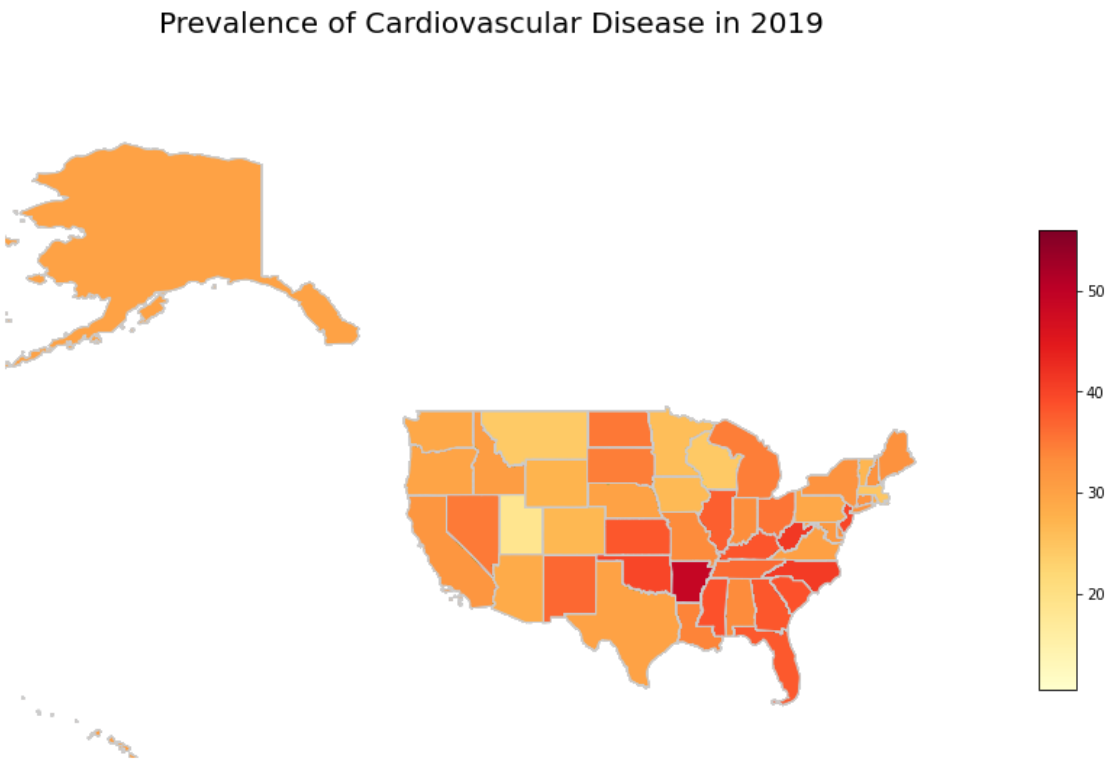
Out[34]:
```
(0.0, 1.0, 0.0, 1.0)
```

Out[34]:
```
<AxesSubplot:>
```

Out[34]:
```
(-170.0, -60.0)
```

Out[34]:
```
(20.0, 80.0)
```

Out[34]:
```
Text(0.5, 1.0, 'Prevalence of Cardiovascular Disease in 2019')
```


Prevalence of Cardiovascular Disease in 2019
50
40
30
20

In [35]:
```
kidney = cdi_adultprev[cdi_adultprev['Topic'] == 'Chronic Kidney Disease']

# merge US states shapefile with the kidney dataset
merge_kidney = US_states.merge(kidney, left_on='NAME',right_on='LocationDe
merge_kidney = merge_kidney.dropna(subset=['DataValue'])
merge_kidney['DataValue'] = merge_kidney['DataValue'].astype(float)

# create heatmap
col = 'DataValue'
merge_kidney['DataValue'].dtypes
vmin = merge_kidney['DataValue'].min()
vmax = merge_kidney['DataValue'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_kidney.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap=cma
plt.xlim(-170,-60)
plt.ylim(20,80)
plt.title('Prevalence of Chronic Kidney Disease in 2019', fontsize=20)

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

Out[35]:
```
dtype('float64')
```

Out[35]:
```
(0.0, 1.0, 0.0, 1.0)
```

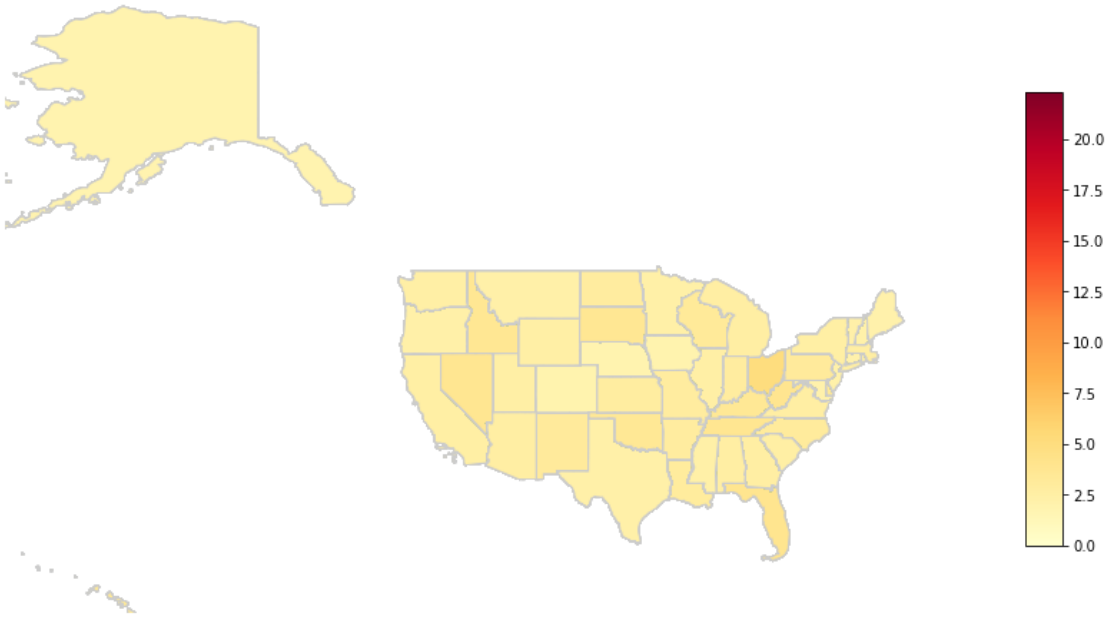
Out[35]:
```
<AxesSubplot:>
```

Out[35]:
```
(-170.0, -60.0)
```

Out[35]:
```
(20.0, 80.0)
```

Out[35]:
```
Text(0.5, 1.0, 'Prevalence of Chronic Kidney Disease in 2019')
```

Prevalence of Chronic Kidney Disease in 2019



In [62]:
```
copd = cdi_adultprev[cdi_adultprev['Topic'] == 'Chronic Obstructive Pulmor

# merge US states shapefile with the copd dataset
merge_copd = US_states.merge(copd, left_on='NAME',right_on='LocationDesc')
merge_copd = merge_copd.dropna(subset=['DataValue'])
merge_copd['DataValue'] = merge_copd['DataValue'].astype(float)

# create heatmap
col = 'DataValue'
merge_copd['DataValue'].dtypes
vmin = merge_copd['DataValue'].min()
vmax = merge_copd['DataValue'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_copd.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap=cmap)
plt.xlim(-170,-60)
plt.ylim(20,80)
plt.title('Prevalence of Chronic Obstructive Pulmonary Disease in 2019', f

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

Out[62]:
```
dtype('float64')
```

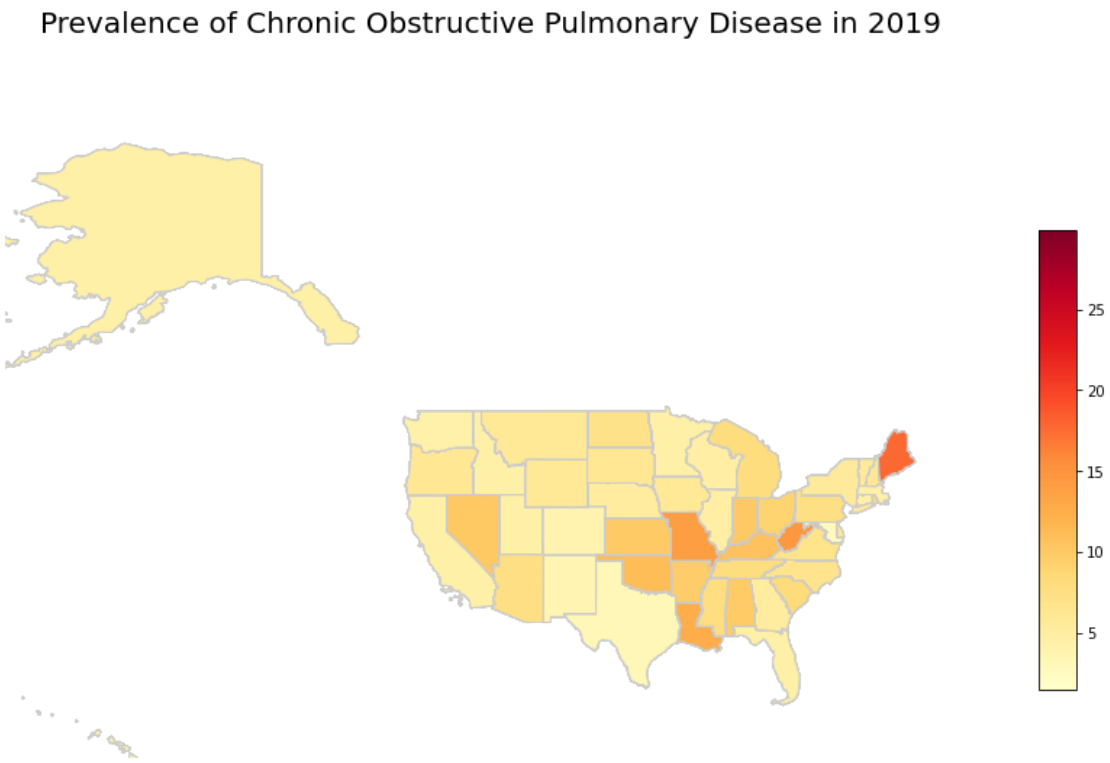
Out[62]:
```
(0.0, 1.0, 0.0, 1.0)
```

Out[62]:
```
<AxesSubplot:>
```

Out[62]:
```
(-170.0, -60.0)
```

Out[62]:
```
(20.0, 80.0)
```

Out[62]:
```
Text(0.5, 1.0, 'Prevalence of Chronic Obstructive Pulmonary Disease in 2
019')
```


Prevalence of Chronic Obstructive Pulmonary Disease in 2019
25
20
15
10
5

In [91]:
```
diabetes = cdi_adultprev[cdi_adultprev['Topic'] == 'Diabetes']

# merge US states shapefile with the diabetes dataset
merge_diabetes = US_states.merge(diabetes, left_on='NAME',right_on='Locati
merge_diabetes = merge_diabetes.dropna(subset=['DataValue'])
merge_diabetes['DataValue'] = merge_diabetes['DataValue'].astype(float)

# create heatmap
col = 'DataValue'
merge_diabetes['DataValue'].dtypes
vmin = merge_diabetes['DataValue'].min()
vmax = merge_diabetes['DataValue'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_diabetes.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap=
plt.xlim(-170,-60)
plt.ylim(20,80)
plt.title('Prevalence of Diabetes in 2019', fontsize=20)

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

Out[91]:
```
dtype('float64')
```

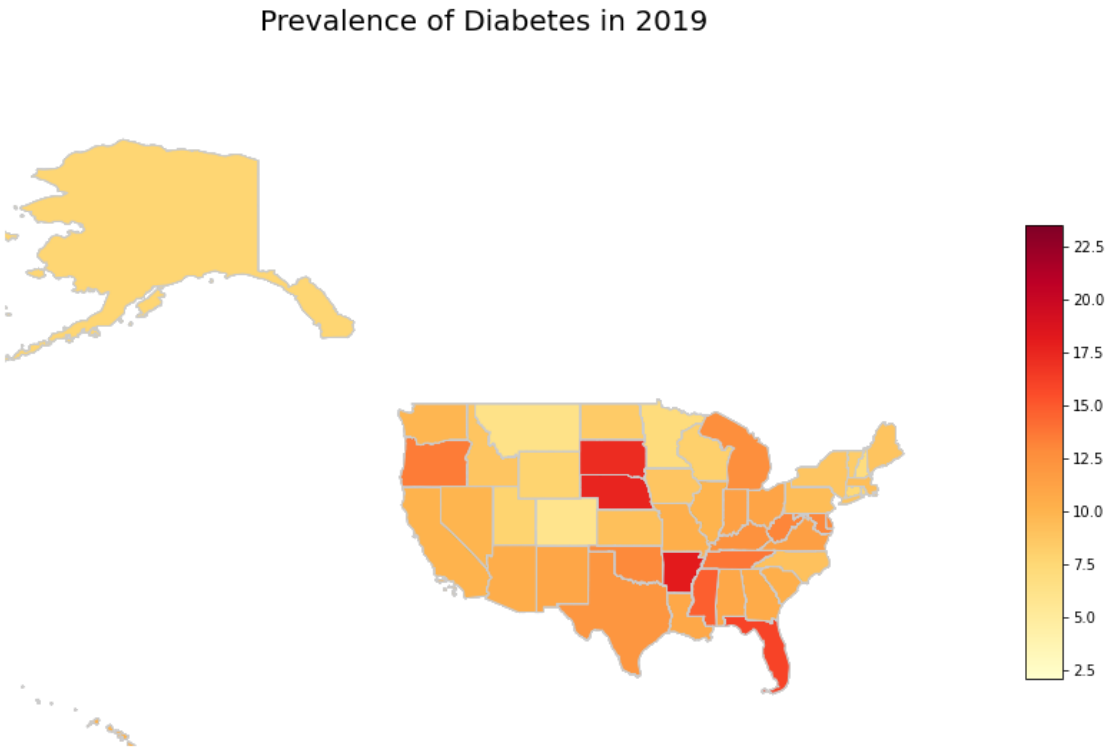
Out[91]:
```
(0.0, 1.0, 0.0, 1.0)
```

Out[91]:
```
<AxesSubplot:>
```

Out[91]:
```
(-170.0, -60.0)
```

Out[91]:
```
(20.0, 80.0)
```

Out[91]:
```
Text(0.5, 1.0, 'Prevalence of Diabetes in 2019')
```



In [36]:
```
# upload 2019 cancer dataset
cancer = pd.read_csv('respiratory cancer.csv')
cancer.head()
```

Out[36]:

	States	States Code	Year	Year Code	Cancer Sites	Cancer Sites Code	Count	Population	Age-Adjusted Rate	Crude Rate
0	Alabama	1	2019	2019	Nose, Nasal Cavity and Middle Ear	22010	41	4907965	0.7	0.8
1	Alabama	1	2019	2019	Larynx	22020	263	4907965	3.9	5.4
2	Alabama	1	2019	2019	Lung and Bronchus	22030	3956	4907965	59.5	80.6
3	Alaska	2	2019	2019	Lung and Bronchus	22030	391	733603	51.4	53.3
4	Alaska	2	2019	2019	Pleura	22050	0	733603	0.0	0.0

In [37]:
```
# filter for lung cancer
lung_cancer = cancer[cancer['Cancer Sites'] == 'Lung and Bronchus']
lung_cancer.set_index('States Code',inplace=True)
```

In [38]:
```
# merge US states shapefile with the lung cancer dataset
merge_lungcancer = US_states.merge(lung_cancer, left_on='NAME',right_on='S

# create Heat Value column
merge_lungcancer['Heat_Value'] = merge_lungcancer['Age-Adjusted Rate']

# create heatmap
col = 'Heat_Value'
vmin = merge_lungcancer['Heat_Value'].min()
vmax = merge_lungcancer['Heat_Value'].max()
cmap = 'YlOrRd'
fig, ax = plt.subplots(1, figsize=(12,11))
ax.axis('off')
merge_lungcancer.plot(column=col, ax=ax, edgecolor='0.8', linewidth=1,cmap
plt.title('Prevalence of Lung Cancer in 2019', fontsize=20)
plt.xlim(-170,-60)
plt.ylim(20,80)

# create colorbar
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=
sm._A = []
cbaxes = fig.add_axes([0.95,0.25,0.03,0.4])
cbar = fig.colorbar(sm, cbaxes)
```

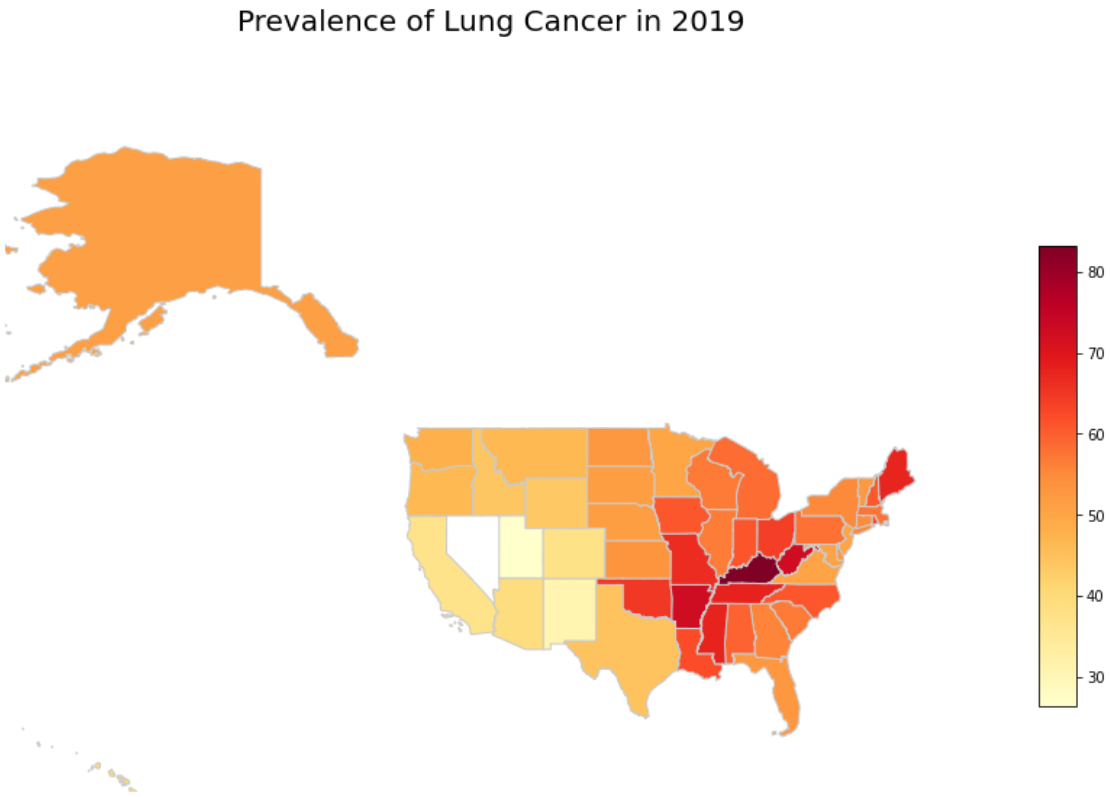
Out[38]:
```
(0.0, 1.0, 0.0, 1.0)
```

Out[38]:
```
<AxesSubplot:>
```

Out[38]:
```
Text(0.5, 1.0, 'Prevalence of Lung Cancer in 2019')
```

Out[38]:
```
(-170.0, -60.0)
```

Out[38]:
```
(20.0, 80.0)
```


Prevalence of Lung Cancer in 2019
80
70
60
50
40
30

In [39]:
```
# group by and average pollutant levels
grouped_one = one.groupby(['State Name', 'Parameter Name']).mean('Arithmet

# create a pivot table for the pollutant dataset
def list_to_columns(x):
    return pd.Series(x)
one_poll = pd.pivot_table(grouped_one, index='State Name', columns='Parame

# fix header row
one_poll = one_poll.reset_index()
one_poll.head()

# view how many monitors in each state for each pollutant
poll_freq = pd.crosstab(index=one['State Name'], columns=one['Parameter Na
poll_freq
```

Out[39]:

Parameter Name	State Name	Black Carbon PM2.5 at 880 nm	Carbon monoxide	Nitric oxide (NO)	Nitrogen dioxide (NO2)	Ozone	PM10 - LC	P Conc
0	Alabama	NaN	0.341273	7.768839	33.628643	0.089537	NaN	9.7
1	Alaska	NaN	0.527240	15.843333	26.082208	0.062254	16.179157	6.9
2	Arizona	NaN	0.352211	2.838873	38.372069	0.102257	NaN	6.5
3	Arkansas	NaN	0.427448	4.906377	24.736374	0.085546	NaN	
4	California	1.149278	0.361167	5.295208	25.656535	0.094085	24.486976	7.2

Out[39]:

Parameter Name	Black Carbon PM2.5 at 880 nm	Carbon monoxide	Nitric oxide (NO)	Nitrogen dioxide (NO2)	Ozone	PM10 - LC	PM2.5 - Local Conditions	Sulfur dioxide
State Name								
Alabama	0	3	3	4	22	0	3	12
Alaska	0	2	2	2	2	8	7	2
Arizona	0	10	4	16	64	0	17	14
Arkansas	0	1	3	4	8	0	0	2
California	4	63	99	216	175	68	92	62
Colorado	1	9	16	28	35	13	17	10
Connecticut	5	4	2	8	12	9	9	6
Country Of Mexico	0	0	0	0	1	0	0	0
Delaware	1	1	2	2	7	0	5	10
District Of Columbia	1	2	5	8	3	0	12	4
Florida	3	10	16	26	58	1	22	38
Georgia	0	2	4	6	19	0	12	14
Hawaii	0	3	1	4	2	0	27	24
Idaho	0	2	2	2	4	0	1	6
Illinois	0	4	7	14	37	0	17	30
Indiana	0	5	5	12	42	2	24	28
Iowa	0	2	3	4	24	0	13	16
Kansas	0	1	6	10	9	0	7	8
Kentucky	0	3	7	14	29	1	25	26
Louisiana	0	2	11	18	22	1	1	22
Maine	0	3	6	6	16	1	10	8
Maryland	2	4	6	16	20	0	11	14
Massachusetts	0	4	12	22	17	0	15	12
Michigan	6	10	12	18	29	0	11	34
Minnesota	0	7	6	10	17	0	24	12
Mississippi	0	1	2	2	10	1	8	4
Missouri	3	3	7	12	44	16	18	36
Montana	0	2	7	12	7	0	44	6
Nebraska	0	2	1	0	5	0	2	8
Nevada	0	7	3	16	23	4	13	4
New Hampshire	0	2	1	2	12	3	6	8

Parameter Name	Black Carbon PM2.5 at 880 nm	Carbon monoxide	Nitric oxide (NO)	Nitrogen dioxide (NO2)	Ozone	PM10 - LC	PM2.5 - Local Conditions	Sulfur dioxide
State Name								
New Jersey	0	6	12	22	17	0	12	18
New Mexico	0	2	12	22	21	1	11	8
New York	0	10	12	16	32	2	6	44
North Carolina	1	4	10	12	38	4	26	24
North Dakota	0	1	10	18	10	9	13	26
Ohio	3	13	10	16	51	1	17	68
Oklahoma	0	4	6	8	21	9	13	20
Oregon	0	2	3	4	10	0	2	4
Pennsylvania	1	10	20	38	53	0	52	46
Puerto Rico	0	5	3	4	3	0	1	6
Rhode Island	0	2	3	6	3	0	5	2
South Carolina	0	1	5	8	18	4	4	16
South Dakota	0	1	5	8	6	0	8	8
Tennessee	0	4	5	8	23	0	22	16
Texas	0	12	52	98	74	3	34	68
Utah	6	10	30	56	31	1	25	14
Vermont	0	3	3	4	3	4	4	4
Virgin Islands	0	0	0	0	0	0	1	0
Virginia	0	7	12	22	23	0	3	20
Washington	1	4	4	6	13	0	24	12
West Virginia	0	1	1	0	11	0	1	26
Wisconsin	0	2	4	6	31	9	23	14
Wyoming	0	3	27	50	28	0	14	30

In [101]:
```
# filter CDI dataset for overall stratification and age-adjusted rates
overall_cdi = cdi_adultprev[cdi_adultprev['StratificationCategoryID1']=='O
overall_cdi = overall_cdi[overall_cdi['DataValueTypeID']=='AGEADJPREV']
# create a pivot table for CDI dataset
def list_to_columns(x):
    return pd.Series(x)
cdi_fin = pd.pivot_table(overall_cdi, index='LocationDesc', columns='Topic

# fix header row for CDI_fin
cdi_fin = cdi_fin.reset_index()
cdi_fin.head()
```

Out[101]:

Topic	LocationDesc	Asthma	Cardiovascular Disease	Chronic Kidney Disease	Chronic Obstructive Pulmonary Disease	Diabetes
0	Alabama	9.4	32.7	3.1	9.3	12.2
1	Alaska	9.7	27.1	1.8	4.6	7.1
2	Arizona	9.8	30.5	3.6	6.0	9.8
3	Arkansas	9.3	33.0	3.7	9.7	12.2
4	California	7.8	28.2	2.8	4.2	9.4

In [102]:
```
# join datasets to create the final dataset
final = one_poll.merge(cdi_fin, left_on='State Name', right_on='LocationDe

# view the columns available in the final dataset
list(final.columns)

# select and drop specific columns
drop_cols = ['LocationDesc', 'States', 'Year Code', 'Cancer Sites Code',
final = final.drop(columns=drop_cols)

# rename columns in the final dataset
final = final.rename(columns={"Black Carbon PM2.5 at 880 nm" : "Black Carb

# view number of NaN values
final.isna().sum()

# view final dataset
final.head()
```

Out[102]:
```
['State Name',
 'Black Carbon PM2.5 at 880 nm',
 'Carbon monoxide',
 'Nitric oxide (NO)',
 'Nitrogen dioxide (NO2)',
 'Ozone',
 'PM10 - LC',
 'PM2.5 - Local Conditions',
 'Sulfur dioxide',
 'LocationDesc',
 'Asthma',
 'Cardiovascular Disease',
 'Chronic Kidney Disease',
 'Chronic Obstructive Pulmonary Disease',
 'Diabetes',
 'States',
 'Year',
 'Year Code',
 'Cancer Sites',
 'Cancer Sites Code',
 'Count',
 'Population',
 'Age-Adjusted Rate',
 'Crude Rate']
```

Out[102]:
```
State Name                          0
Black Carbon PM2.5                 35
Carbon monoxide                     0
Nitric oxide (NO)                   0
Nitrogen dioxide (NO2)              2
Ozone                               0
PM10 - LC                          25
PM2.5 - Local Conditions            1
Sulfur dioxide                      0
Asthma                              0
Cardiovascular Disease              0
Chronic Kidney Disease              0
COPD                                0
Diabetes                            0
Year                                0
Count                               0
Population                          0
Lung and Bronchus Cancer Rate       0
dtype: int64
```

Out[102]:

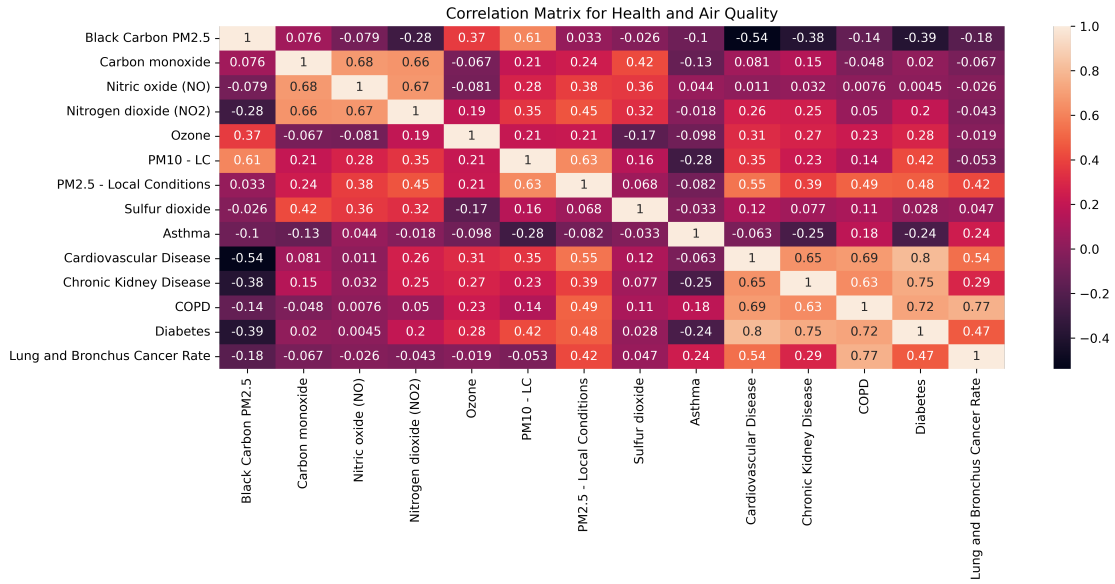
	State Name	Black Carbon PM2.5	Carbon monoxide	Nitric oxide (NO)	Nitrogen dioxide (NO2)	Ozone	PM10 - LC	PM2.5 - Local Conditions	
0	Alabama	NaN	0.341273	7.768839	33.628643	0.089537	NaN	9.799514	
1	Alaska	NaN	0.527240	15.843333	26.082208	0.062254	16.179157	6.903453	1
2	Arizona	NaN	0.352211	2.838873	38.372069	0.102257	NaN	6.557121	1
3	Arkansas	NaN	0.427448	4.906377	24.736374	0.085546	NaN	NaN	
4	California	1.149278	0.361167	5.295208	25.656535	0.094085	24.486976	7.232307	

In [107]:
```
# remove variables that are not needed
drop_col_2 = ['Year', 'Count', 'Population']
srp = final.drop(columns=drop_col_2)

# convert variables from object to float
srp.iloc[:,9:14] = srp.iloc[:,9:14].astype(float)

# create correlation matrix
fig = plt.figure(figsize=(14,5), dpi = 480)
plt.title('Correlation Matrix for Health and Air Quality')
sns.heatmap(srp.corr(), annot=True)
```

Out[107]:
```
State Name                          object
Black Carbon PM2.5                 float64
Carbon monoxide                    float64
Nitric oxide (NO)                  float64
Nitrogen dioxide (NO2)             float64
Ozone                              float64
PM10 - LC                          float64
PM2.5 - Local Conditions           float64
Sulfur dioxide                     float64
Asthma                             float64
Cardiovascular Disease             float64
Chronic Kidney Disease             float64
COPD                               float64
Diabetes                           float64
Lung and Bronchus Cancer Rate      float64
dtype: object
```

Out[107]:
```
Text(0.5, 1.0, 'Correlation Matrix for Health and Air Quality')
```

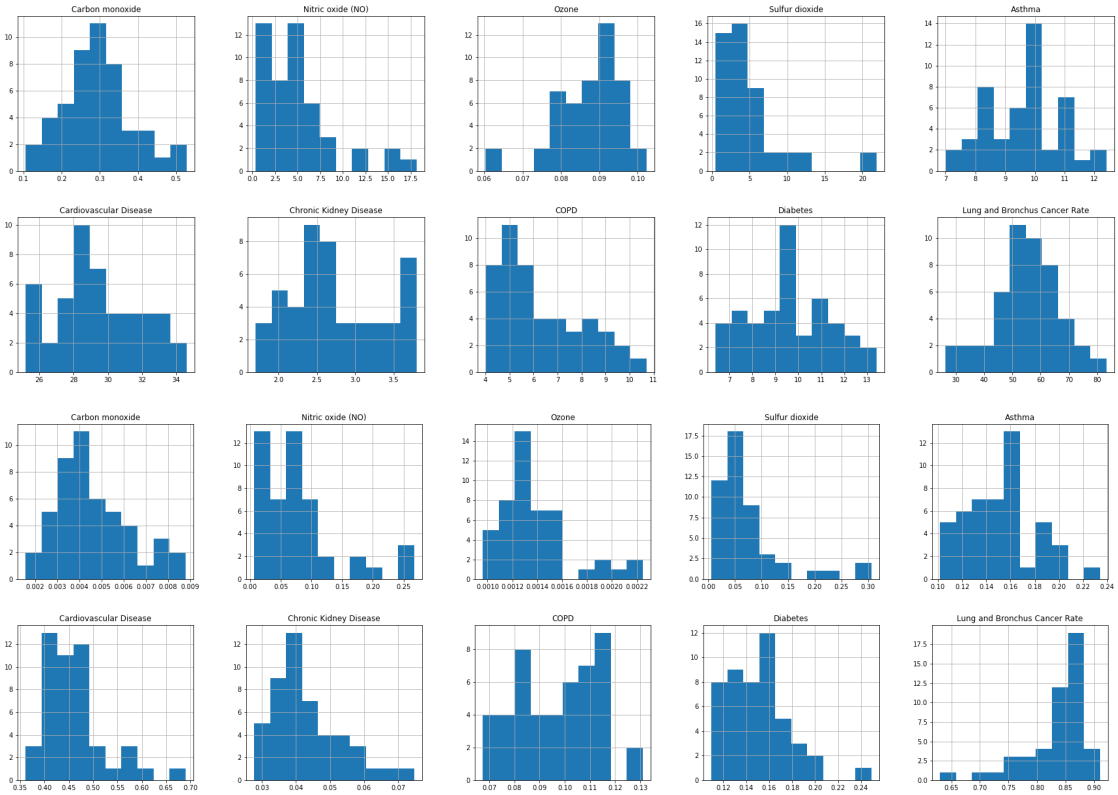
Out[107]:
```
<AxesSubplot:title={'center':'Correlation Matrix for Health and Air Quality'}>
```



In [108]:
```
# multiple linear regression
from sklearn import linear_model
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn import metrics

# adjust dataset for mlr
mlr_data = srp.iloc[:,1:]
#mlr_data.isna().sum()
drop_mlr_cols = ['Black Carbon PM2.5', 'Nitrogen dioxide (NO2)', 'PM10 - L
mlr_data = mlr_data.drop(columns=drop_mlr_cols)

# create histogram for each column in mlr_data
fig, axis = plt.subplots(2, 5, figsize=(30,10))
mlr_data.hist(ax=axis)

# normalize data
from sklearn import preprocessing
norm_data = preprocessing.normalize(mlr_data)
norm_data = pd.DataFrame(norm_data, columns=mlr_data.columns)


# create histogram for each column in norm_data
fig, axis = plt.subplots(2, 5, figsize=(30,10))
norm_data.hist(ax=axis)
```

Out[108]:
```
array([[<AxesSubplot:title={'center':'Carbon monoxide'}>,
        <AxesSubplot:title={'center':'Nitric oxide (NO)'}>,
        <AxesSubplot:title={'center':'Ozone'}>,
        <AxesSubplot:title={'center':'Sulfur dioxide'}>,
        <AxesSubplot:title={'center':'Asthma'}>],
       [<AxesSubplot:title={'center':'Cardiovascular Disease'}>,
        <AxesSubplot:title={'center':'Chronic Kidney Disease'}>,
        <AxesSubplot:title={'center':'COPD'}>,
        <AxesSubplot:title={'center':'Diabetes'}>,
        <AxesSubplot:title={'center':'Lung and Bronchus Cancer Rate'}
>]],
      dtype=object)
```

Out[108]:
```
array([[<AxesSubplot:title={'center':'Carbon monoxide'}>,
        <AxesSubplot:title={'center':'Nitric oxide (NO)'}>,
        <AxesSubplot:title={'center':'Ozone'}>,
        <AxesSubplot:title={'center':'Sulfur dioxide'}>,
        <AxesSubplot:title={'center':'Asthma'}>],
       [<AxesSubplot:title={'center':'Cardiovascular Disease'}>,
        <AxesSubplot:title={'center':'Chronic Kidney Disease'}>,
        <AxesSubplot:title={'center':'COPD'}>,
        <AxesSubplot:title={'center':'Diabetes'}>,
        <AxesSubplot:title={'center':'Lung and Bronchus Cancer Rate'}
>]],
      dtype=object)
```


Carbon monoxide
Nitric oxide (NO)
Ozone
Sulfur dioxide
Asthma
Cardiovascular Disease
Chronic Kidney Disease
COPD
Diabetes
Lung and Bronchus Cancer Rate
Carbon monoxide
Nitric oxide (NO)
Ozone
Sulfur dioxide
Asthma
Cardiovascular Disease
Chronic Kidney Disease
COPD
Diabetes
Lung and Bronchus Cancer Rate

In [143]:

```
# multiple linear regression for asthma
from sklearn.model_selection import train_test_split

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['Asthma']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for Asthma:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Scatterplot of Actual vs Predicted Prevalence Rates of Asthma')
plt.show()

# distribution of residuals
#sns.distplot((y_test - mlr_ypred)).set_title('Density Plot of Residuals f
#pip install scipy
#pip install statsmodels
#conda install -c conda-forge statsmodels
#import statsmodel.api as sm
!pip install statsmodels.api
import statsmodels.api as sm
residuals = y_test-mlr_ypred
fig, ax = plt.subplots(1, 1, figsize=(10, 4))
ax.set_title("QQ Plot of Residuals")
sm.qqplot(residuals, fit=True,line='45',ax=ax,c='#4C72B0')
# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

import statsmodels.api as sm
mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values
```

```
# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[143]:
```
LinearRegression()
```

```
Linear Regression Coefficients for Asthma:
```

Out[143]:

	0	1
0	Carbon monoxide	0.257828
1	Nitric oxide (NO)	0.036755
2	Ozone	57.051247
3	Sulfur dioxide	-0.003844

```
Linear Regression Intercept: 0.06853815566012296
```

Out[143]:

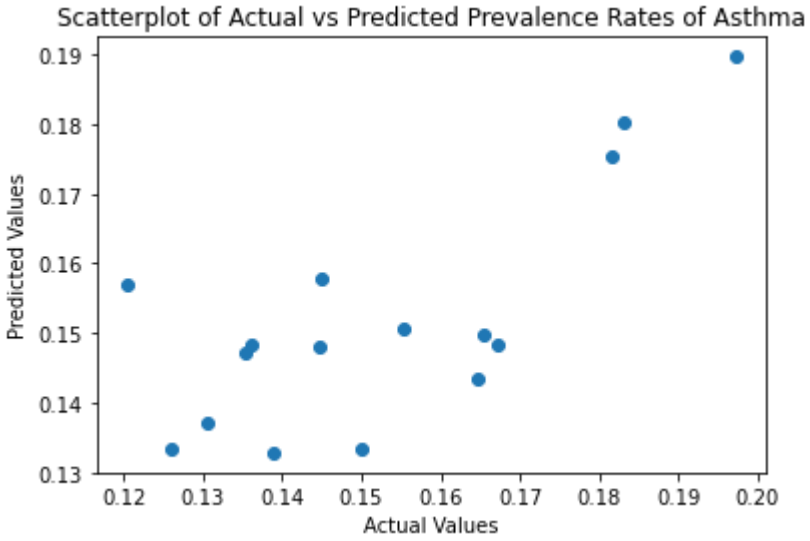
	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
26	0.005346	0.027863	0.001340	0.048832
35	0.004325	0.068692	0.001342	0.066115
29	0.004373	0.073111	0.001329	0.042034
31	0.003137	0.012340	0.001383	0.082327
47	0.002552	0.014610	0.001859	0.063306

Out[143]:
```
<matplotlib.collections.PathCollection at 0x16f4d84a460>
```

Out[143]:
```
Text(0.5, 0, 'Actual Values')
```

Out[143]:
```
Text(0, 0.5, 'Predicted Values')
```

Out[143]:
```
Text(0.5, 1.0, 'Scatterplot of Actual vs Predicted Prevalence Rates of A
sthma')
```

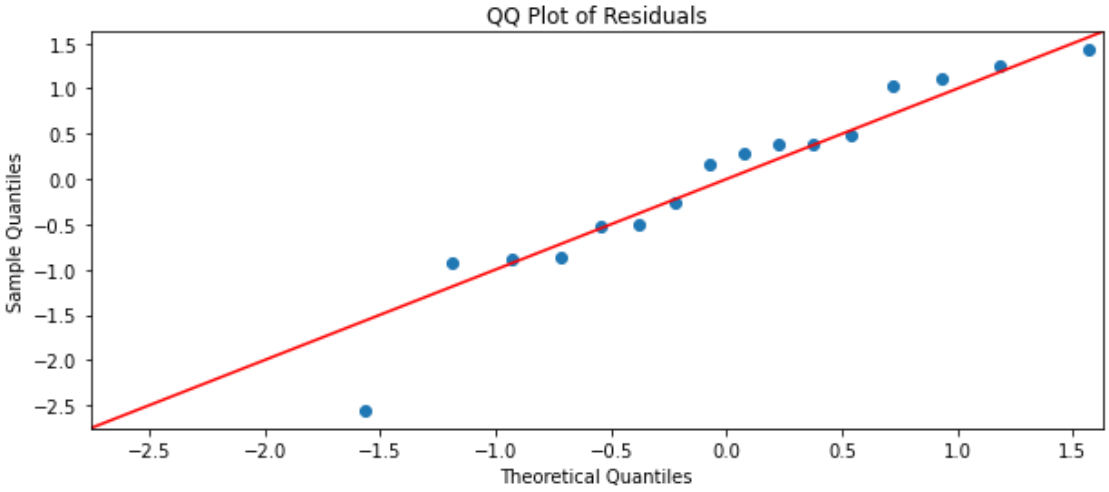



```
ERROR: Could not find a version that satisfies the requirement statsmode
ls.api (from versions: none)
ERROR: No matching distribution found for statsmodels.api
```

Out[143]:
```
Text(0.5, 1.0, 'QQ Plot of Residuals')
```

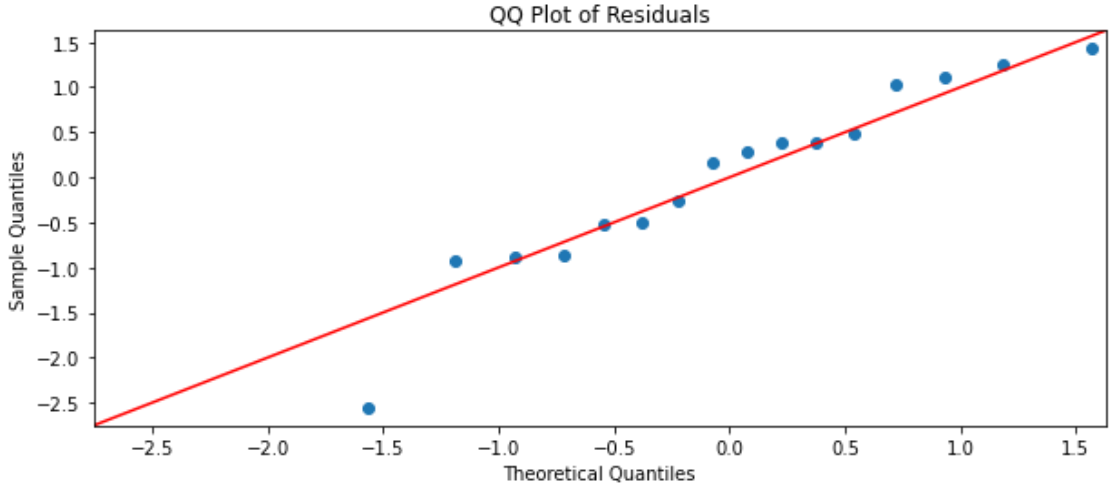
```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\statsmodels\graphics
\gofplots.py:1045: UserWarning: color is redundantly defined by the 'col
or' keyword argument and the fmt string "b" (-> color=(0.0, 0.0, 1.0,
1)). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```

Out[143]:




```
RMSE: 0.014513843185410326
R squared value: 0.354749081133537
```

Out[143]:
```
Carbon monoxide      8.320375e-01
Nitric oxide (NO)    5.463083e-01
Ozone                1.362516e-13
Sulfur dioxide       5.272349e-01
Name: P>|t|, dtype: float64

F statistic: 13.977275503439701
p-value for F statistic: 1.9447538657235697e-07
```


QQ Plot of Residuals
Sample Quantiles
Theoretical Quantiles
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

In [152]:

```
# multiple linear regression for cardiovascular disease

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['Cardiovascular Disease']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for Cardiovascular Disease:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)
len(mlr_ypred)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
p1 = max(max(mlr_ypred), max(y_test))
p2 = min(min(mlr_ypred), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')

plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Prevalence Rates of Cardiovascular Disease'
plt.show()

# distribution of residuals
residuals = y_test-mlr_ypred
fig, ax = plt.subplots(1, 1, figsize=(10, 4))
ax.set_title("QQ Plot of Residuals for Cardiovascular Disease MLR")
sm.qqplot(residuals, fit=True,line='45',ax=ax,c='#4C72B0')

# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values


# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
```

```
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[152]:
```
LinearRegression()
```

```
Linear Regression Coefficients for Cardiovascular Disease:
```

Out[152]:

	0	1
0	Carbon monoxide	9.553479
1	Nitric oxide (NO)	-0.107413
2	Ozone	169.605245
3	Sulfur dioxide	0.044602

```
Linear Regression Intercept: 0.18874967317519392
```

Out[152]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
37	0.001562	0.021218	0.001349	0.013345
8	0.004694	0.067004	0.001297	0.058707
42	0.004044	0.043474	0.001326	0.016566
19	0.003802	0.081459	0.001575	0.078319
9	0.007546	0.267061	0.001360	0.082329

Out[152]:
```
16
```

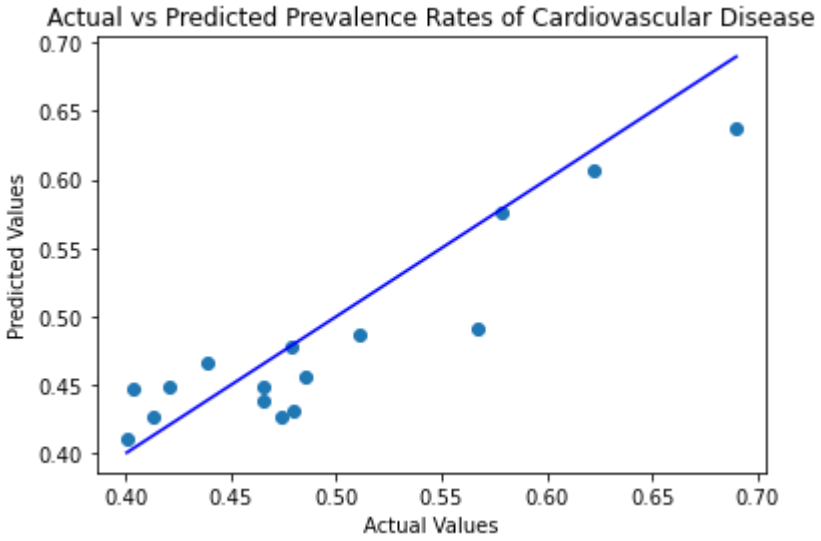
Out[152]:
```
<matplotlib.collections.PathCollection at 0x16f4df2d760>
```

Out[152]:
```
[<matplotlib.lines.Line2D at 0x16f4df03460>]
```

Out[152]:
```
Text(0.5, 0, 'Actual Values')
```

Out[152]:
```
Text(0, 0.5, 'Predicted Values')
```

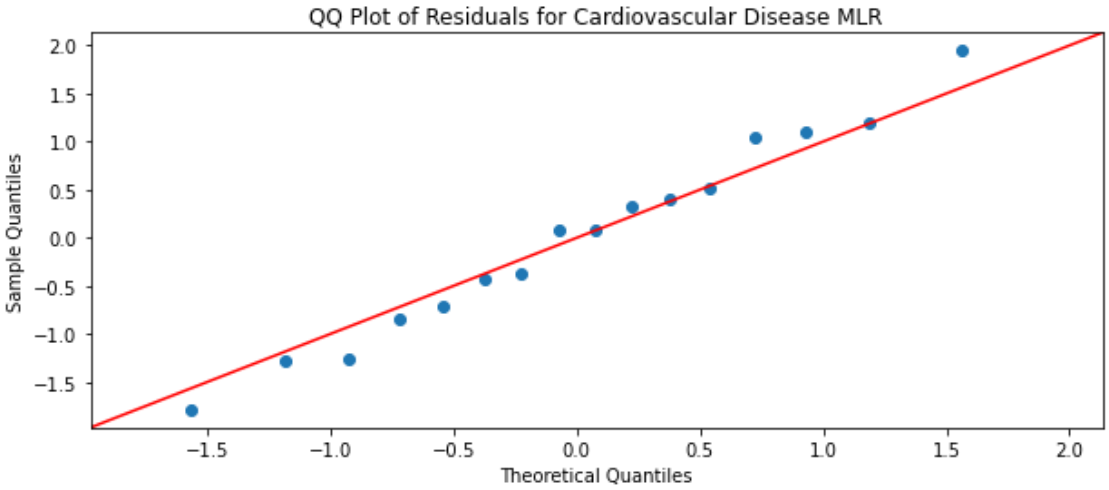
Out[152]:
```
Text(0.5, 1.0, 'Actual vs Predicted Prevalence Rates of Cardiovascular D
isease')
```



Out[152]: Text(0.5, 1.0, 'QQ Plot of Residuals for Cardiovascular Disease MLR')

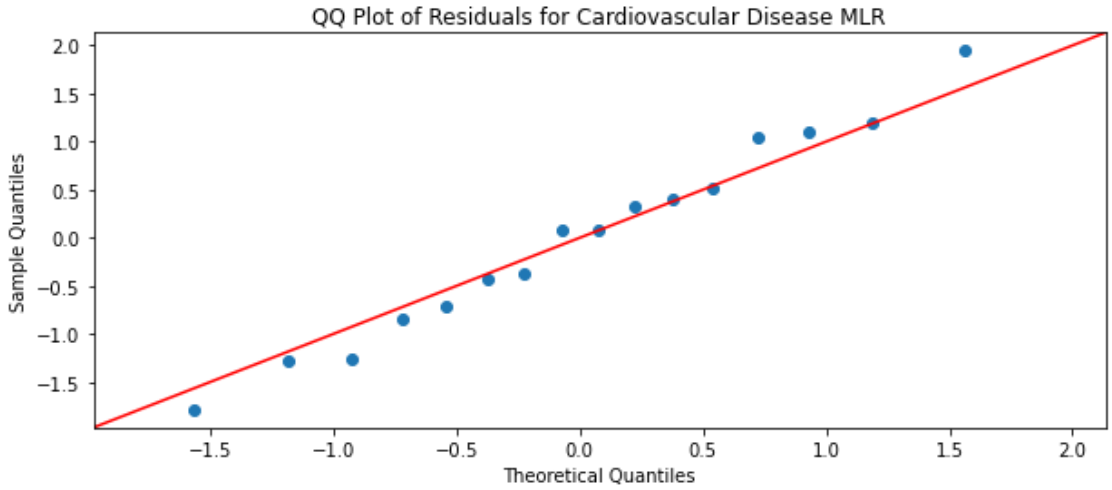
```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\statsmodels\graphics
\gofplots.py:1045: UserWarning: color is redundantly defined by the 'col
or' keyword argument and the fmt string "b" (-> color=(0.0, 0.0, 1.0,
1)). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```

Out[152]:




```
RMSE: 0.034927119610951005
R squared value: 0.7595572515262523
```

Out[152]:
```
Carbon monoxide      1.101957e-01
Nitric oxide (NO)    6.562002e-01
Ozone                1.461757e-20
Sulfur dioxide       1.678601e-01
Name: P>|t|, dtype: float64

F statistic: 41.67382529936937
p-value for F statistic: 1.9984014443252818e-14
```


QQ Plot of Residuals for Cardiovascular Disease MLR
Sample Quantiles
Theoretical Quantiles
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

In [141]:

```
# multiple linear regression for chronic kidney disease

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['Chronic Kidney Disease']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for Chronic Kidney Disease:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)
len(mlr_ypred)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Prevalence Rates of Chronic Kidney Disease'
plt.show()

# distribution of residuals
sns.distplot((y_test - mlr_ypred)).set_title('Density Plot of Residuals fo

# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values

# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
```

```
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[141]:
```
LinearRegression()
```

```
Linear Regression Coefficients for Chronic Kidney Disease:
```

Out[141]:

	0	1
0	Carbon monoxide	1.195397
1	Nitric oxide (NO)	-0.017510
2	Ozone	17.038947
3	Sulfur dioxide	0.005702

```
Linear Regression Intercept: 0.014836942197314226
```

Out[141]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
17	0.003638	0.034626	0.001114	0.061756
20	0.004234	0.076055	0.001334	0.017719
32	0.003382	0.095599	0.001280	0.052953
29	0.004373	0.073111	0.001329	0.042034
21	0.006379	0.096853	0.001228	0.095591

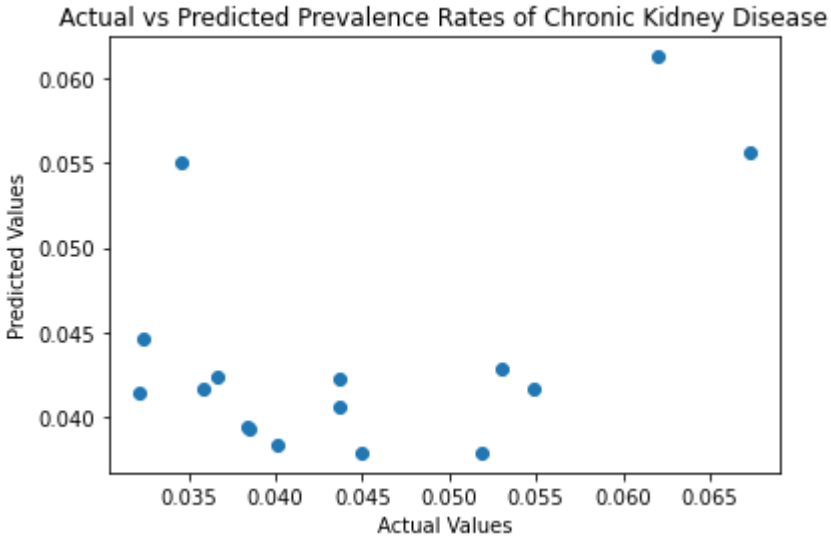
Out[141]:
```
16
```

Out[141]:
```
<matplotlib.collections.PathCollection at 0x16f4d69df40>
```

Out[141]:
```
Text(0.5, 0, 'Actual Values')
```

Out[141]:
```
Text(0, 0.5, 'Predicted Values')
```

Out[141]:
```
Text(0.5, 1.0, 'Actual vs Predicted Prevalence Rates of Chronic Kidney D
isease')
```



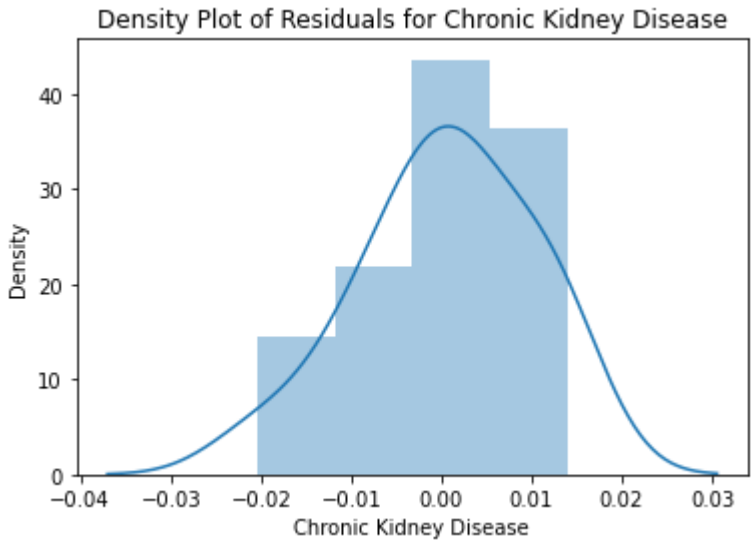

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[141]:
```
Text(0.5, 1.0, 'Density Plot of Residuals for Chronic Kidney Disease')
```

```
RMSE: 0.009331444528551286
R squared value: 0.3294827494197977
```

Out[141]:
```
Carbon monoxide      2.888981e-01
Nitric oxide (NO)    6.031292e-01
Ozone                1.043716e-10
Sulfur dioxide       4.830964e-01
Name: P>|t|, dtype: float64
```

```
F statistic: 1.5624039310503188
p-value for F statistic: 0.201124626479946
```



In [140]:

```
# multiple linear regression for COPD

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['COPD']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for COPD:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)
len(mlr_ypred)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Prevalence Rates of Chronic Kidney Disease'
plt.show()

# distribution of residuals
sns.distplot((y_test - mlr_ypred)).set_title('Density Plot of Residuals fo

# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values

# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
```

```
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[140]:
```
LinearRegression()
```

```
Linear Regression Coefficients for COPD:
```

Out[140]:

	0	1
0	Carbon monoxide	-3.507925
1	Nitric oxide (NO)	0.020296
2	Ozone	8.331242
3	Sulfur dioxide	0.023116

```
Linear Regression Intercept: 0.09790719509882989
```

Out[140]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
10	0.004418	0.021908	0.001192	0.094449
22	0.006098	0.072316	0.001327	0.054327
32	0.003382	0.095599	0.001280	0.052953
18	0.002462	0.015621	0.001028	0.006143
44	0.004818	0.245759	0.001255	0.129474

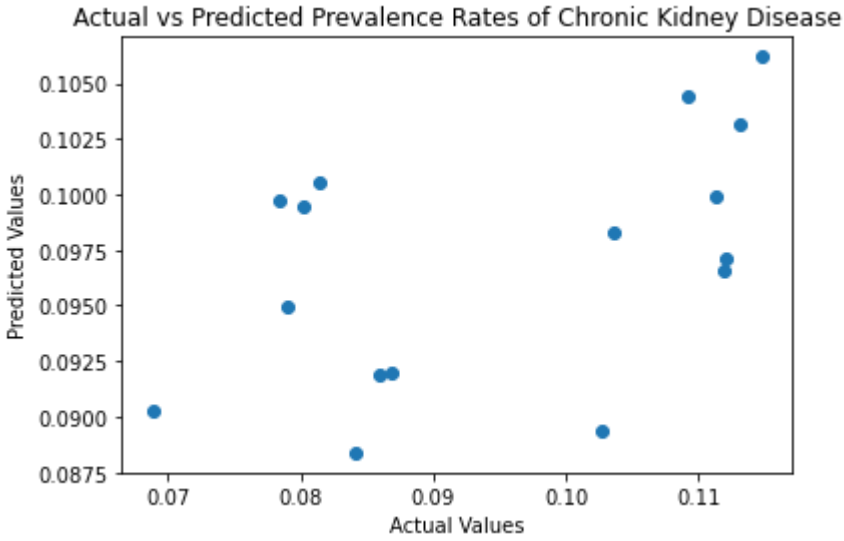
Out[140]:
```
16
```

Out[140]:
```
<matplotlib.collections.PathCollection at 0x16f4d5d5fd0>
```

Out[140]:
```
Text(0.5, 0, 'Actual Values')
```

Out[140]:
```
Text(0, 0.5, 'Predicted Values')
```

Out[140]:
```
Text(0.5, 1.0, 'Actual vs Predicted Prevalence Rates of Chronic Kidney D
isease')
```



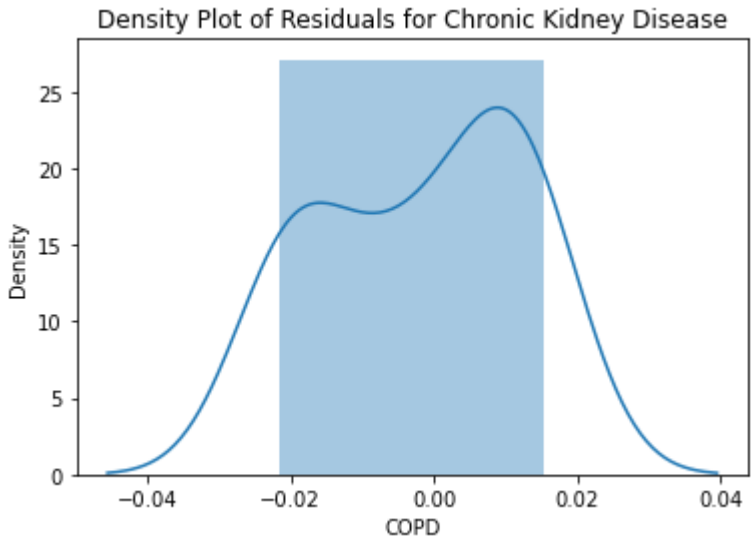

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[140]:
```
Text(0.5, 1.0, 'Density Plot of Residuals for Chronic Kidney Disease')
```

```
RMSE: 0.013666030236718219
R squared value: 0.06486818942343897
```

Out[140]:
```
Carbon monoxide      4.313636e-01
Nitric oxide (NO)    6.070580e-01
Ozone                7.227694e-10
Sulfur dioxide       1.623686e-01
Name: P>|t|, dtype: float64
```

```
F statistic: 0.93687803050809
p-value for F statistic: 0.4515216543467446
```



In [139]:

```
# multiple linear regression for Diabetes

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['Diabetes']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for Diabetes:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)
len(mlr_ypred)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Prevalence Rates of Diabetes')
plt.show()

# distribution of residuals
sns.distplot((y_test - mlr_ypred)).set_title('Density Plot of Residuals fo

# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values

# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
```

```
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[139]:
```
LinearRegression()
```

```
Linear Regression Coefficients for Diabetes:
```

Out[139]:

	0	1
0	Carbon monoxide	1.756228
1	Nitric oxide (NO)	-0.035480
2	Ozone	48.165777
3	Sulfur dioxide	-0.002440

```
Linear Regression Intercept: 0.07839957378806481
```

Out[139]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
18	0.002462	0.015621	0.001028	0.006143
34	0.004383	0.209265	0.001350	0.034207
6	0.004625	0.058390	0.001483	0.012228
17	0.003638	0.034626	0.001114	0.061756
30	0.004679	0.047913	0.001347	0.039445

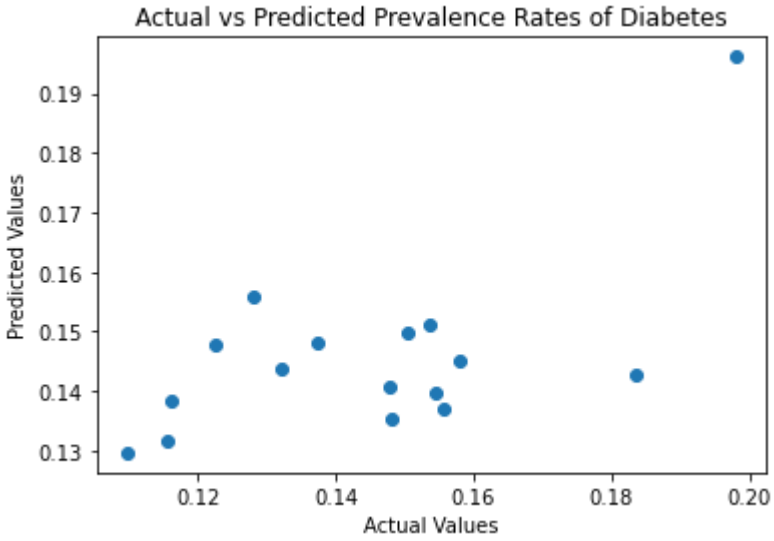
Out[139]:
```
16
```

Out[139]:
```
<matplotlib.collections.PathCollection at 0x16f4d484a90>
```

Out[139]:
```
Text(0.5, 0, 'Actual Values')
```

Out[139]:
```
Text(0, 0.5, 'Predicted Values')
```

Out[139]:
```
Text(0.5, 1.0, 'Actual vs Predicted Prevalence Rates of Diabetes')
```



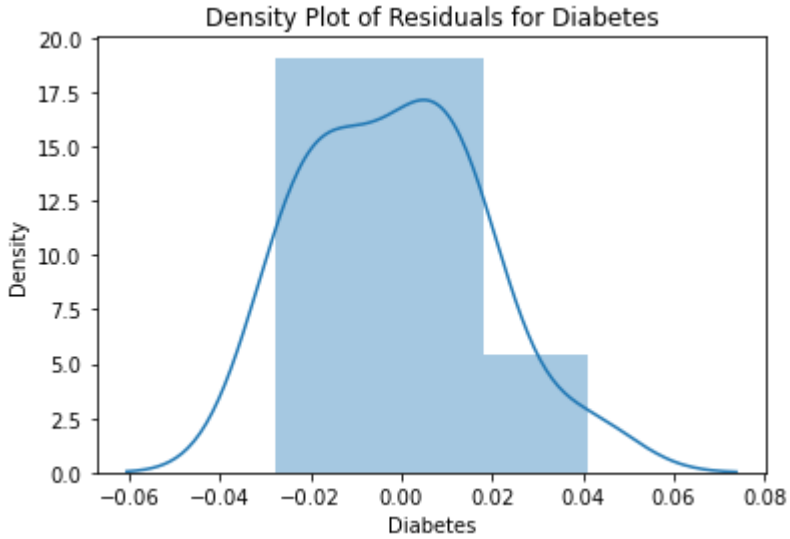

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[139]:
```
Text(0.5, 1.0, 'Density Plot of Residuals for Diabetes')
```

```
RMSE: 0.018442536830763627
R squared value: 0.300280762369334
```

Out[139]:
```
Carbon monoxide      5.421586e-01
Nitric oxide (NO)    8.254411e-01
Ozone                1.087366e-12
Sulfur dioxide       4.700053e-01
Name: P>|t|, dtype: float64
```

```
F statistic: 7.8466156517459
p-value for F statistic: 7.284120323625132e-05
```



In [151]:

```
# multiple linear regression for cancer

# determine x and y
mlr_x = norm_data.iloc[:,:4]
mlr_y = norm_data['Lung and Bronchus Cancer Rate']

# conduct mlr
mlr = linear_model.LinearRegression()
mlr.fit(mlr_x,mlr_y)

# print results
co_table = pd.DataFrame(zip(mlr_x.columns, mlr.coef_))
print("\nLinear Regression Coefficients for Lung and Bronchus Cancer:")
co_table
print("Linear Regression Intercept:", mlr.intercept_)

# train test split
X_train, X_test, y_train, y_test = train_test_split(mlr_x, mlr_y, test_siz
X_test.head()

# predict y values
mlr_ypred = mlr.predict(X_test)
len(mlr_ypred)

# scatterplot of actual vs predicted
plt.scatter(y_test, mlr_ypred)
p1 = max(max(mlr_ypred), max(y_test))
p2 = min(min(mlr_ypred), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')

plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Prevalence Rates of Lung and Bronchus Cance
plt.show()

# distribution of residuals
residuals = y_test-mlr_ypred
fig, ax = plt.subplots(1, 1, figsize=(10, 4))
ax.set_title("QQ Plot of Residuals for Lung/Bronchus Cancer MLR")
sm.qqplot(residuals, fit=True,line='45',ax=ax,c='#4C72B0')

# errors
mmae = metrics.mean_absolute_error(y_test, mlr_ypred)
mmse = np.sqrt(metrics.mean_squared_error(y_test, mlr_ypred))

print("RMSE:", mmse)

print("R squared value:", mlr.score(mlr_x,mlr_y))

mod = sm.OLS(mlr_y, mlr_x)
fii = mod.fit()
p_values = fii.summary2().tables[1]['P>|t|']
p_values


# calculate degrees of freedom
n_samples, n_features = mlr_x.shape
```

```
df_model = n_features
df_residual = n_samples - n_features

# calculate MSE
mse_model = np.sum((mlr_ypred - np.mean(mlr_y))**2) / df_model
mse_residual = np.sum(residuals**2)/df_residual

# f-statistic
f_stat = (mse_model/mse_residual)
print("F statistic:", f_stat)

# calculate p-value for F-statistic
from scipy.stats import f
fp_value = 1-f.cdf(f_stat, df_model, df_residual)
print("p-value for F statistic:", fp_value)
```

Out[151]:
```
LinearRegression()
```

```
Linear Regression Coefficients for Lung and Bronchus Cancer:
```

Out[151]:

	0	1
0	Carbon monoxide	-6.353958
1	Nitric oxide (NO)	-0.081793
2	Ozone	-129.491146
3	Sulfur dioxide	-0.188285

```
Linear Regression Intercept: 1.0661601974095771
```

Out[151]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide
31	0.003137	0.012340	0.001383	0.082327
11	0.008033	0.092317	0.001419	0.214581
21	0.006379	0.096853	0.001228	0.095591
37	0.001562	0.021218	0.001349	0.013345
26	0.005346	0.027863	0.001340	0.048832

Out[151]:
```
16
```

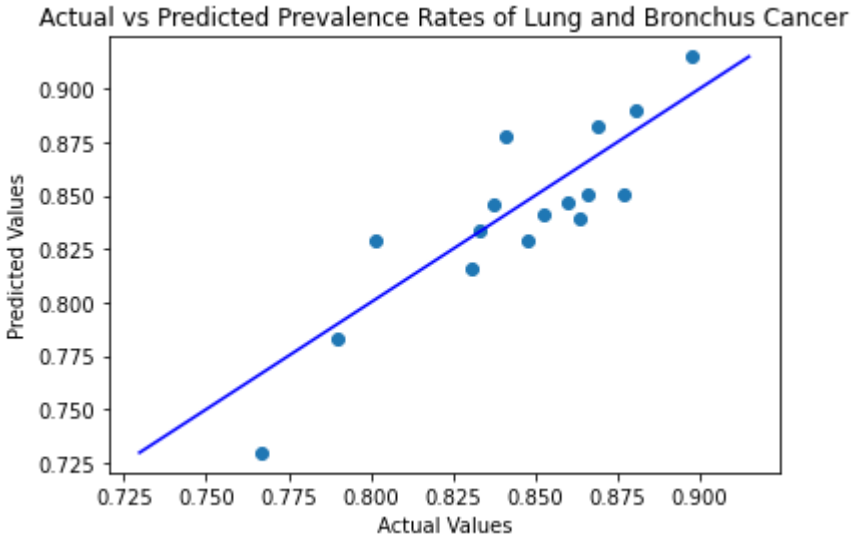
Out[151]:
```
<matplotlib.collections.PathCollection at 0x16f4df16df0>
```

Out[151]:
```
[<matplotlib.lines.Line2D at 0x16f4df27430>]
```

Out[151]:
```
Text(0.5, 0, 'Actual Values')
```

Out[151]:
```
Text(0, 0.5, 'Predicted Values')
```

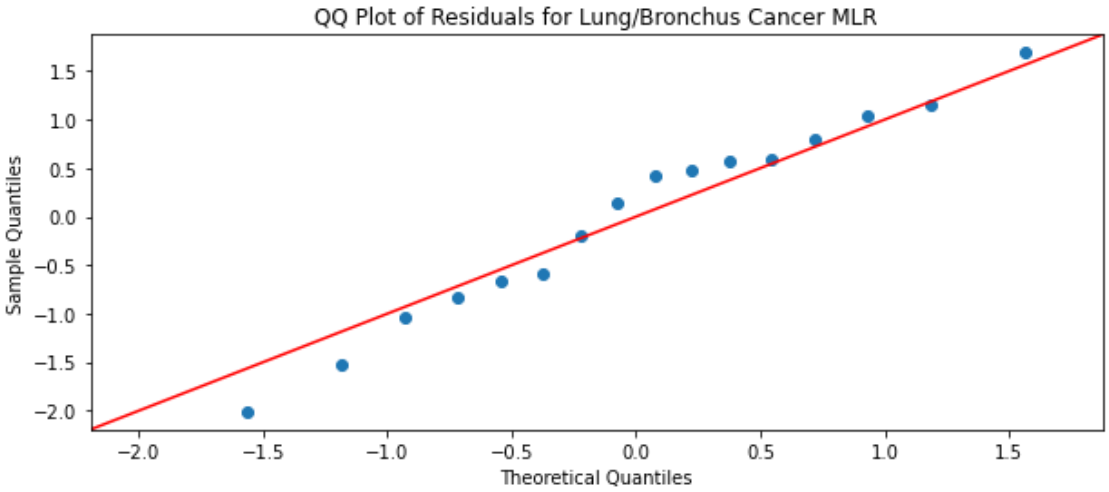
Out[151]:
```
Text(0.5, 1.0, 'Actual vs Predicted Prevalence Rates of Lung and Bronchu
s Cancer')
```



Out[151]: Text(0.5, 1.0, 'QQ Plot of Residuals for Lung/Bronchus Cancer MLR')

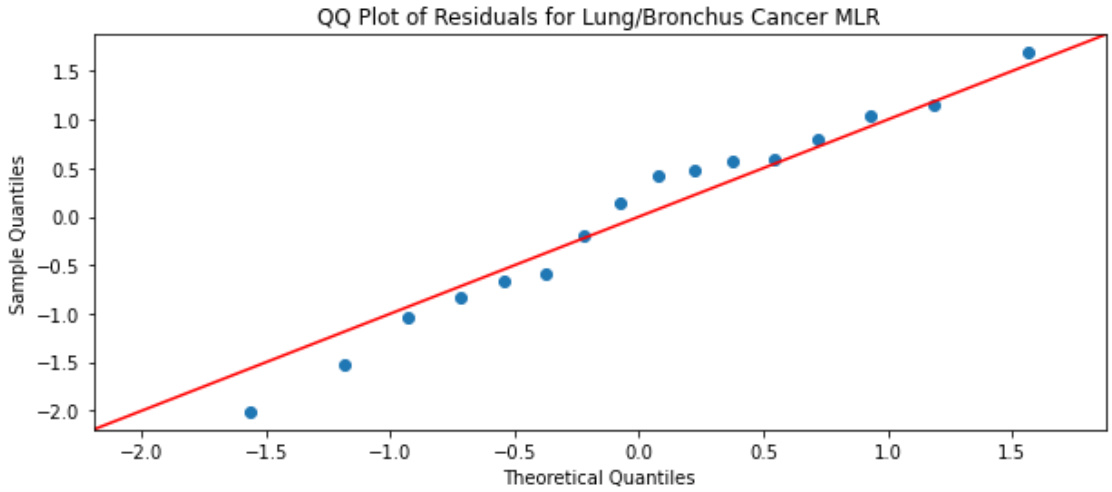
```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\statsmodels\graphics
\gofplots.py:1045: UserWarning: color is redundantly defined by the 'col
or' keyword argument and the fmt string "b" (-> color=(0.0, 0.0, 1.0,
1)). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```

Out[151]:




```
RMSE: 0.02019320103748823
R squared value: 0.776911059560959
```

Out[151]:
```
Carbon monoxide      9.477625e-01
Nitric oxide (NO)    8.355055e-01
Ozone                7.428161e-09
Sulfur dioxide       3.144067e-01
Name: P>|t|, dtype: float64

F statistic: 47.294870772808586
p-value for F statistic: 2.220446049250313e-15
```


QQ Plot of Residuals for Lung/Bronchus Cancer MLR
Sample Quantiles
Theoretical Quantiles
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

In [123]:
```
# scale data
norm_data.head()
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(norm_data)
scaled = scaler.transform(norm_data)
type(scaled)

# kmeans clustering
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3)

# cluster assignment for each observation
y_kmeans = kmeans.fit_predict(scaled)
y_kmeans

# centers of clusters
centers = kmeans.cluster_centers_
centers

from sklearn.metrics import silhouette_samples, silhouette_score
silhouette_score(scaled, y_kmeans)

# visualize clusters
plt.scatter(scaled[:,0], scaled[:,9], c=y_kmeans,cmap="rainbow")
plt.xlabel('Carbon Monoxide (µg/m³)')
plt.ylabel('Rate of Lung/Bronchus Cancer Prevalence')
plt.title('Cluster Plot of Lung/Bronchus Cancer and Carbon Monoxide')
```

Out[123]:

	Carbon monoxide	Nitric oxide (NO)	Ozone	Sulfur dioxide	Asthma	Cardiovascular Disease	Chronic Kidney Disease	COPD
0	0.004833	0.110016	0.001268	0.093632	0.133115	0.133115	0.463071	0.043900
1	0.008105	0.243543	0.000957	0.304240	0.149108	0.149108	0.416580	0.027670
2	0.006579	0.053031	0.001910	0.228478	0.183067	0.183067	0.569748	0.067249
3	0.005213	0.059842	0.001043	0.045230	0.113430	0.113430	0.402493	0.045128
4	0.007450	0.109221	0.001941	0.038439	0.160886	0.160886	0.581664	0.057754

Out[123]:
```
StandardScaler()
```

Out[123]:
```
numpy.ndarray
```

Out[123]:
```
array([0, 2, 1, 0, 1, 1, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 1, 2, 0, 2, 0, 0, 2, 2, 2, 0, 2, 0, 2, 1, 2, 2,
       2, 0, 2, 2])
```

Out[123]:
```
array([[-4.26187024e-01, -1.14175974e-01, -5.86729774e-01,
        -9.70226877e-02, -7.94562649e-01, -7.94562649e-01,
        -5.41335464e-01,  5.10356284e-02,  8.35251663e-01,
         5.55655505e-01],
       [ 1.37455996e+00,  4.04796320e-01,  2.37619667e+00,
         2.58565168e-01,  1.49770964e+00,  1.49770964e+00,
         2.21393769e+00,  1.61360539e+00,  2.34652613e-01,
        -2.28636617e+00],
       [ 3.19426663e-02,  1.24743749e-03, -5.27938960e-02,
         1.81433015e-02,  2.72543179e-01,  2.72543179e-01,
        -5.30260045e-02, -3.59466730e-01, -6.48311720e-01,
         5.72012717e-02]])
```

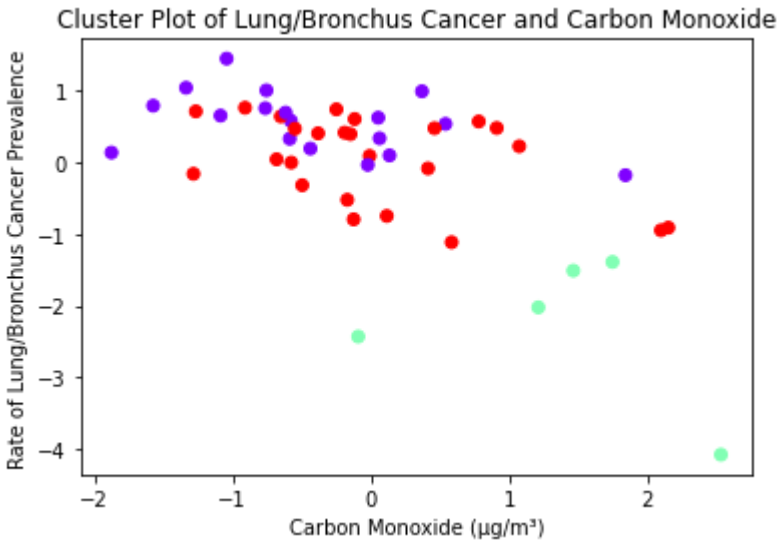
Out[123]:
```
0.1843118909996279
```

Out[123]:
```
<matplotlib.collections.PathCollection at 0x2115dd1db50>
```

Out[123]:
```
Text(0.5, 0, 'Carbon Monoxide (µg/m³)')
```

Out[123]:
```
Text(0, 0.5, 'Rate of Lung/Bronchus Cancer Prevalence')
```

Out[123]:
```
Text(0.5, 1.0, 'Cluster Plot of Lung/Bronchus Cancer and Carbon Monoxid
e')
```



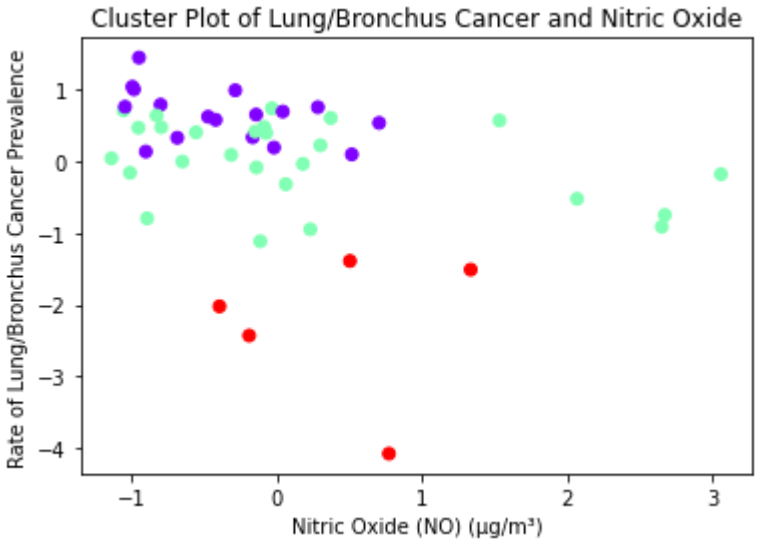
In [40]:
```
plt.scatter(scaled[:,1], scaled[:,9], c=y_kmeans,cmap="rainbow")
plt.xlabel('Nitric Oxide (NO) (µg/m³)')
plt.ylabel('Rate of Lung/Bronchus Cancer Prevalence')
plt.title('Cluster Plot of Lung/Bronchus Cancer and Nitric Oxide')
```

Out[40]:
```
<matplotlib.collections.PathCollection at 0x2111c051b50>
```

Out[40]:
```
Text(0.5, 0, 'Nitric Oxide (NO) (µg/m³)')
```

Out[40]:
```
Text(0, 0.5, 'Rate of Lung/Bronchus Cancer Prevalence')
```

Out[40]:
```
Text(0.5, 1.0, 'Cluster Plot of Lung/Bronchus Cancer and Nitric Oxide')
```



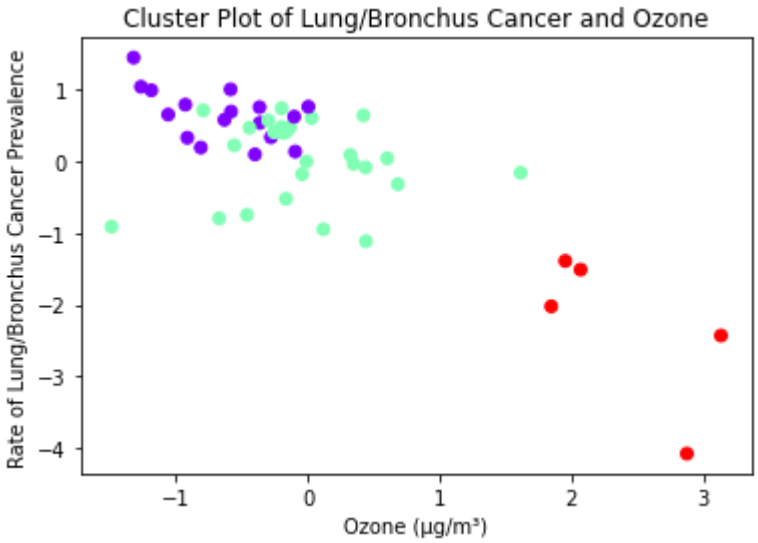
In [41]:
```
plt.scatter(scaled[:,2], scaled[:,9], c=y_kmeans,cmap="rainbow")
plt.xlabel('Ozone (µg/m³)')
plt.ylabel('Rate of Lung/Bronchus Cancer Prevalence')
plt.title('Cluster Plot of Lung/Bronchus Cancer and Ozone')
```

Out[41]: <matplotlib.collections.PathCollection at 0x211198d48b0>

Out[41]: Text(0.5, 0, 'Ozone (µg/m³)')

Out[41]: Text(0, 0.5, 'Rate of Lung/Bronchus Cancer Prevalence')

Out[41]: Text(0.5, 1.0, 'Cluster Plot of Lung/Bronchus Cancer and Ozone')



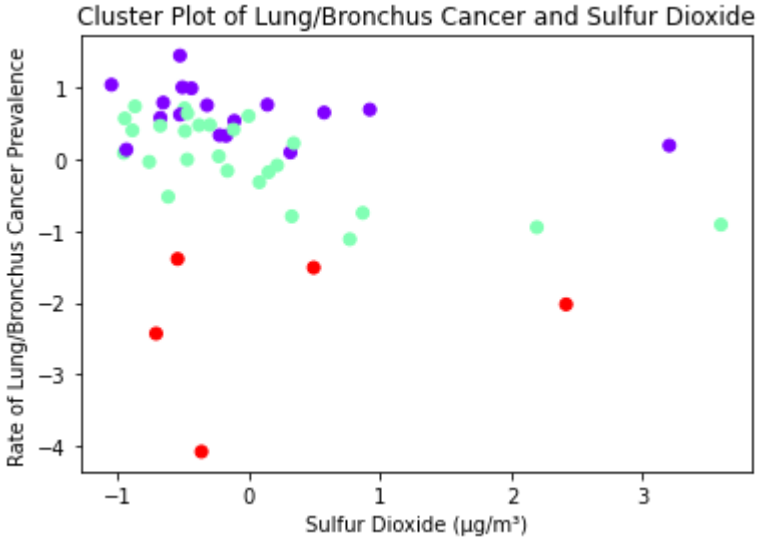
In [42]:
```
plt.scatter(scaled[:,3], scaled[:,9], c=y_kmeans,cmap="rainbow")
plt.xlabel('Sulfur Dioxide (µg/m³)')
plt.ylabel('Rate of Lung/Bronchus Cancer Prevalence')
plt.title('Cluster Plot of Lung/Bronchus Cancer and Sulfur Dioxide')
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x2111992b2b0>
```

Out[42]:
```
Text(0.5, 0, 'Sulfur Dioxide (µg/m³)')
```

Out[42]:
```
Text(0, 0.5, 'Rate of Lung/Bronchus Cancer Prevalence')
```

Out[42]:
```
Text(0.5, 1.0, 'Cluster Plot of Lung/Bronchus Cancer and Sulfur Dioxid
e')
```



In [145]:
```
from sklearn.decomposition import PCA

# apply PCA
pca = PCA()
pcs = pca.fit_transform(scaled)
pc = pca.components_

# variance percentages
pca.explained_variance_ratio_*100

plt.scatter(pcs[:,0], pcs[:,1], c = y_kmeans, cmap = "rainbow")
for i, (x, y) in enumerate(zip(pc[0, :], pc[1, :])):
    plt.arrow(0, 0, x, y, color='black', width=0.03, head_width=0.2)

plt.xlabel('PC1 45.3%')
plt.ylabel('PC2 17.9%')
plt.title('PCA Plot')
```

Out[145]:
```
array([4.53304959e+01, 1.78983721e+01, 1.58517673e+01, 7.68867336e+00,
       6.70831268e+00, 2.65785443e+00, 2.35986456e+00, 1.33329142e+00,
       1.71368243e-01, 2.16582552e-31])
```

Out[145]: <matplotlib.collections.PathCollection at 0x2115e0fbf40>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e108400>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e1086d0>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e108910>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e108b50>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e108d90>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e108dc0>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e0fbe50>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e111250>

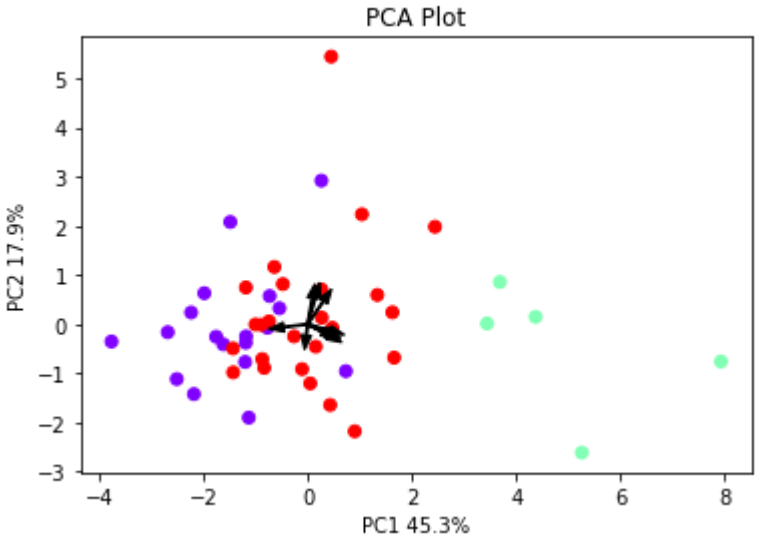
Out[145]: <matplotlib.patches.FancyArrow at 0x2115e111670>

Out[145]: <matplotlib.patches.FancyArrow at 0x2115e1117c0>

Out[145]: Text(0.5, 0, 'PC1 45.3%')

Out[145]: Text(0, 0.5, 'PC2 17.9%')

Out[145]: Text(0.5, 1.0, 'PCA Plot')


PCA Plot
PC2 17.9%
PC1 45.3%
5
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6
8

In [45]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_asthma = mlr_data.iloc[:,:5]
scaler.fit(mlr_asthma.drop('Asthma',axis=1))
scaled_asthma = scaler.transform(mlr_asthma.drop('Asthma', axis=1))
scaled_asthma = pd.DataFrame(scaled_asthma, columns = mlr_asthma.drop('Ast

# split dataset into training and test data
from sklearn.model_selection import train_test_split
asthma_x = scaled_asthma
asthma_y = mlr_asthma['Asthma']
x_train, x_test, y_train, y_test = train_test_split(asthma_x, asthma_y, te

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
asthma_model = KNeighborsClassifier(n_neighbors = 5)
asthma_knn = asthma_model.fit(x_train, y_train)

# predict values
asthma_predict = asthma_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,asthma_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(asthma_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", asthma_knn.score(x_test, y_test))
```

Out[45]:
```
StandardScaler()
```

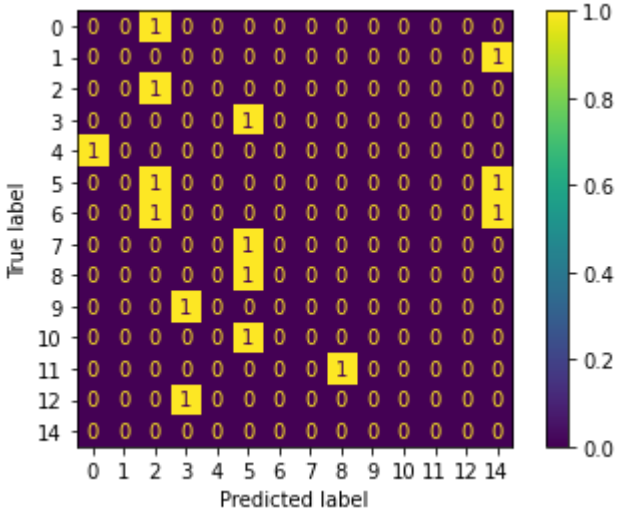
Out[45]:
```
LabelEncoder()
```

Out[45]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	1
2	0.25	1.00	0.40	1
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	2
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
10	0.00	0.00	0.00	1
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
14	0.00	0.00	0.00	0
accuracy			0.07	15
macro avg	0.02	0.07	0.03	15
weighted avg	0.02	0.07	0.03	15

```
[[0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.06666666666666667
```



In [79]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_copd = mlr_data.iloc[:,[0,1,2,3,7]]
scaler.fit(mlr_copd.drop('COPD',axis=1))
scaled_copd = scaler.transform(mlr_copd.drop('COPD', axis=1))
scaled_copd = pd.DataFrame(scaled_copd, columns = mlr_copd.drop('COPD', ax

# split dataset into training and test data
from sklearn.model_selection import train_test_split
copd_x = scaled_copd
copd_y = mlr_copd['COPD']
x_train, x_test, y_train, y_test = train_test_split(copd_x, copd_y, test_s

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
copd_model = KNeighborsClassifier(n_neighbors = 5)
copd_knn = copd_model.fit(x_train, y_train)

# predict values
copd_predict = copd_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,copd_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(copd_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", copd_knn.score(x_test, y_test))
```

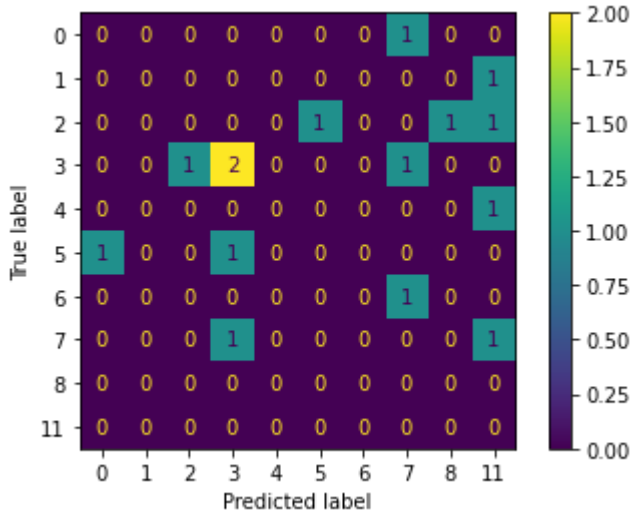
Out[79]:
```
StandardScaler()
```

Out[79]:
```
LabelEncoder()
```

Out[79]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         3
           3       0.50      0.50      0.50         4
           4       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         0
          11       0.00      0.00      0.00         0

    accuracy                           0.13        15
   macro avg       0.05      0.05      0.05        15
weighted avg       0.13      0.13      0.13        15

[[0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 1 0 0 1 1]
 [0 0 1 2 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1]
 [1 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0]
 [0 0 0 1 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.13333333333333333
```



In [135]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_cardio = mlr_data.iloc[:,[0,1,2,3,5]]
scaler.fit(mlr_cardio.drop('Cardiovascular Disease',axis=1))
scaled_cardio = scaler.transform(mlr_cardio.drop('Cardiovascular Disease',
scaled_cardio = pd.DataFrame(scaled_cardio, columns = mlr_cardio.drop('Car

# split dataset into training and test data
from sklearn.model_selection import train_test_split
cardio_x = scaled_cardio
cardio_y = mlr_cardio['Cardiovascular Disease']
x_train, x_test, y_train, y_test = train_test_split(cardio_x, cardio_y, te

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
cardio_model = KNeighborsClassifier(n_neighbors = 5)
cardio_knn = cardio_model.fit(x_train, y_train)

# predict values
cardio_predict = cardio_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,cardio_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(cardio_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", cardio_knn.score(x_test, y_test))
```

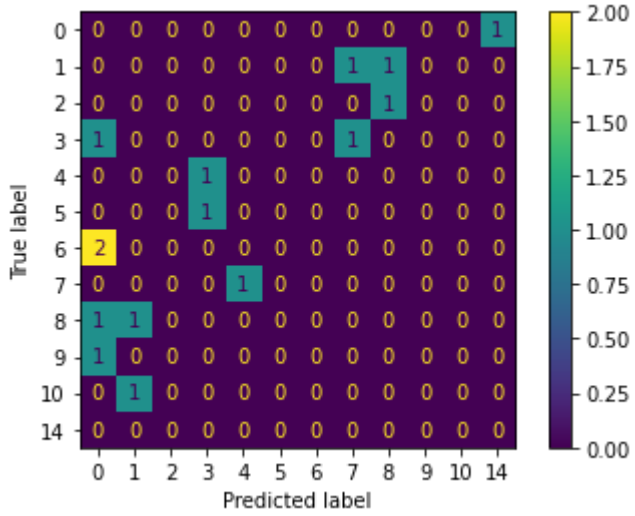
Out[135]:
```
StandardScaler()
```

Out[135]:
```
LabelEncoder()
```

Out[135]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       2.0
           2       0.00      0.00      0.00       1.0
           3       0.00      0.00      0.00       2.0
           4       0.00      0.00      0.00       1.0
           5       0.00      0.00      0.00       1.0
           6       0.00      0.00      0.00       2.0
           7       0.00      0.00      0.00       1.0
           8       0.00      0.00      0.00       2.0
           9       0.00      0.00      0.00       1.0
          10       0.00      0.00      0.00       1.0
          14       0.00      0.00      0.00       0.0

    accuracy                           0.00      15.0
   macro avg       0.00      0.00      0.00      15.0
weighted avg       0.00      0.00      0.00      15.0

[[0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 1 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0]
 [1 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0]
 [2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0]
 [1 1 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.0
```



In [84]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_kidney = mlr_data.iloc[:,[0,1,2,3,6]]
scaler.fit(mlr_kidney.drop('Chronic Kidney Disease',axis=1))
scaled_kidney = scaler.transform(mlr_kidney.drop('Chronic Kidney Disease',
scaled_kidney = pd.DataFrame(scaled_kidney, columns = mlr_kidney.drop('Chr

# split dataset into training and test data
from sklearn.model_selection import train_test_split
kidney_x = scaled_kidney
kidney_y = mlr_kidney['Chronic Kidney Disease']
x_train, x_test, y_train, y_test = train_test_split(kidney_x, kidney_y, te

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
kidney_model = KNeighborsClassifier(n_neighbors = 5)
kidney_knn = kidney_model.fit(x_train, y_train)

# predict values
kidney_predict = kidney_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,kidney_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(kidney_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", kidney_knn.score(x_test, y_test))
```

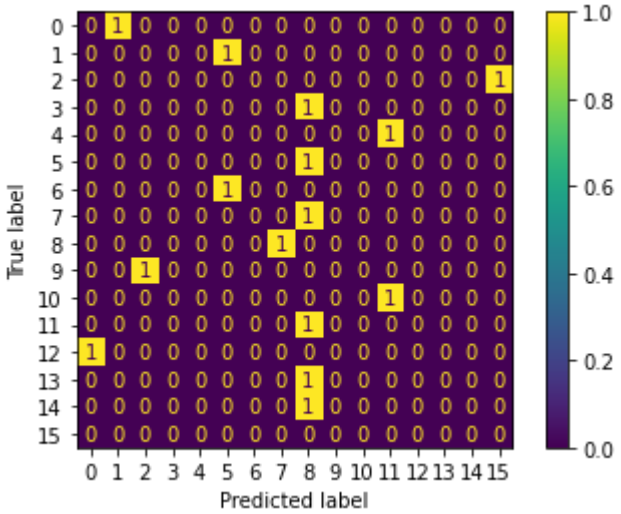
Out[84]:
```
StandardScaler()
```

Out[84]:
```
LabelEncoder()
```

Out[84]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
                   precision    recall  f1-score   support

                0       0.00      0.00      0.00       1.0
                1       0.00      0.00      0.00       1.0
                2       0.00      0.00      0.00       1.0
                3       0.00      0.00      0.00       1.0
                4       0.00      0.00      0.00       1.0
                5       0.00      0.00      0.00       1.0
                6       0.00      0.00      0.00       1.0
                7       0.00      0.00      0.00       1.0
                8       0.00      0.00      0.00       1.0
                9       0.00      0.00      0.00       1.0
               10       0.00      0.00      0.00       1.0
               11       0.00      0.00      0.00       1.0
               12       0.00      0.00      0.00       1.0
               13       0.00      0.00      0.00       1.0
               14       0.00      0.00      0.00       1.0
               15       0.00      0.00      0.00       0.0

    accuracy                           0.00      15.0
   macro avg       0.00      0.00      0.00      15.0
weighted avg       0.00      0.00      0.00      15.0

[[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.0
```



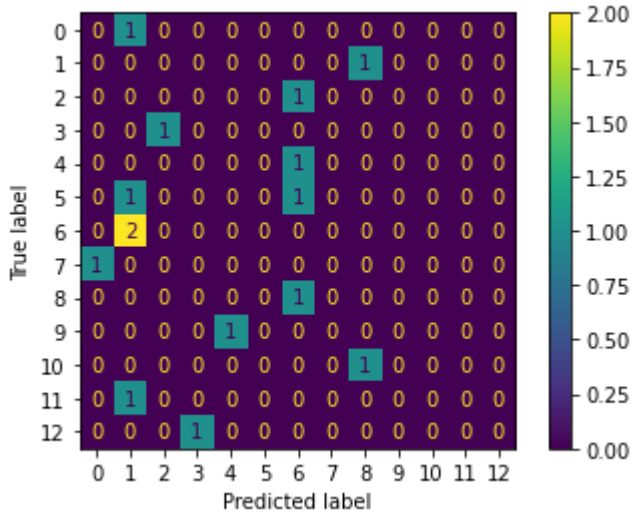
In [86]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_diabetes = mlr_data.iloc[:,[0,1,2,3,8]]
scaler.fit(mlr_diabetes.drop('Diabetes',axis=1))
scaled_diabetes = scaler.transform(mlr_diabetes.drop('Diabetes', axis=1))
scaled_diabetes = pd.DataFrame(scaled_diabetes, columns = mlr_diabetes.dro

# split dataset into training and test data
from sklearn.model_selection import train_test_split
dia_x = scaled_diabetes
dia_y = mlr_diabetes['Diabetes']
x_train, x_test, y_train, y_test = train_test_split(dia_x, dia_y, test_siz

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
diabetes_model = KNeighborsClassifier(n_neighbors = 5)
diabetes_knn = diabetes_model.fit(x_train, y_train)

# predict values
diabetes_predict = diabetes_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,diabetes_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(diabetes_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", diabetes_knn.score(x_test, y_test))
```

Out[86]:
```
StandardScaler()
```

Out[86]:
```
LabelEncoder()
```

Out[86]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       1.0
           2       0.00      0.00      0.00       1.0
           3       0.00      0.00      0.00       1.0
           4       0.00      0.00      0.00       1.0
           5       0.00      0.00      0.00       2.0
           6       0.00      0.00      0.00       2.0
           7       0.00      0.00      0.00       1.0
           8       0.00      0.00      0.00       1.0
           9       0.00      0.00      0.00       1.0
          10       0.00      0.00      0.00       1.0
          11       0.00      0.00      0.00       1.0
          12       0.00      0.00      0.00       1.0

    accuracy                           0.00      15.0
   macro avg       0.00      0.00      0.00      15.0
weighted avg       0.00      0.00      0.00      15.0

[[0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 1 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.0
```


True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

In [87]:
```
# standardize dataset
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mlr_lung = mlr_data.iloc[:,[0,1,2,3,9]]
scaler.fit(mlr_lung.drop('Lung and Bronchus Cancer Rate',axis=1))
scaled_lung = scaler.transform(mlr_lung.drop('Lung and Bronchus Cancer Rat
scaled_lung = pd.DataFrame(scaled_lung, columns = mlr_lung.drop('Lung and

# split dataset into training and test data
from sklearn.model_selection import train_test_split
lung_x = scaled_lung
lung_y = mlr_lung['Lung and Bronchus Cancer Rate']
x_train, x_test, y_train, y_test = train_test_split(lung_x, lung_y, test_s

# transform continuous variables
from sklearn import preprocessing
lab = preprocessing.LabelEncoder()
lab.fit(y_train)
y_train = lab.transform(y_train)
lab.fit(y_test)
y_test = lab.transform(y_test)

# train kNN model
from sklearn.neighbors import KNeighborsClassifier
lung_model = KNeighborsClassifier(n_neighbors = 5)
lung_knn = lung_model.fit(x_train, y_train)

# predict values
lung_predict = lung_model.predict(x_test)

# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,lung_predict))

# confusion matrix
from sklearn.metrics import plot_confusion_matrix
disp = plot_confusion_matrix(lung_knn, x_test, y_test)
print(disp.confusion_matrix)

# accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", lung_knn.score(x_test, y_test))
```

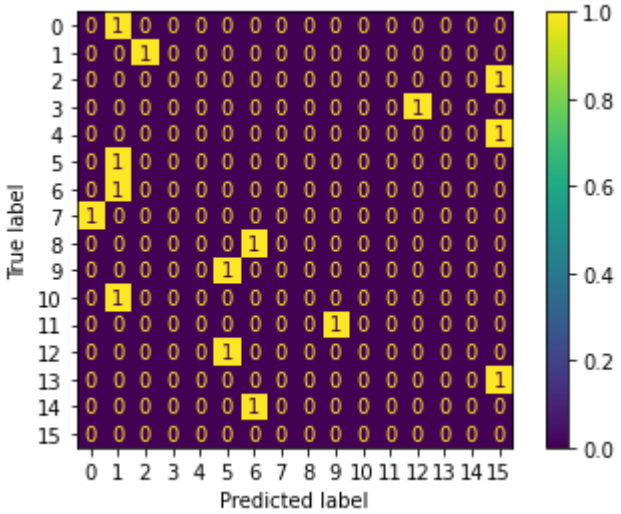
Out[87]:
```
StandardScaler()
```

Out[87]:
```
LabelEncoder()
```

Out[87]:
```
LabelEncoder()
```

```
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Precision and F-score are i
ll-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\metrics\_cla
ssification.py:1318: UndefinedMetricWarning: Recall and F-score are ill-
defined and being set to 0.0 in labels with no true samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amulya Cherian\anaconda3\lib\site-packages\sklearn\utils\deprec
ation.py:87: FutureWarning: Function plot_confusion_matrix is deprecate
d; Function `plot_confusion_matrix` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_p
redictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
                   precision    recall  f1-score   support

                0       0.00      0.00      0.00       1.0
                1       0.00      0.00      0.00       1.0
                2       0.00      0.00      0.00       1.0
                3       0.00      0.00      0.00       1.0
                4       0.00      0.00      0.00       1.0
                5       0.00      0.00      0.00       1.0
                6       0.00      0.00      0.00       1.0
                7       0.00      0.00      0.00       1.0
                8       0.00      0.00      0.00       1.0
                9       0.00      0.00      0.00       1.0
               10       0.00      0.00      0.00       1.0
               11       0.00      0.00      0.00       1.0
               12       0.00      0.00      0.00       1.0
               13       0.00      0.00      0.00       1.0
               14       0.00      0.00      0.00       1.0
               15       0.00      0.00      0.00       0.0

    accuracy                           0.00      15.0
   macro avg       0.00      0.00      0.00      15.0
weighted avg       0.00      0.00      0.00      15.0

[[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy: 0.0
```



In [ ]: