Importing, Understanding and Cleaning of Data

In [ ]:

```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
#importing initial necessary libraries

import pandas as pd
import numpy as np
```

In [ ]:

```
#importing datset

df = pd.read_csv('https://query.data.world/s/xlh353wvypzveoxm7h4u4c6hnucftk', encoding =
"ISO-8859-1")
```

In [ ]:

```
#looking at thr first five rows

df.head()
```

Out[ ]:

	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	studyPerCa
0	1397.0	469	164.9	489.8	61898	260131	11.2	499.74820
1	173.0	70	161.3	411.6	48127	43269	18.6	23.11123
2	102.0	50	174.7	349.7	49348	21026	14.6	47.56016
3	427.0	202	194.8	430.4	44243	75882	17.1	342.63725
4	57.0	26	144.4	350.1	49955	10321	12.5	0.00000

5 rows × 34 columns

Inspecting various aspects of the dataframe

In [ ]:

```
df.shape
```

Out[ ]:

```
(3047, 34)
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 34 columns):
```

```
Data columns (total 34 columns):
avgAnnCount                3047 non-null float64
avgDeathsPerYear           3047 non-null int64
TARGET_deathRate           3047 non-null float64
incidenceRate              3047 non-null float64
medIncome                  3047 non-null int64
popEst2015                 3047 non-null int64
povertyPercent             3047 non-null float64
studyPerCap                3047 non-null float64
binnedInc                  3047 non-null object
MedianAge                  3047 non-null float64
MedianAgeMale              3047 non-null float64
MedianAgeFemale            3047 non-null float64
Geography                  3047 non-null object
AvgHouseholdSize           3047 non-null float64
PercentMarried             3047 non-null float64
PctNoHS18_24               3047 non-null float64
PctHS18_24                 3047 non-null float64
PctSomeCol18_24            762 non-null float64
PctBachDeg18_24            3047 non-null float64
PctHS25_Over               3047 non-null float64
PctBachDeg25_Over          3047 non-null float64
PctEmployed16_Over         2895 non-null float64
PctUnemployed16_Over       3047 non-null float64
PctPrivateCoverage         3047 non-null float64
PctPrivateCoverageAlone    2438 non-null float64
PctEmpPrivCoverage         3047 non-null float64
PctPublicCoverage          3047 non-null float64
PctPublicCoverageAlone     3047 non-null float64
PctWhite                   3047 non-null float64
PctBlack                   3047 non-null float64
PctAsian                   3047 non-null float64
PctOtherRace               3047 non-null float64
PctMarriedHouseholds       3047 non-null float64
BirthRate                  3047 non-null float64
dtypes: float64(29), int64(3), object(2)
memory usage: 809.4+ KB
```

In [ ]:

```
df.isnull().sum() * 100 / len(df)
```

Out[ ]:

```
avgAnnCount                 0.000000
avgDeathsPerYear            0.000000
TARGET_deathRate            0.000000
incidenceRate               0.000000
medIncome                   0.000000
popEst2015                  0.000000
povertyPercent              0.000000
studyPerCap                 0.000000
binnedInc                   0.000000
MedianAge                   0.000000
MedianAgeMale               0.000000
MedianAgeFemale             0.000000
Geography                   0.000000
AvgHouseholdSize            0.000000
PercentMarried              0.000000
PctNoHS18_24                0.000000
PctHS18_24                  0.000000
PctSomeCol18_24            74.991795
PctBachDeg18_24             0.000000
PctHS25_Over                0.000000
PctBachDeg25_Over           0.000000
PctEmployed16_Over          4.988513
PctUnemployed16_Over        0.000000
PctPrivateCoverage          0.000000
PctPrivateCoverageAlone    19.986872
PctEmpPrivCoverage          0.000000
PctPublicCoverage           0.000000
PctPublicCoverageAlone      0.000000
PctWhite                    0.000000
```

```
PctBlack                   0.000000
PctAsian                   0.000000
PctOtherRace               0.000000
PctMarriedHouseholds       0.000000
BirthRate                  0.000000
dtype: float64
```

Since PctSomeCol18_24 is having 75%(approx), we can remove this column

In [ ]:

```
df = df.drop(['PctSomeCol18_24'], axis = 1)
```

Since PctPrivateCoverageAlone is having 20%(approx) missing values, we can remove the rows for which this value is missing

In [ ]:

```
df = df[df['PctPrivateCoverageAlone'].notna()]
```

In [ ]:

```
df.isnull().sum() * 100 / len(df)
```

Out[ ]:

```
avgAnnCount                0.000000
avgDeathsPerYear           0.000000
TARGET_deathRate           0.000000
incidenceRate              0.000000
medIncome                  0.000000
popEst2015                 0.000000
povertyPercent             0.000000
studyPerCap                0.000000
binnedInc                  0.000000
MedianAge                  0.000000
MedianAgeMale              0.000000
MedianAgeFemale            0.000000
Geography                  0.000000
AvgHouseholdSize           0.000000
PercentMarried             0.000000
PctNoHS18_24               0.000000
PctHS18_24                 0.000000
PctBachDeg18_24            0.000000
PctHS25_Over               0.000000
PctBachDeg25_Over          0.000000
PctEmployed16_Over         4.347826
PctUnemployed16_Over       0.000000
PctPrivateCoverage         0.000000
PctPrivateCoverageAlone    0.000000
PctEmpPrivCoverage         0.000000
PctPublicCoverage          0.000000
PctPublicCoverageAlone     0.000000
PctWhite                   0.000000
PctBlack                   0.000000
PctAsian                   0.000000
PctOtherRace               0.000000
PctMarriedHouseholds       0.000000
BirthRate                  0.000000
dtype: float64
```

Missing values are still there in column PctEmployed16_Over, again removing those rows having missing values in this column

In [ ]:

```
df = df[df['PctEmployed16_Over'].notna()]
```

In [ ]:

```
df.isnull().sum() * 100 / len(df)
```

Out[ ]:

```
avgAnnCount                0.0
avgDeathsPerYear           0.0
TARGET_deathRate           0.0
incidenceRate              0.0
medIncome                  0.0
popEst2015                 0.0
povertyPercent             0.0
studyPerCap                0.0
binnedInc                  0.0
MedianAge                  0.0
MedianAgeMale              0.0
MedianAgeFemale            0.0
Geography                  0.0
AvgHouseholdSize           0.0
PercentMarried             0.0
PctNoHS18_24               0.0
PctHS18_24                 0.0
PctBachDeg18_24            0.0
PctHS25_Over               0.0
PctBachDeg25_Over          0.0
PctEmployed16_Over         0.0
PctUnemployed16_Over       0.0
PctPrivateCoverage         0.0
PctPrivateCoverageAlone    0.0
PctEmpPrivCoverage         0.0
PctPublicCoverage          0.0
PctPublicCoverageAlone     0.0
PctWhite                   0.0
PctBlack                   0.0
PctAsian                   0.0
PctOtherRace               0.0
PctMarriedHouseholds       0.0
BirthRate                  0.0
dtype: float64
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2332 entries, 1 to 3046
Data columns (total 33 columns):
avgAnnCount                2332 non-null float64
avgDeathsPerYear           2332 non-null int64
TARGET_deathRate           2332 non-null float64
incidenceRate              2332 non-null float64
medIncome                  2332 non-null int64
popEst2015                 2332 non-null int64
povertyPercent             2332 non-null float64
studyPerCap                2332 non-null float64
binnedInc                  2332 non-null object
MedianAge                  2332 non-null float64
MedianAgeMale              2332 non-null float64
MedianAgeFemale            2332 non-null float64
Geography                  2332 non-null object
AvgHouseholdSize           2332 non-null float64
PercentMarried             2332 non-null float64
PctNoHS18_24               2332 non-null float64
PctHS18_24                 2332 non-null float64
PctBachDeg18_24            2332 non-null float64
PctHS25_Over               2332 non-null float64
PctBachDeg25_Over          2332 non-null float64
PctEmployed16_Over         2332 non-null float64
PctUnemployed16_Over       2332 non-null float64
PctPrivateCoverage         2332 non-null float64
PctPrivateCoverageAlone    2332 non-null float64
PctEmpPrivCoverage         2332 non-null float64
```

```
PctPublicCoverage           2332 non-null float64
PctPublicCoverageAlone      2332 non-null float64
PctWhite                    2332 non-null float64
PctBlack                    2332 non-null float64
PctAsian                    2332 non-null float64
PctOtherRace                2332 non-null float64
PctMarriedHouseholds        2332 non-null float64
BirthRate                   2332 non-null float64
dtypes: float64(28), int64(3), object(2)
memory usage: 619.4+ KB
```

No missing values are present now

In [ ]:

```
df.describe()
```

Out[ ]:

	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	stu
count	2332.000000	2332.000000	2332.000000	2332.000000	2332.000000	2.332000e+03	2332.000000	23
mean	609.979056	186.875214	178.918053	448.247260	46946.153516	1.035370e+05	16.961149	1
std	1440.417501	510.206766	27.482661	53.130088	12073.977204	3.428013e+05	6.432940	5
min	6.000000	3.000000	59.700000	201.300000	22640.000000	8.270000e+02	3.200000	
25%	75.000000	28.000000	161.400000	420.900000	38592.000000	1.154125e+04	12.200000	
50%	173.000000	62.000000	178.600000	453.549422	45079.000000	2.692300e+04	16.000000	
75%	517.250000	145.250000	195.325000	480.425000	52350.500000	6.803800e+04	20.600000	
max	38150.000000	14010.000000	293.900000	1014.200000	125635.000000	1.017029e+07	47.400000	97

8 rows × 31 columns

In [ ]:

```
#Checking for IQR, since we can figure out a lot of outliers present in the dataset

q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
IQR = q3 - q1
print(IQR)
```

```
avgAnnCount                   442.250000
avgDeathsPerYear              117.250000
TARGET_deathRate               33.925000
incidenceRate                  59.525000
medIncome                   13758.500000
popEst2015                  56496.750000
povertyPercent                  8.400000
studyPerCap                    79.990393
MedianAge                       6.100000
MedianAgeMale                   6.200000
MedianAgeFemale                 6.100000
AvgHouseholdSize                0.260000
PercentMarried                  8.600000
PctNoHS18_24                   10.100000
PctHS18_24                     11.400000
PctBachDeg18_24                 5.000000
PctHS25_Over                    9.000000
PctBachDeg25_Over               6.500000
PctEmployed16_Over             11.525000
PctUnemployed16_Over            4.100000
PctPrivateCoverage             15.125000
PctPrivateCoverageAlone        14.600000
PctEmpPrivCoverage             13.325000
PctPublicCoverage              10.900000
```

```
PctPublicCoverageAlone         8.400000
PctWhite                      18.436178
PctBlack                      10.450693
PctAsian                       0.954343
PctOtherRace                   1.880944
PctMarriedHouseholds           7.615916
BirthRate                      1.997401
dtype: float64
```

In [ ]:

```
df_out = df[~((df < (q1 - 1.5 * IQR)) |(df > (q3 + 1.5 * IQR))).any(axis=1)]
print(df_out.shape)
```

```
(944, 33)
```

I want to state a point over here, since i don't have business understanding related to the dataset given i can't decide which variables are important for me and which are not so that is why i have decided to remove all the data points which were outliers and did not considered the importance of that variable.

If you want to treat outlier and not remove them, I am doing it in one way (Capping) below, you can explore other ways

In [ ]:

```
df.skew()
```

Out[ ]:

```
avgAnnCount                10.852524
avgDeathsPerYear           12.086274
TARGET_deathRate            0.176615
incidenceRate              -0.016202
medIncome                   1.441171
popEst2015                 14.741562
povertyPercent              0.945489
studyPerCap                 8.640821
MedianAge                  10.457288
MedianAgeMale               0.098081
MedianAgeFemale            -0.236170
AvgHouseholdSize           -3.441399
PercentMarried             -0.688005
PctNoHS18_24                0.942008
PctHS18_24                  0.204684
PctBachDeg18_24             2.110282
PctHS25_Over               -0.341117
PctBachDeg25_Over           1.138598
PctEmployed16_Over         -0.364232
PctUnemployed16_Over        0.956774
PctPrivateCoverage         -0.382789
PctPrivateCoverageAlone    -0.011272
PctEmpPrivCoverage          0.097200
PctPublicCoverage          -0.047304
PctPublicCoverageAlone      0.463558
PctWhite                   -1.673339
PctBlack                    2.249444
PctAsian                    7.420264
PctOtherRace                5.090213
PctMarriedHouseholds       -0.610326
BirthRate                   1.443647
dtype: float64
```

In [ ]:

```
col = df.select_dtypes(include=['float64', 'int64'])
```

In [ ]:

```
for x in col:
    ten = df[x].quantile(0.10)
    nin = df[x].quantile(0.90)
    df[x] = np.where(df[x] < ten, ten,df[x])
    df[x] = np.where(df[x] > nin, nin,df[x])
```

In [ ]:

```
df.skew()
```

Out[ ]:

```
avgAnnCount                 1.637213
avgDeathsPerYear            1.322061
TARGET_deathRate            0.076585
incidenceRate              -0.247302
medIncome                   0.308420
popEst2015                  1.458895
povertyPercent              0.325621
studyPerCap                 1.665699
MedianAge                   0.027977
MedianAgeMale               0.026105
MedianAgeFemale            -0.088244
AvgHouseholdSize            0.196529
PercentMarried             -0.194958
PctNoHS18_24                0.261958
PctHS18_24                  0.022854
PctBachDeg18_24             0.386740
PctHS25_Over               -0.112480
PctBachDeg25_Over           0.425570
PctEmployed16_Over         -0.096057
PctUnemployed16_Over        0.175450
PctPrivateCoverage         -0.140704
PctPrivateCoverageAlone    -0.003782
PctEmpPrivCoverage          0.037438
PctPublicCoverage          -0.034643
PctPublicCoverageAlone      0.116176
PctWhite                   -0.885286
PctBlack                    1.347077
PctAsian                    1.210965
PctOtherRace                1.116929
PctMarriedHouseholds       -0.217603
BirthRate                   0.327491
dtype: float64
```

In [ ]:

```
df.describe()
```

Out[ ]:

	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	stu
count	2332.000000	2332.000000	2332.000000	2332.000000	2332.000000	2332.000000	2332.000000	23
mean	469.082597	109.530532	178.816012	449.315587	46057.169554	55113.109605	16.667007	
std	628.471413	112.548608	21.469822	39.367407	8626.632061	63234.224956	5.072359	1
min	36.000000	13.000000	146.110000	381.020000	34116.000000	5660.200000	9.900000	
25%	75.000000	28.000000	161.400000	420.900000	38592.000000	11541.250000	12.200000	
50%	173.000000	62.000000	178.600000	453.549422	45079.000000	26923.000000	16.000000	
75%	517.250000	145.250000	195.325000	480.425000	52350.500000	68038.000000	20.600000	
max	1962.667684	366.800000	213.500000	507.590000	61151.100000	203405.200000	25.390000	4

8 rows × 31 columns

In [ ]:

```
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
IQR = q3 - q1
print(IQR)
```

```
avgAnnCount                  442.250000
avgDeathsPerYear             117.250000
TARGET_deathRate              33.925000
incidenceRate                 59.525000
medIncome                  13758.500000
popEst2015                 56496.750000
povertyPercent                 8.400000
studyPerCap                   79.990393
MedianAge                      6.100000
MedianAgeMale                  6.200000
MedianAgeFemale                6.100000
AvgHouseholdSize               0.260000
PercentMarried                 8.600000
PctNoHS18_24                  10.100000
PctHS18_24                    11.400000
PctBachDeg18_24                5.000000
PctHS25_Over                   9.000000
PctBachDeg25_Over              6.500000
PctEmployed16_Over            11.525000
PctUnemployed16_Over           4.100000
PctPrivateCoverage            15.125000
PctPrivateCoverageAlone       14.600000
PctEmpPrivCoverage            13.325000
PctPublicCoverage             10.900000
PctPublicCoverageAlone         8.400000
PctWhite                      18.436178
PctBlack                      10.450693
PctAsian                       0.954343
PctOtherRace                   1.880944
PctMarriedHouseholds           7.615916
BirthRate                      1.997401
dtype: float64
```

In [ ]:

```
df_out2 = df[~((df < (q1 - 1.5 * IQR)) |(df > (q3 + 1.5 * IQR))).any(axis=1)]
print(df_out2.shape)
```

```
(1348, 33)
```

So now i will be moving on with df_out2 dataframe, Removal of outliers and missing value is done, cleaned the data

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns

#visulising numeric data using pair plot is of no use since there are a lot of numeric da
ta type columss
#lets check categorical variables binnedInc and geography

df_out2.binnedInc.unique()
```

Out[ ]:

```
array(['(48021.6, 51046.4]', '(51046.4, 54545.6]', '(37413.8, 40362.7]',
       '(40362.7, 42724.4]', '(42724.4, 45201]', '(34218.1, 37413.8]',
       '[22640, 34218.1]', '(61494.5, 125635]', '(45201, 48021.6]',
       '(54545.6, 61494.5]'], dtype=object)
```

In [ ]:

```
# Defining the map function
```

```
varlist = ['binnedInc']
def mmap(x):
    return x.map({'[22640, 34218.1]': 'BI1', "(34218.1, 37413.8]": 'BI2', '(37413.8, 40362.7]':'BI3',
                  '(40362.7, 42724.4]':'BI4', '(42724.4, 45201]':'BI5', '(45201, 48021.6]':'BI6',
                  '(48021.6, 51046.4]':'BI7', '(51046.4, 54545.6]':'BI8', '(54545.6, 61494.5]':'BI9',
                  '(61494.5, 125635]':'BI10' })

# Applying the function to the housing list
df_out2[varlist] = df_out2[varlist].apply(mmap)
```

In [ ]:

```
df_out2.binnedInc.unique()
```

Out[ ]:

```
array(['BI7', 'BI8', 'BI3', 'BI4', 'BI5', 'BI2', 'BI1', 'BI10', 'BI6',
       'BI9'], dtype=object)
```

In [ ]:

```
df_out2.Geography.value_counts()
```

Out[ ]:

```
Okfuskee County, Oklahoma         1
King William County, Virginia     1
Bailey County, Texas              1
Johnson County, Arkansas          1
Menominee County, Wisconsin       1
Bossier Parish, Louisiana         1
Warren County, Indiana            1
Shelby County, Illinois           1
Montgomery County, Indiana        1
Mineral County, Montana           1
Todd County, Kentucky             1
Charlevoix County, Michigan       1
Bowman County, North Dakota       1
Tipton County, Tennessee          1
Dent County, Missouri             1
Hamilton County, Illinois         1
Susquehanna County, Pennsylvania  1
Fulton County, Indiana            1
Tuscarawas County, Ohio           1
Dubois County, Indiana            1
McKean County, Pennsylvania       1
Palo Alto County, Iowa            1
Keokuk County, Iowa               1
Clinch County, Georgia            1
Carlisle County, Kentucky         1
Warren County, Virginia           1
Rock County, Nebraska             1
Kossuth County, Iowa              1
Lincoln County, Oregon            1
Cibola County, New Mexico         1
                                 ..
Menard County, Texas              1
Sevier County, Arkansas           1
Champaign County, Ohio            1
Garrett County, Maryland          1
Webster County, West Virginia     1
Greenbrier County, West Virginia  1
Wayne County, Nebraska            1
Pike County, Indiana              1
Lincoln County, Wisconsin         1
Potter County, Pennsylvania       1
Lewis County, Idaho               1
San Miguel County, Colorado       1
Ochiltree County, Texas           1
Union County, Kentucky            1
```

```
Union County, Kentucky             1
Hancock County, Iowa               1
Ionia County, Michigan             1
Aroostook County, Maine            1
New Madrid County, Missouri        1
Wise County, Texas                 1
Gregory County, South Dakota       1
Green County, Kentucky             1
Lake County, Montana               1
Red Willow County, Nebraska        1
Falls County, Texas                1
Polk County, Wisconsin             1
Caldwell County, Missouri          1
Livingston County, New York        1
Grant County, Kentucky             1
Cass County, Missouri              1
Patrick County, Virginia           1
Name: Geography, Length: 1348, dtype: int64
```

Since mostly geography is unique in every row we may drop this column

In [ ]:

```
df_out2 = df_out2.drop(['Geography'], axis = 1)
```

Dummy Variables

In [ ]:

```
status = pd.get_dummies(df_out2['binnedInc'], drop_first = True)
status.head()
```

Out[ ]:

	BI10	BI2	BI3	BI4	BI5	BI6	BI7	BI8	BI9
1	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0

In [ ]:

```
df_out2 = pd.concat([df_out2, status], axis = 1)
```

In [ ]:

```
df_out2.drop(['binnedInc'], axis = 1, inplace = True)
```

In [ ]:

```
df_out2.head()
```

Out[ ]:

	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	studyPerCa
1	173.0	70.0	161.30	411.60	48127.0	43269.0	18.6	23.11123
2	102.0	50.0	174.70	381.02	49348.0	21026.0	14.6	47.56016
4	57.0	26.0	146.11	381.02	49955.0	10321.0	12.5	0.00000
5	428.0	152.0	176.00	505.40	52313.0	61023.0	15.6	180.25990
6	250.0	97.0	175.90	461.80	37782.0	41516.0	23.2	0.00000

5 rows × 40 columns

Training Testing Split

In [ ]:

```
from sklearn.model_selection import train_test_split

# We specify this so that the train and test data set always have the same rows, respecti
vely
np.random.seed(0)
df_train, df_test = train_test_split(df_out2, train_size = 0.7, test_size = 0.3, random_
state = 100)
```

In [ ]:

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

In [ ]:

```
#will be scaling all continuous variables
var = df_train.columns.tolist()
var
```

Out[ ]:

```
['avgAnnCount',
 'avgDeathsPerYear',
 'TARGET_deathRate',
 'incidenceRate',
 'medIncome',
 'popEst2015',
 'povertyPercent',
 'studyPerCap',
 'MedianAge',
 'MedianAgeMale',
 'MedianAgeFemale',
 'AvgHouseholdSize',
 'PercentMarried',
 'PctNoHS18_24',
 'PctHS18_24',
 'PctBachDeg18_24',
 'PctHS25_Over',
 'PctBachDeg25_Over',
 'PctEmployed16_Over',
 'PctUnemployed16_Over',
 'PctPrivateCoverage',
 'PctPrivateCoverageAlone',
 'PctEmpPrivCoverage',
 'PctPublicCoverage',
 'PctPublicCoverageAlone',
 'PctWhite',
 'PctBlack',
 'PctAsian',
 'PctOtherRace',
 'PctMarriedHouseholds',
 'BirthRate',
 'BI10',
 'BI2',
 'BI3',
 'BI4',
 'BI5',
 'BI6',
 'BI7',
 'BI8',
 'BI9']
```

In [ ]:

In [ ]:

```
bin_vars = ['BI10', 'BI2', 'BI3', 'BI4', 'BI5', 'BI6', 'BI7', 'BI8', 'BI9']
```

In [ ]:

```
con_vars = set(var) - set(bin_vars)

con_vars = list(con_vars)

con_vars
```

Out[ ]:

```
['avgDeathsPerYear',
 'PctBachDeg18_24',
 'PctMarriedHouseholds',
 'avgAnnCount',
 'PctUnemployed16_Over',
 'PctAsian',
 'PctPublicCoverage',
 'PctPrivateCoverage',
 'popEst2015',
 'PctHS18_24',
 'povertyPercent',
 'PctPrivateCoverageAlone',
 'PctHS25_Over',
 'MedianAgeFemale',
 'PctWhite',
 'AvgHouseholdSize',
 'incidenceRate',
 'PctBachDeg25_Over',
 'studyPerCap',
 'MedianAgeMale',
 'PctEmpPrivCoverage',
 'TARGET_deathRate',
 'medIncome',
 'PctEmployed16_Over',
 'PctPublicCoverageAlone',
 'PctNoHS18_24',
 'PctBlack',
 'BirthRate',
 'MedianAge',
 'PctOtherRace',
 'PercentMarried']
```

In [ ]:

```
df_train[con_vars] = scaler.fit_transform(df_train[con_vars])
```

In [ ]:

```
df_train.head()
```

Out[ ]:

	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	studyPe
1159	0.086165	0.077922	0.000000	0.250296	0.538818	0.100935	0.116204	0.0
835	0.245146	0.256494	0.655735	0.709331	0.985201	0.321684	0.142027	0.1
1445	0.064320	0.084416	0.637928	0.424113	0.159163	0.069661	0.852163	0.0
2227	0.158981	0.181818	0.777415	0.917911	0.275975	0.133116	0.400258	0.0
2162	0.064320	0.064935	0.670574	1.000000	0.065175	0.061243	0.606843	0.0

5 rows × 40 columns

In [ ]:

```
df_train.describe()
```

Out[ ]:

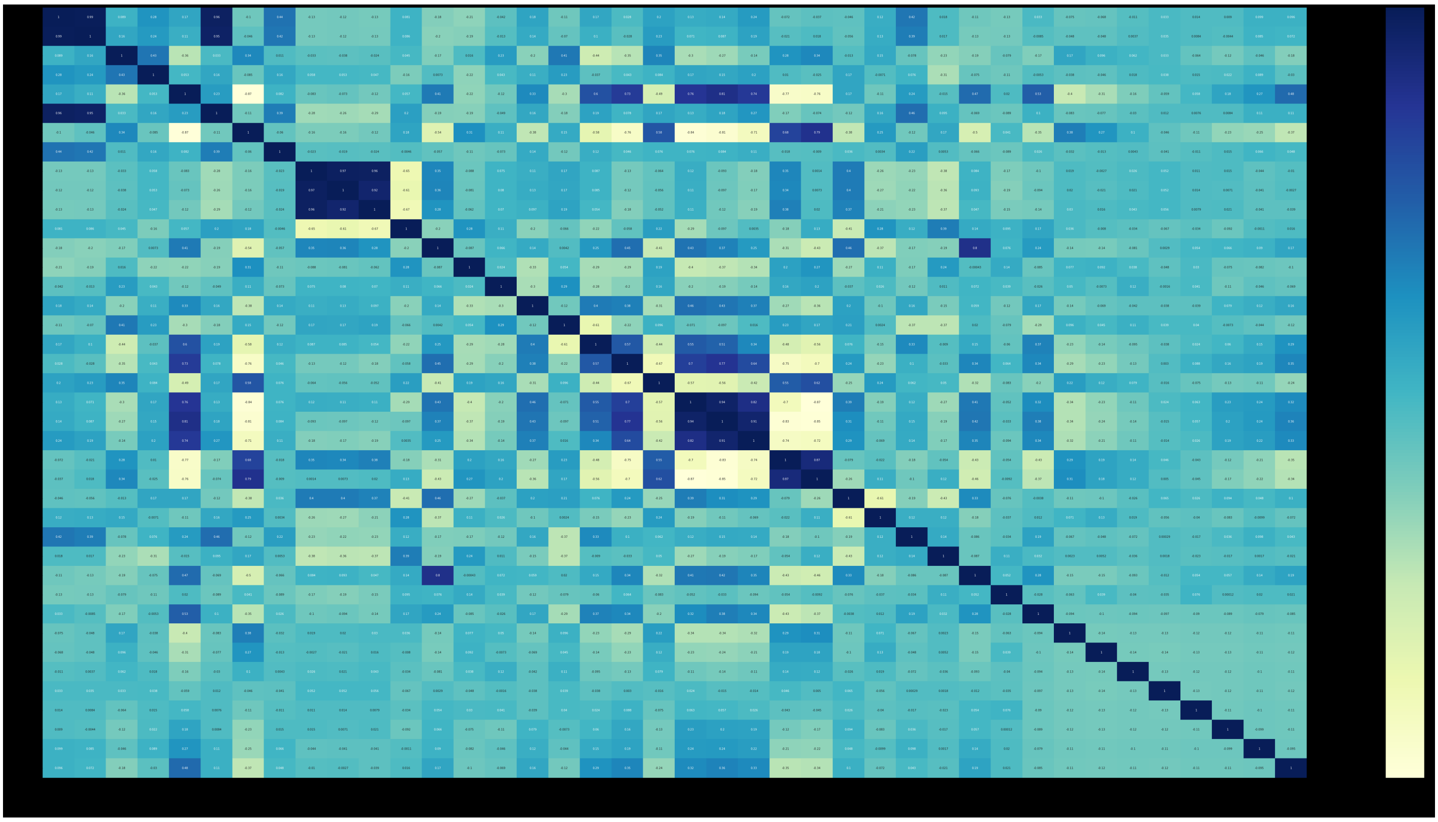
	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2015	povertyPercent	studyP
count	943.000000	943.000000	943.000000	943.000000	943.000000	943.000000	943.000000	943.0
mean	0.154023	0.174953	0.501790	0.501659	0.410995	0.166354	0.429230	0.0
std	0.179556	0.190067	0.324935	0.334278	0.297022	0.192196	0.305570	0.1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.023058	0.035714	0.229114	0.216323	0.159718	0.027540	0.167850	0.0
50%	0.092233	0.116883	0.505861	0.512602	0.365081	0.100505	0.387347	0.0
75%	0.218447	0.253247	0.768512	0.777277	0.615015	0.237275	0.658489	0.0
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0

8 rows × 40 columns

In [ ]:

```
# Let's check the correlation coefficients to see which variables are highly correlated

plt.figure(figsize = (100, 50))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



Model Building

In [ ]:

```
y_train = df_train.pop('TARGET_deathRate')
X_train = df_train
```

In [ ]:

```
#Build a linear model

import statsmodels.api as sm
```

```
X_train_lm = sm.add_constant(X_train)

lr_1 = sm.OLS(y_train, X_train_lm).fit()

lr_1.params
```

Out[ ]:

```
const                      0.477953
avgAnnCount               -5.065736
avgDeathsPerYear           4.747157
incidenceRate              0.468833
medIncome                  0.315416
popEst2015                -0.006729
povertyPercent            -0.073564
studyPerCap                0.069910
MedianAge                  0.034678
MedianAgeMale             -0.109428
MedianAgeFemale            0.000759
AvgHouseholdSize           0.016972
PercentMarried             0.162154
PctNoHS18_24              -0.022086
PctHS18_24                 0.033804
PctBachDeg18_24           -0.020034
PctHS25_Over               0.156264
PctBachDeg25_Over         -0.014574
PctEmployed16_Over        -0.112340
PctUnemployed16_Over       0.057618
PctPrivateCoverage        -0.291955
PctPrivateCoverageAlone   -0.034688
PctEmpPrivCoverage         0.123249
PctPublicCoverage         -0.166569
PctPublicCoverageAlone     0.001181
PctWhite                   0.017678
PctBlack                   0.088760
PctAsian                   0.063314
PctOtherRace              -0.111957
PctMarriedHouseholds      -0.128317
BirthRate                 -0.025339
BI10                      -0.407421
BI2                       -0.056350
BI3                       -0.119285
BI4                       -0.189957
BI5                       -0.181227
BI6                       -0.270431
BI7                       -0.319631
BI8                       -0.320093
BI9                       -0.393268
dtype: float64
```

In [ ]:

```
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      TARGET_deathRate   R-squared:                       0.620
Model:                           OLS   Adj. R-squared:                  0.604
Method:                Least Squares   F-statistic:                     37.81
Date:               Sat, 05 Sep 2020   Prob (F-statistic):          1.47e-161
Time:                       13:11:35   Log-Likelihood:                 179.00
No. Observations:                943   AIC:                            -278.0
Df Residuals:                    903   BIC:                            -84.03
Df Model:                         39
Covariance Type:           nonrobust
================================================================================
==
                             coef    std err          t      P>|t|      [0.025      0.9
75]
------------------------------------------------------------------------------
----
const                      0.4780      0.116      4.123      0.000       0.250       0.
705
```

```
705
avgAnnCount             -5.0657      0.340    -14.887      0.000      -5.734      -4.
398
avgDeathsPerYear         4.7472      0.266     17.858      0.000       4.225       5.
269
incidenceRate            0.4688      0.026     17.702      0.000       0.417       0.
521
medIncome                0.3154      0.206      1.532      0.126      -0.089       0.
720
popEst2015              -0.0067      0.172     -0.039      0.969      -0.344       0.
331
povertyPercent          -0.0736      0.071     -1.035      0.301      -0.213       0.
066
studyPerCap              0.0699      0.041      1.685      0.092      -0.012       0.
151
MedianAge                0.0347      0.142      0.244      0.807      -0.244       0.
314
MedianAgeMale           -0.1094      0.098     -1.122      0.262      -0.301       0.
082
MedianAgeFemale          0.0008      0.093      0.008      0.994      -0.182       0.
184
AvgHouseholdSize         0.0170      0.040      0.424      0.672      -0.062       0.
096
PercentMarried           0.1622      0.062      2.598      0.010       0.040       0.
285
PctNoHS18_24            -0.0221      0.026     -0.861      0.389      -0.072       0.
028
PctHS18_24               0.0338      0.025      1.354      0.176      -0.015       0.
083
PctBachDeg18_24         -0.0200      0.028     -0.711      0.477      -0.075       0.
035
PctHS25_Over             0.1563      0.037      4.168      0.000       0.083       0.
230
PctBachDeg25_Over       -0.0146      0.047     -0.311      0.756      -0.107       0.
077
PctEmployed16_Over      -0.1123      0.050     -2.255      0.024      -0.210      -0.
015
PctUnemployed16_Over     0.0576      0.034      1.673      0.095      -0.010       0.
125
PctPrivateCoverage      -0.2920      0.122     -2.395      0.017      -0.531      -0.
053
PctPrivateCoverageAlone -0.0347      0.145     -0.239      0.811      -0.319       0.
250
PctEmpPrivCoverage       0.1232      0.066      1.861      0.063      -0.007       0.
253
PctPublicCoverage       -0.1666      0.104     -1.600      0.110      -0.371       0.
038
PctPublicCoverageAlone   0.0012      0.092      0.013      0.990      -0.180       0.
183
PctWhite                 0.0177      0.045      0.392      0.695      -0.071       0.
106
PctBlack                 0.0888      0.042      2.134      0.033       0.007       0.
170
PctAsian                 0.0633      0.041      1.557      0.120      -0.016       0.
143
PctOtherRace            -0.1120      0.028     -4.031      0.000      -0.166      -0.
057
PctMarriedHouseholds    -0.1283      0.060     -2.131      0.033      -0.246      -0.
010
BirthRate               -0.0253      0.023     -1.121      0.263      -0.070       0.
019
BI10                    -0.4074      0.199     -2.050      0.041      -0.798      -0.
017
BI2                     -0.0563      0.034     -1.638      0.102      -0.124       0.
011
BI3                     -0.1193      0.046     -2.582      0.010      -0.210      -0.
029
BI4                     -0.1900      0.063     -3.023      0.003      -0.313      -0.
067
BI5                     -0.1812      0.078     -2.317      0.021      -0.335      -0.
028
BI6                     -0.2704      0.096     -2.810      0.005      -0.459      -0.
082
```

```
062
BI7                           -0.3196      0.117     -2.735      0.006      -0.549      -0.
090
BI8                           -0.3201      0.138     -2.312      0.021      -0.592      -0.
048
BI9                           -0.3933      0.171     -2.306      0.021      -0.728      -0.
059
==============================================================================
Omnibus:                       55.338   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               84.006
Skew:                           0.472   Prob(JB):                     5.73e-19
Kurtosis:                       4.117   Cond. No.                         178.
==============================================================================

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

In [ ]:

```python
# Importing RFE and LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
```

In [ ]:

```python
# Running RFE with the output number of the variable equal to 10
lm = LinearRegression()
lm.fit(X_train, y_train)

rfe = RFE(lm, 20)             # running RFE
rfe = rfe.fit(X_train, y_train)
```

In [ ]:

```python
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[ ]:

```
[('avgAnnCount', True, 1),
 ('avgDeathsPerYear', True, 1),
 ('incidenceRate', True, 1),
 ('medIncome', True, 1),
 ('popEst2015', False, 18),
 ('povertyPercent', True, 1),
 ('studyPerCap', False, 5),
 ('MedianAge', False, 11),
 ('MedianAgeMale', True, 1),
 ('MedianAgeFemale', False, 20),
 ('AvgHouseholdSize', False, 16),
 ('PercentMarried', True, 1),
 ('PctNoHS18_24', False, 13),
 ('PctHS18_24', False, 9),
 ('PctBachDeg18_24', False, 14),
 ('PctHS25_Over', True, 1),
 ('PctBachDeg25_Over', False, 17),
 ('PctEmployed16_Over', True, 1),
 ('PctUnemployed16_Over', False, 6),
 ('PctPrivateCoverage', True, 1),
 ('PctPrivateCoverageAlone', False, 10),
 ('PctEmpPrivCoverage', True, 1),
 ('PctPublicCoverage', True, 1),
 ('PctPublicCoverageAlone', False, 19),
 ('PctWhite', False, 15),
 ('PctBlack', True, 1),
 ('PctAsian', False, 7),
 ('PctOtherRace', True, 1),
 ('PctMarriedHouseholds', True, 1),
 ('BirthRate', False, 12),
 ('BI10', True, 1),
 ('BI2', False, 8),
 ('BI3', False, 4),
 ('BI4', True, 1)
```

```
 ('BI4', True, 1),
 ('BI5', False, 3),
 ('BI6', True, 1),
 ('BI7', True, 1),
 ('BI8', False, 2),
 ('BI9', True, 1)]
```

In [ ]:

```
col = X_train.columns[rfe.support_]
col
```

Out[ ]:

```
Index(['avgAnnCount', 'avgDeathsPerYear', 'incidenceRate', 'medIncome',
       'povertyPercent', 'MedianAgeMale', 'PercentMarried', 'PctHS25_Over',
       'PctEmployed16_Over', 'PctPrivateCoverage', 'PctEmpPrivCoverage',
       'PctPublicCoverage', 'PctBlack', 'PctOtherRace', 'PctMarriedHouseholds',
       'BI10', 'BI4', 'BI6', 'BI7', 'BI9'],
      dtype='object')
```

In [ ]:

```
X_train.columns[~rfe.support_]
```

Out[ ]:

```
Index(['popEst2015', 'studyPerCap', 'MedianAge', 'MedianAgeFemale',
       'AvgHouseholdSize', 'PctNoHS18_24', 'PctHS18_24', 'PctBachDeg18_24',
       'PctBachDeg25_Over', 'PctUnemployed16_Over', 'PctPrivateCoverageAlone',
       'PctPublicCoverageAlone', 'PctWhite', 'PctAsian', 'BirthRate', 'BI2',
       'BI3', 'BI5', 'BI8'],
      dtype='object')
```

In [ ]:

```
# Creating X_test dataframe with RFE selected variables
X_train_rfe = X_train[col]
```

In [ ]:

```
# Adding a constant variable
import statsmodels.api as sm
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      TARGET_deathRate   R-squared:                       0.611
Model:                           OLS   Adj. R-squared:                  0.602
Method:                Least Squares   F-statistic:                     72.32
Date:               Sat, 05 Sep 2020   Prob (F-statistic):          4.42e-173
Time:                       13:11:36   Log-Likelihood:                 167.33
No. Observations:                943   AIC:                            -292.7
Df Residuals:                    922   BIC:                            -190.8
Df Model:                         20
Covariance Type:           nonrobust
==========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
-
const                   0.4784      0.087      5.473      0.000       0.307       0.650
avgAnnCount            -5.0634      0.280    -18.085      0.000      -5.613      -4.514
avgDeathsPerYear        4.8071      0.259     18.549      0.000       4.299       5.316
incidenceRate           0.4825      0.024     19.799      0.000       0.435       0.530
medIncome              -0.0844      0.076     -1.110      0.267      -0.234       0.065
povertyPercent         -0.0713      0.067     -1.069      0.286      -0.202       0.060
MedianAgeMale          -0.0799      0.035     -2.252      0.025      -0.150      -0.010
PercentMarried          0.1494      0.058      2.596      0.010       0.036       0.262
PctHS25_Over            0.1565      0.029      5.332      0.000       0.099       0.214
PctEmployed16_Over     -0.1492      0.043     -3.432      0.001      -0.235      -0.064
PctPrivateCoverage     -0.3486      0.053     -6.570      0.000      -0.453      -0.244
```

```
PctEmpPrivCoverage          0.1362      0.049      2.772      0.006       0.040       0.233
PctPublicCoverage          -0.1375      0.049     -2.779      0.006      -0.235      -0.040
PctBlack                    0.0839      0.034      2.464      0.014       0.017       0.151
PctOtherRace               -0.1106      0.026     -4.280      0.000      -0.161      -0.060
PctMarriedHouseholds       -0.1109      0.053     -2.098      0.036      -0.215      -0.007
BI10                        0.0508      0.044      1.149      0.251      -0.036       0.137
BI4                        -0.0446      0.022     -2.005      0.045      -0.088      -0.001
BI6                        -0.0501      0.024     -2.049      0.041      -0.098      -0.002
BI7                        -0.0504      0.027     -1.888      0.059      -0.103       0.002
BI9                        -0.0085      0.036     -0.238      0.812      -0.078       0.061
==============================================================================
Omnibus:                       50.662   Durbin-Watson:                   2.059
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               76.963
Skew:                           0.440   Prob(JB):                     1.94e-17
Kurtosis:                       4.089   Cond. No.                         118.
==============================================================================

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

In [ ]:

```
lm = sm.OLS(y_train,X_train_rfe).fit()    # Running the linear model
```

In [ ]:

```
#Let's see the summary of our linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.611
Model:                            OLS   Adj. R-squared:                  0.602
Method:                 Least Squares   F-statistic:                     72.32
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          4.42e-173
Time:                        13:11:36   Log-Likelihood:                 167.33
No. Observations:                 943   AIC:                            -292.7
Df Residuals:                     922   BIC:                            -190.8
Df Model:                          20
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
-
const                       0.4784      0.087      5.473      0.000       0.307       0.650
avgAnnCount                -5.0634      0.280    -18.085      0.000      -5.613      -4.514
avgDeathsPerYear            4.8071      0.259     18.549      0.000       4.299       5.316
incidenceRate               0.4825      0.024     19.799      0.000       0.435       0.530
medIncome                  -0.0844      0.076     -1.110      0.267      -0.234       0.065
povertyPercent             -0.0713      0.067     -1.069      0.286      -0.202       0.060
MedianAgeMale              -0.0799      0.035     -2.252      0.025      -0.150      -0.010
PercentMarried              0.1494      0.058      2.596      0.010       0.036       0.262
PctHS25_Over                0.1565      0.029      5.332      0.000       0.099       0.214
PctEmployed16_Over         -0.1492      0.043     -3.432      0.001      -0.235      -0.064
PctPrivateCoverage         -0.3486      0.053     -6.570      0.000      -0.453      -0.244
PctEmpPrivCoverage          0.1362      0.049      2.772      0.006       0.040       0.233
PctPublicCoverage          -0.1375      0.049     -2.779      0.006      -0.235      -0.040
PctBlack                    0.0839      0.034      2.464      0.014       0.017       0.151
PctOtherRace               -0.1106      0.026     -4.280      0.000      -0.161      -0.060
PctMarriedHouseholds       -0.1109      0.053     -2.098      0.036      -0.215      -0.007
BI10                        0.0508      0.044      1.149      0.251      -0.036       0.137
BI4                        -0.0446      0.022     -2.005      0.045      -0.088      -0.001
BI6                        -0.0501      0.024     -2.049      0.041      -0.098      -0.002
BI7                        -0.0504      0.027     -1.888      0.059      -0.103       0.002
BI9                        -0.0085      0.036     -0.238      0.812      -0.078       0.061
==============================================================================
Omnibus:                       50.662   Durbin-Watson:                   2.059
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               76.963
Skew:                           0.440   Prob(JB):                     1.94e-17
Kurtosis:                       4.089   Cond. No.                         118.
==============================================================================
```

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

In [ ]:

```
X_train_rfe.columns
```

Out[ ]:

```
Index(['const', 'avgAnnCount', 'avgDeathsPerYear', 'incidenceRate',
       'medIncome', 'povertyPercent', 'MedianAgeMale', 'PercentMarried',
       'PctHS25_Over', 'PctEmployed16_Over', 'PctPrivateCoverage',
       'PctEmpPrivCoverage', 'PctPublicCoverage', 'PctBlack', 'PctOtherRace',
       'PctMarriedHouseholds', 'BI10', 'BI4', 'BI6', 'BI7', 'BI9'],
      dtype='object')
```

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
```

In [ ]:

```
# Calculate the VIFs for the new model
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
1	avgDeathsPerYear	100.03
0	avgAnnCount	97.98
6	PercentMarried	33.65
14	PctMarriedHouseholds	27.64
3	medIncome	24.51
9	PctPrivateCoverage	19.96
11	PctPublicCoverage	16.93
10	PctEmpPrivCoverage	16.04
4	povertyPercent	11.45
8	PctEmployed16_Over	11.12
5	MedianAgeMale	10.75
7	PctHS25_Over	8.57
2	incidenceRate	4.84
15	BI10	2.80
19	BI9	2.59
13	PctOtherRace	2.51
12	PctBlack	1.82
18	BI7	1.61
17	BI6	1.39
16	BI4	1.22

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["BI9"], axis = 1)
```

In [ ]:

```
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.611
Model:                            OLS   Adj. R-squared:                  0.603
Method:                 Least Squares   F-statistic:                     76.20
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          5.13e-174
Time:                        13:11:37   Log-Likelihood:                 167.30
No. Observations:                 943   AIC:                            -294.6
Df Residuals:                     923   BIC:                            -197.6
Df Model:                          19
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
-
const                      0.4818      0.086      5.586      0.000       0.313       0.651
avgAnnCount               -5.0666      0.280    -18.127      0.000      -5.615      -4.518
avgDeathsPerYear           4.8102      0.259     18.595      0.000       4.303       5.318
incidenceRate              0.4831      0.024     19.954      0.000       0.436       0.531
medIncome                 -0.0945      0.063     -1.499      0.134      -0.218       0.029
povertyPercent            -0.0746      0.065     -1.144      0.253      -0.203       0.053
MedianAgeMale             -0.0807      0.035     -2.284      0.023      -0.150      -0.011
PercentMarried             0.1484      0.057      2.587      0.010       0.036       0.261
PctHS25_Over               0.1565      0.029      5.333      0.000       0.099       0.214
PctEmployed16_Over        -0.1496      0.043     -3.444      0.001      -0.235      -0.064
PctPrivateCoverage        -0.3475      0.053     -6.577      0.000      -0.451      -0.244
PctEmpPrivCoverage         0.1357      0.049      2.765      0.006       0.039       0.232
PctPublicCoverage         -0.1370      0.049     -2.773      0.006      -0.234      -0.040
PctBlack                   0.0842      0.034      2.476      0.013       0.017       0.151
PctOtherRace              -0.1101      0.026     -4.277      0.000      -0.161      -0.060
PctMarriedHouseholds      -0.1099      0.053     -2.087      0.037      -0.213      -0.007
BI10                       0.0573      0.035      1.654      0.098      -0.011       0.125
BI4                       -0.0439      0.022     -1.992      0.047      -0.087      -0.001
BI6                       -0.0481      0.023     -2.096      0.036      -0.093      -0.003
BI7                       -0.0476      0.024     -1.982      0.048      -0.095      -0.000
==============================================================================
Omnibus:                       50.561   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               76.757
Skew:                           0.439   Prob(JB):                     2.15e-17
Kurtosis:                       4.087   Cond. No.                         118.
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["povertyPercent"], axis = 1)
```

In [ ]:

```
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.610
Model:                            OLS   Adj. R-squared:                  0.603
Method:                 Least Squares   F-statistic:                     80.34
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          1.08e-174
```

```
Time:                        13:11:37   Log-Likelihood:                 166.64
No. Observations:                 943   AIC:                            -295.3
Df Residuals:                     924   BIC:                            -203.1
Df Model:                          18
Covariance Type:            nonrobust
=========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
-
const                   0.4069      0.056      7.246      0.000       0.297       0.517
avgAnnCount            -5.0594      0.279    -18.103      0.000      -5.608      -4.511
avgDeathsPerYear        4.8053      0.259     18.575      0.000       4.298       5.313
incidenceRate           0.4817      0.024     19.919      0.000       0.434       0.529
medIncome              -0.0512      0.050     -1.015      0.310      -0.150       0.048
MedianAgeMale          -0.0695      0.034     -2.047      0.041      -0.136      -0.003
PercentMarried          0.1550      0.057      2.715      0.007       0.043       0.267
PctHS25_Over            0.1593      0.029      5.447      0.000       0.102       0.217
PctEmployed16_Over     -0.1371      0.042     -3.260      0.001      -0.220      -0.055
PctPrivateCoverage     -0.3302      0.051     -6.521      0.000      -0.430      -0.231
PctEmpPrivCoverage      0.1352      0.049      2.755      0.006       0.039       0.231
PctPublicCoverage      -0.1382      0.049     -2.797      0.005      -0.235      -0.041
PctBlack                0.0813      0.034      2.396      0.017       0.015       0.148
PctOtherRace           -0.1105      0.026     -4.295      0.000      -0.161      -0.060
PctMarriedHouseholds   -0.1102      0.053     -2.092      0.037      -0.214      -0.007
BI10                    0.0511      0.034      1.494      0.136      -0.016       0.118
BI4                    -0.0407      0.022     -1.862      0.063      -0.084       0.002
BI6                    -0.0451      0.023     -1.980      0.048      -0.090      -0.000
BI7                    -0.0449      0.024     -1.876      0.061      -0.092       0.002
==============================================================================
Omnibus:                       49.772   Durbin-Watson:                   2.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               76.217
Skew:                           0.431   Prob(JB):                     2.82e-17
Kurtosis:                       4.094   Cond. No.                         116.
==============================================================================

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

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(["medIncome"], axis = 1)
```

In [ ]:

```python
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.610
Model:                            OLS   Adj. R-squared:                  0.603
Method:                 Least Squares   F-statistic:                     85.00
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          1.91e-175
Time:                        13:11:37   Log-Likelihood:                 166.11
No. Observations:                 943   AIC:                            -296.2
Df Residuals:                     925   BIC:                            -208.9
Df Model:                          17
Covariance Type:            nonrobust
=========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
-
const                   0.3998      0.056      7.176      0.000       0.290       0.509
avgAnnCount            -5.0788      0.279    -18.215      0.000      -5.626      -4.532
avgDeathsPerYear        4.8177      0.258     18.644      0.000       4.311       5.325
incidenceRate           0.4815      0.024     19.912      0.000       0.434       0.529
MedianAgeMale          -0.0757      0.033     -2.267      0.024      -0.141      -0.010
PercentMarried          0.1620      0.057      2.858      0.004       0.051       0.273
PctHS25_Over            0.1662      0.028      5.849      0.000       0.110       0.222
```

```
PctEmployed16_Over    -0.1477      0.041     -3.623      0.000      -0.228      -0.068
PctPrivateCoverage    -0.3388      0.050     -6.785      0.000      -0.437      -0.241
PctEmpPrivCoverage     0.1256      0.048      2.608      0.009       0.031       0.220
PctPublicCoverage     -0.1293      0.049     -2.659      0.008      -0.225      -0.034
PctBlack               0.0810      0.034      2.386      0.017       0.014       0.148
PctOtherRace          -0.1127      0.026     -4.397      0.000      -0.163      -0.062
PctMarriedHouseholds  -0.1224      0.051     -2.387      0.017      -0.223      -0.022
BI10                   0.0396      0.032      1.227      0.220      -0.024       0.103
BI4                   -0.0400      0.022     -1.831      0.067      -0.083       0.003
BI6                   -0.0468      0.023     -2.060      0.040      -0.091      -0.002
BI7                   -0.0478      0.024     -2.011      0.045      -0.094      -0.001
==============================================================================
Omnibus:                       50.138   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               78.306
Skew:                           0.427   Prob(JB):                     9.91e-18
Kurtosis:                       4.124   Cond. No.                         113.
==============================================================================

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

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(["BI10"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.609
Model:                            OLS   Adj. R-squared:                  0.602
Method:                 Least Squares   F-statistic:                     90.17
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          4.17e-176
Time:                        13:11:37   Log-Likelihood:                 165.34
No. Observations:                 943   AIC:                            -296.7
Df Residuals:                     926   BIC:                            -214.3
Df Model:                          16
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
-
const                    0.4063      0.055      7.322      0.000       0.297       0.515
avgAnnCount             -5.0473      0.278    -18.174      0.000      -5.592      -4.502
avgDeathsPerYear         4.7859      0.257     18.610      0.000       4.281       5.291
incidenceRate            0.4818      0.024     19.919      0.000       0.434       0.529
MedianAgeMale           -0.0746      0.033     -2.234      0.026      -0.140      -0.009
PercentMarried           0.1657      0.057      2.928      0.003       0.055       0.277
PctHS25_Over             0.1581      0.028      5.719      0.000       0.104       0.212
PctEmployed16_Over      -0.1479      0.041     -3.628      0.000      -0.228      -0.068
PctPrivateCoverage      -0.3425      0.050     -6.871      0.000      -0.440      -0.245
PctEmpPrivCoverage       0.1338      0.048      2.804      0.005       0.040       0.227
PctPublicCoverage       -0.1359      0.048     -2.811      0.005      -0.231      -0.041
PctBlack                 0.0835      0.034      2.465      0.014       0.017       0.150
PctOtherRace            -0.1137      0.026     -4.435      0.000      -0.164      -0.063
PctMarriedHouseholds    -0.1197      0.051     -2.337      0.020      -0.220      -0.019
BI4                     -0.0413      0.022     -1.891      0.059      -0.084       0.002
BI6                     -0.0508      0.023     -2.258      0.024      -0.095      -0.007
BI7                     -0.0534      0.023     -2.290      0.022      -0.099      -0.008
==============================================================================
Omnibus:                       51.008   Durbin-Watson:                   2.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               78.689
Skew:                           0.437   Prob(JB):                     8.18e-18
Kurtosis:                       4.113   Cond. No.                         113.
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["BI4"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.608
Model:                            OLS   Adj. R-squared:                  0.601
Method:                 Least Squares   F-statistic:                     95.68
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          2.45e-176
Time:                        13:11:38   Log-Likelihood:                 163.53
No. Observations:                 943   AIC:                            -295.1
Df Residuals:                     927   BIC:                            -217.5
Df Model:                          15
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
-
const                    0.4024      0.056      7.248      0.000       0.293       0.511
avgAnnCount             -5.0348      0.278    -18.109      0.000      -5.580      -4.489
avgDeathsPerYear         4.7736      0.257     18.542      0.000       4.268       5.279
incidenceRate            0.4817      0.024     19.889      0.000       0.434       0.529
MedianAgeMale           -0.0730      0.033     -2.185      0.029      -0.139      -0.007
PercentMarried           0.1663      0.057      2.933      0.003       0.055       0.278
PctHS25_Over             0.1538      0.028      5.576      0.000       0.100       0.208
PctEmployed16_Over      -0.1475      0.041     -3.612      0.000      -0.228      -0.067
PctPrivateCoverage      -0.3457      0.050     -6.930      0.000      -0.444      -0.248
PctEmpPrivCoverage       0.1368      0.048      2.865      0.004       0.043       0.230
PctPublicCoverage       -0.1387      0.048     -2.868      0.004      -0.234      -0.044
PctBlack                 0.0841      0.034      2.479      0.013       0.018       0.151
PctOtherRace            -0.1133      0.026     -4.414      0.000      -0.164      -0.063
PctMarriedHouseholds    -0.1179      0.051     -2.298      0.022      -0.219      -0.017
BI6                     -0.0451      0.022     -2.020      0.044      -0.089      -0.001
BI7                     -0.0481      0.023     -2.076      0.038      -0.094      -0.003
==============================================================================
Omnibus:                       49.338   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               75.284
Skew:                           0.429   Prob(JB):                     4.49e-17
Kurtosis:                       4.086   Cond. No.                         112.
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
1	avgDeathsPerYear	97.18
0	avgAnnCount	95.14
4	PercentMarried	32.27
12	PctMarriedHouseholds	23.92
7	PctPrivateCoverage	18.67

	Features	VIF
8	PctEmpPrivCoverage	14.68
3	MedianAgeMale	10.07
6	PctEmployed16_Over	9.43
9	PctPublicCoverage	9.12
5	PctHS25_Over	7.64
2	incidenceRate	4.67
11	PctOtherRace	2.15
10	PctBlack	1.59
14	BI7	1.21
13	BI6	1.16

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["avgDeathsPerYear"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.462
Model:                            OLS   Adj. R-squared:                  0.454
Method:                 Least Squares   F-statistic:                     56.93
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          1.71e-114
Time:                        13:11:38   Log-Likelihood:                 14.791
No. Observations:                 943   AIC:                            0.4189
Df Residuals:                     928   BIC:                             73.15
Df Model:                          14
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
-
const                      0.5422      0.064      8.423      0.000       0.416       0.669
avgAnnCount                0.0604      0.049      1.220      0.223      -0.037       0.157
incidenceRate              0.3464      0.027     12.818      0.000       0.293       0.399
MedianAgeMale             -0.1262      0.039     -3.237      0.001      -0.203      -0.050
PercentMarried             0.2813      0.066      4.266      0.000       0.152       0.411
PctHS25_Over               0.2683      0.031      8.526      0.000       0.207       0.330
PctEmployed16_Over        -0.2429      0.047     -5.126      0.000      -0.336      -0.150
PctPrivateCoverage        -0.4301      0.058     -7.397      0.000      -0.544      -0.316
PctEmpPrivCoverage         0.1790      0.056      3.208      0.001       0.069       0.288
PctPublicCoverage         -0.1096      0.057     -1.937      0.053      -0.221       0.001
PctBlack                   0.0772      0.040      1.944      0.052      -0.001       0.155
PctOtherRace              -0.1461      0.030     -4.874      0.000      -0.205      -0.087
PctMarriedHouseholds      -0.2274      0.060     -3.813      0.000      -0.344      -0.110
BI6                       -0.0525      0.026     -2.009      0.045      -0.104      -0.001
BI7                       -0.0559      0.027     -2.063      0.039      -0.109      -0.003
==============================================================================
Omnibus:                        5.213   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.074   Jarque-Bera (JB):                5.358
Skew:                           0.130   Prob(JB):                       0.0686
Kurtosis:                       3.263   Cond. No.                         24.4
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
```

```
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
3	PercentMarried	31.96
11	PctMarriedHouseholds	23.78
6	PctPrivateCoverage	18.58
7	PctEmpPrivCoverage	14.60
2	MedianAgeMale	10.01
5	PctEmployed16_Over	9.39
8	PctPublicCoverage	8.75
4	PctHS25_Over	7.17
1	incidenceRate	4.29
0	avgAnnCount	2.24
10	PctOtherRace	2.15
9	PctBlack	1.59
13	BI7	1.21
12	BI6	1.16

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["PctEmpPrivCoverage"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        TARGET_deathRate   R-squared:                       0.456
Model:                             OLS   Adj. R-squared:                  0.448
Method:                  Least Squares   F-statistic:                     59.91
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          3.34e-113
Time:                         13:11:38   Log-Likelihood:                 9.5913
No. Observations:                  943   AIC:                             8.817
Df Residuals:                      929   BIC:                             76.70
Df Model:                           13
Covariance Type:             nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
-
const                    0.5832      0.063      9.196      0.000       0.459       0.708
avgAnnCount              0.0967      0.048      1.998      0.046       0.002       0.192
incidenceRate            0.3539      0.027     13.078      0.000       0.301       0.407
MedianAgeMale           -0.1484      0.039     -3.852      0.000      -0.224      -0.073
PercentMarried           0.2437      0.065      3.737      0.000       0.116       0.372
PctHS25_Over             0.2960      0.030      9.738      0.000       0.236       0.356
PctEmployed16_Over      -0.2323      0.048     -4.890      0.000      -0.326      -0.139
PctPrivateCoverage      -0.3223      0.048     -6.758      0.000      -0.416      -0.229
PctPublicCoverage       -0.1555      0.055     -2.826      0.005      -0.263      -0.048
PctBlack                 0.0688      0.040      1.728      0.084      -0.009       0.147
PctOtherRace            -0.1485      0.030     -4.932      0.000      -0.208      -0.089
PctMarriedHouseholds    -0.1988      0.059     -3.354      0.001      -0.315      -0.082
BI6                     -0.0587      0.026     -2.240      0.025      -0.110      -0.007
BI7                     -0.0540      0.027     -1.981      0.048      -0.107      -0.001
==============================================================================
Omnibus:                        3.373   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.185   Jarque-Bera (JB):                3.333
```

```
Skew:                           0.100   Prob(JB):                        0.189
Kurtosis:                       3.211   Cond. No.                         23.7
==============================================================================

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

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
3	PercentMarried	30.69
10	PctMarriedHouseholds	22.55
6	PctPrivateCoverage	11.25
2	MedianAgeMale	9.71
5	PctEmployed16_Over	9.03
7	PctPublicCoverage	8.53
4	PctHS25_Over	6.41
1	incidenceRate	4.24
9	PctOtherRace	2.14
0	avgAnnCount	2.11
8	PctBlack	1.59
12	BI7	1.21
11	BI6	1.15

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(["PctBlack"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()    # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        TARGET_deathRate   R-squared:                       0.454
Model:                             OLS   Adj. R-squared:                  0.447
Method:                  Least Squares   F-statistic:                     64.52
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          1.77e-113
Time:                         13:11:39   Log-Likelihood:                 8.0784
No. Observations:                  943   AIC:                             9.843
Df Residuals:                      930   BIC:                             72.88
Df Model:                           12
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
-
const                    0.6242      0.059     10.603      0.000       0.509       0.740
avgAnnCount              0.1023      0.048      2.116      0.035       0.007       0.197
incidenceRate            0.3550      0.027     13.106      0.000       0.302       0.408
MedianAgeMale           -0.1502      0.039     -3.895      0.000      -0.226      -0.075
```

```
PercentMarried           0.2255      0.064      3.500      0.000      0.099      0.352
PctHS25_Over             0.2973      0.030      9.771      0.000      0.238      0.357
PctEmployed16_Over      -0.2483      0.047     -5.321      0.000     -0.340     -0.157
PctPrivateCoverage      -0.3300      0.048     -6.943      0.000     -0.423     -0.237
PctPublicCoverage       -0.1777      0.054     -3.319      0.001     -0.283     -0.073
PctOtherRace            -0.1491      0.030     -4.949      0.000     -0.208     -0.090
PctMarriedHouseholds    -0.1942      0.059     -3.276      0.001     -0.310     -0.078
BI6                     -0.0593      0.026     -2.264      0.024     -0.111     -0.008
BI7                     -0.0558      0.027     -2.048      0.041     -0.109     -0.002
==============================================================================
Omnibus:                        3.650   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.161   Jarque-Bera (JB):                3.598
Skew:                           0.110   Prob(JB):                        0.165
Kurtosis:                       3.207   Cond. No.                         23.4
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
3	PercentMarried	29.11
9	PctMarriedHouseholds	21.61
6	PctPrivateCoverage	11.23
2	MedianAgeMale	9.71
5	PctEmployed16_Over	9.03
7	PctPublicCoverage	8.48
4	PctHS25_Over	6.30
1	incidenceRate	4.21
0	avgAnnCount	2.09
8	PctOtherRace	2.08
11	BI7	1.20
10	BI6	1.15

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["PercentMarried"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        TARGET_deathRate   R-squared:                       0.447
Model:                             OLS   Adj. R-squared:                  0.441
Method:                  Least Squares   F-statistic:                     68.44
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          8.46e-112
Time:                         13:11:39   Log-Likelihood:                 1.9070
No. Observations:                  943   AIC:                             20.19
Df Residuals:                      931   BIC:                             78.37
```

```
Df Model:                          11
Covariance Type:            nonrobust
===========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
-
const                   0.5821      0.058     10.041      0.000       0.468       0.696
avgAnnCount             0.0759      0.048      1.580      0.114      -0.018       0.170
incidenceRate           0.3680      0.027     13.636      0.000       0.315       0.421
MedianAgeMale          -0.0910      0.035     -2.609      0.009      -0.159      -0.023
PctHS25_Over            0.2883      0.030      9.453      0.000       0.228       0.348
PctEmployed16_Over     -0.1690      0.041     -4.118      0.000      -0.250      -0.088
PctPrivateCoverage     -0.3475      0.048     -7.308      0.000      -0.441      -0.254
PctPublicCoverage      -0.1462      0.053     -2.753      0.006      -0.250      -0.042
PctOtherRace           -0.1526      0.030     -5.037      0.000      -0.212      -0.093
PctMarriedHouseholds   -0.0242      0.034     -0.707      0.480      -0.091       0.043
BI6                    -0.0619      0.026     -2.349      0.019      -0.114      -0.010
BI7                    -0.0570      0.027     -2.080      0.038      -0.111      -0.003
==============================================================================
Omnibus:                        4.437   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.109   Jarque-Bera (JB):                4.427
Skew:                           0.126   Prob(JB):                        0.109
Kurtosis:                       3.223   Cond. No.                         18.7
==============================================================================

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

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
5	PctPrivateCoverage	10.85
6	PctPublicCoverage	8.47
2	MedianAgeMale	7.77
4	PctEmployed16_Over	7.19
8	PctMarriedHouseholds	7.01
3	PctHS25_Over	6.15
1	incidenceRate	4.16
7	PctOtherRace	2.04
0	avgAnnCount	2.02
10	BI7	1.20
9	BI6	1.15

In [ ]:

```python
X_train_rfe = X_train_rfe.drop(["PctPrivateCoverage"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:        TARGET_deathRate   R-squared:                       0.415
Model:                             OLS   Adj. R-squared:                  0.409
Method:                  Least Squares   F-statistic:                     66.22
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          1.38e-101
Time:                         13:11:39   Log-Likelihood:                -24.397
No. Observations:                  943   AIC:                             70.79
Df Residuals:                      932   BIC:                             124.1
Df Model:                           10
Covariance Type:             nonrobust
==========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
-
const                    0.4308      0.056      7.741      0.000       0.322       0.540
avgAnnCount              0.0148      0.049      0.304      0.761      -0.081       0.110
incidenceRate            0.3426      0.028     12.457      0.000       0.289       0.397
MedianAgeMale           -0.1943      0.033     -5.934      0.000      -0.259      -0.130
PctHS25_Over             0.2869      0.031      9.155      0.000       0.225       0.348
PctEmployed16_Over      -0.2631      0.040     -6.567      0.000      -0.342      -0.184
PctPublicCoverage        0.0541      0.047      1.158      0.247      -0.038       0.146
PctOtherRace            -0.0982      0.030     -3.252      0.001      -0.157      -0.039
PctMarriedHouseholds    -0.0450      0.035     -1.286      0.199      -0.114       0.024
BI6                     -0.0669      0.027     -2.472      0.014      -0.120      -0.014
BI7                     -0.0955      0.028     -3.455      0.001      -0.150      -0.041
==============================================================================
Omnibus:                         2.212   Durbin-Watson:                   1.983
Prob(Omnibus):                   0.331   Jarque-Bera (JB):                2.236
Skew:                            0.118   Prob(JB):                        0.327
Kurtosis:                        2.960   Cond. No.                         16.3
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
5	PctPublicCoverage	7.12
2	MedianAgeMale	6.53
7	PctMarriedHouseholds	6.50
3	PctHS25_Over	6.05
4	PctEmployed16_Over	4.83
1	incidenceRate	4.01
6	PctOtherRace	2.02
0	avgAnnCount	1.91
9	BI7	1.16
8	BI6	1.15

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["PctPublicCoverage"], axis = 1)
```

```
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        TARGET_deathRate   R-squared:                       0.415
Model:                             OLS   Adj. R-squared:                  0.409
Method:                  Least Squares   F-statistic:                     73.41
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          3.03e-102
Time:                         13:11:39   Log-Likelihood:                -25.075
No. Observations:                  943   AIC:                             70.15
Df Residuals:                      933   BIC:                             118.6
Df Model:                            9
Covariance Type:             nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
-
const                     0.4739      0.041     11.444      0.000       0.393       0.555
avgAnnCount               0.0105      0.048      0.217      0.828      -0.085       0.106
incidenceRate             0.3433      0.028     12.480      0.000       0.289       0.397
MedianAgeMale            -0.1785      0.030     -5.996      0.000      -0.237      -0.120
PctHS25_Over              0.2902      0.031      9.294      0.000       0.229       0.351
PctEmployed16_Over       -0.2950      0.029    -10.161      0.000      -0.352      -0.238
PctOtherRace             -0.0965      0.030     -3.198      0.001      -0.156      -0.037
PctMarriedHouseholds     -0.0600      0.033     -1.844      0.066      -0.124       0.004
BI6                      -0.0659      0.027     -2.436      0.015      -0.119      -0.013
BI7                      -0.0962      0.028     -3.479      0.001      -0.150      -0.042
==============================================================================
Omnibus:                         2.163   Durbin-Watson:                   1.980
Prob(Omnibus):                   0.339   Jarque-Bera (JB):                2.187
Skew:                            0.116   Prob(JB):                        0.335
Kurtosis:                        2.959   Cond. No.                         11.4
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
6	PctMarriedHouseholds	6.46
3	PctHS25_Over	4.84
2	MedianAgeMale	4.06
1	incidenceRate	3.86
4	PctEmployed16_Over	3.82
0	avgAnnCount	1.89
5	PctOtherRace	1.55
8	BI7	1.16
7	BI6	1.15

In [ ]:

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["PctMarriedHouseholds"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()    # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        TARGET_deathRate   R-squared:                       0.412
Model:                             OLS   Adj. R-squared:                  0.407
Method:                  Least Squares   F-statistic:                     81.95
Date:                 Sat, 05 Sep 2020   Prob (F-statistic):          1.74e-102
Time:                         13:11:40   Log-Likelihood:                -26.790
No. Observations:                  943   AIC:                             71.58
Df Residuals:                      934   BIC:                             115.2
Df Model:                            8
Covariance Type:             nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                    0.4488      0.039     11.461      0.000       0.372       0.526
avgAnnCount              0.0164      0.048      0.339      0.735      -0.079       0.111
incidenceRate            0.3487      0.027     12.734      0.000       0.295       0.402
MedianAgeMale           -0.1846      0.030     -6.230      0.000      -0.243      -0.126
PctHS25_Over             0.2853      0.031      9.158      0.000       0.224       0.346
PctEmployed16_Over      -0.3145      0.027    -11.608      0.000      -0.368      -0.261
PctOtherRace            -0.0947      0.030     -3.138      0.002      -0.154      -0.035
BI6                     -0.0669      0.027     -2.468      0.014      -0.120      -0.014
BI7                     -0.0962      0.028     -3.477      0.001      -0.151      -0.042
==============================================================================
Omnibus:                        2.180   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.336   Jarque-Bera (JB):                2.218
Skew:                           0.116   Prob(JB):                        0.330
Kurtosis:                       2.949   Cond. No.                         10.4
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
3	PctHS25_Over	4.33
1	incidenceRate	3.86
2	MedianAgeMale	3.61
4	PctEmployed16_Over	2.75
0	avgAnnCount	1.89
5	PctOtherRace	1.47
7	BI7	1.16
6	BI6	1.15

In [ ]:

```
X_train_rfe = X_train_rfe.drop(["avgAnnCount"], axis = 1)
X_train_rfe = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe).fit()    # Running the linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       TARGET_deathRate   R-squared:                       0.412
Model:                            OLS   Adj. R-squared:                  0.408
Method:                 Least Squares   F-statistic:                     93.72
Date:                Sat, 05 Sep 2020   Prob (F-statistic):          1.83e-103
Time:                        13:11:40   Log-Likelihood:                -26.847
No. Observations:                 943   AIC:                             69.69
Df Residuals:                     935   BIC:                             108.5
Df Model:                           7
Covariance Type:            nonrobust
======================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
const                  0.4516      0.038     11.789      0.000       0.376       0.527
incidenceRate          0.3516      0.026     13.527      0.000       0.301       0.403
MedianAgeMale         -0.1856      0.029     -6.302      0.000      -0.243      -0.128
PctHS25_Over           0.2836      0.031      9.223      0.000       0.223       0.344
PctEmployed16_Over    -0.3148      0.027    -11.635      0.000      -0.368      -0.262
PctOtherRace          -0.0945      0.030     -3.133      0.002      -0.154      -0.035
BI6                   -0.0667      0.027     -2.462      0.014      -0.120      -0.014
BI7                   -0.0961      0.028     -3.475      0.001      -0.150      -0.042
==============================================================================
Omnibus:                        1.907   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.385   Jarque-Bera (JB):                1.950
Skew:                           0.108   Prob(JB):                        0.377
Kurtosis:                       2.945   Cond. No.                         9.87
==============================================================================

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

In [ ]:

```
X_train_rfe = X_train_rfe.drop(['const'], axis=1)
# Calculate the VIFs for the new model
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[ ]:

	Features	VIF
2	PctHS25_Over	4.32
1	MedianAgeMale	3.61
0	incidenceRate	3.29
3	PctEmployed16_Over	2.73
4	PctOtherRace	1.41
6	BI7	1.16
5	BI6	1.15

In [ ]: