Data Analytics Bootcamp

Case Western Reserve University

Final Project Presentation

Raksha Karthikeyan

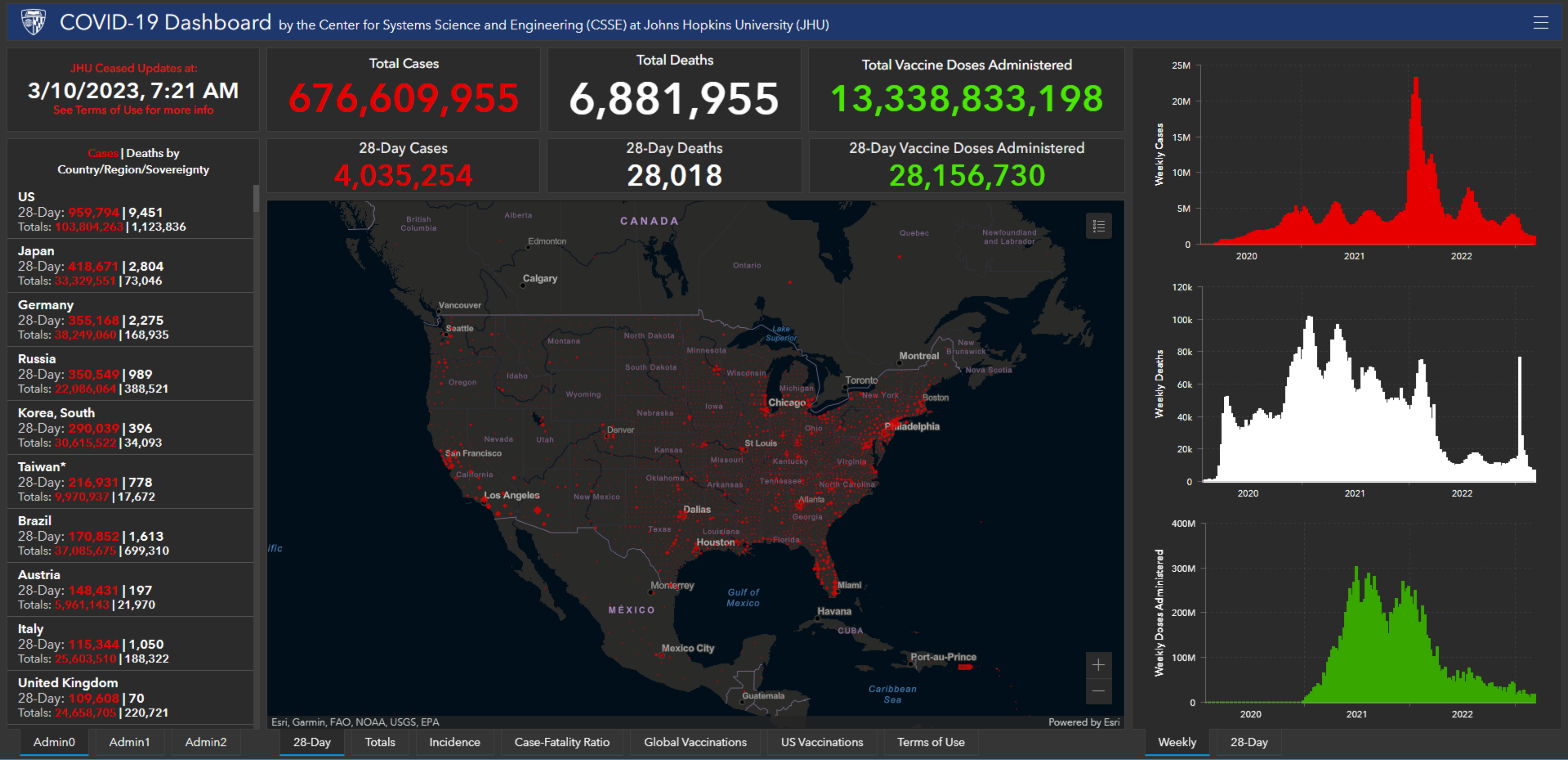
Alexander Melamed

Erickson Vo

Stephanie Garrett

Research Topic

Research Data



Johns Hopkins Center for Systems Science and Engineering (CSSE)

2019 Community Resilience Estimates for Equity and Disasters

Select State: Select one state

Select County: Select one or more counties

Select Risk Factor: 0 RF | 1-2 RF | 3+ RF

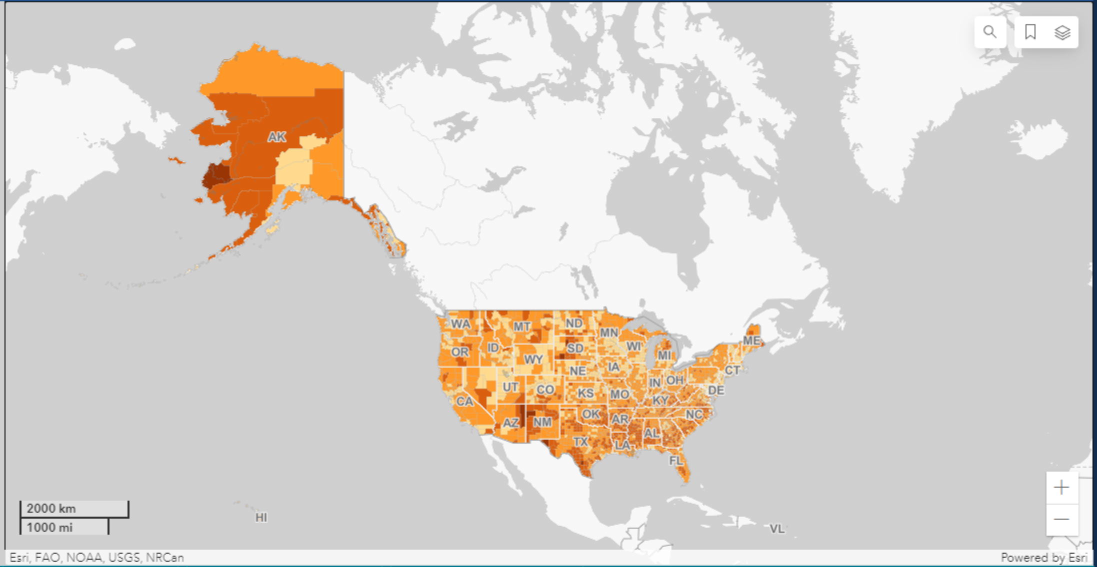
Thematic Risk Factor (RF) based on the risk factor selected above

Use Layer List below to turn on and off supplemental layers

Thematic Risk Factor (Counties) 2019

Estimated Population (%)

- > 50 - 78
- > 30 - 50
- > 20 - 30
- > 10 - 20
- 0 - 10



Thematic Risk map | Predominant Risk map | Statistical Difference Map | COVID-19 Impact Report

Community Resilience Estimates

34.6%	43.9%	21.6%
Est. Pop with 0 Risk Factors	Est. Pop with 1-2 Risk Factors	Est. Pop with 3+ Risk Factors

Showing Statistics for: **United States** for 3,142 County(s): Autauga County, Alabama; Baldwin County, Alabama

Population

Households Below the Poverty Level
12.9%
15,610,142 Total

Households Without Vehicle
8.6%
10,395,713 Total

Households w/Pop 65+ Living Alone
11%
13,259,766 Total

Households with Disability
25.5%
30,781,341 Total

Female Householder no spouse*
5.3%
6,453,219 Total

Male Householder no spouse*
1.3%
1,536,353 Total

Households with Broadband Internet
82.7%
99,824,789 Total

*Householder, no spouse present, with own children of the householder under 18 years

Key Facts

U.S. Census Bureau Community Resilience Estimates (CRE)

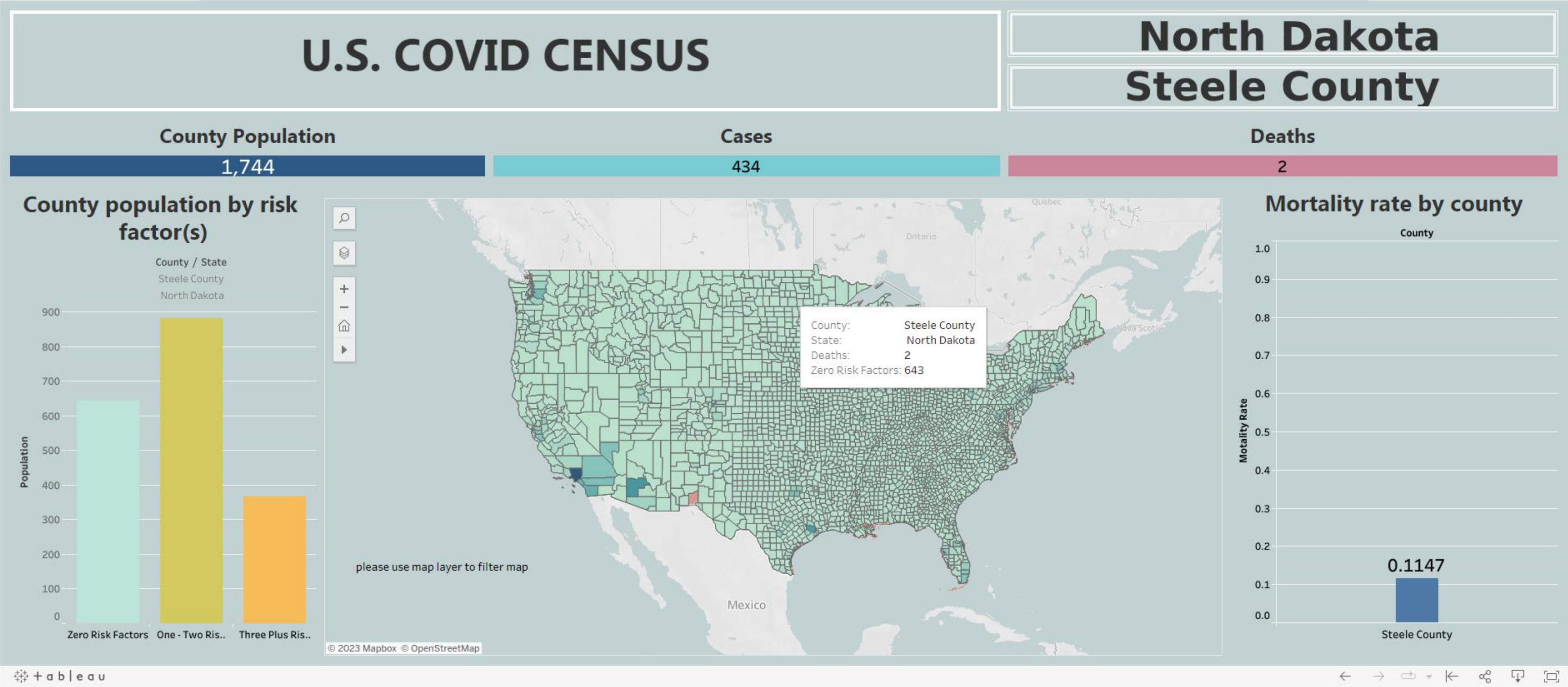
Research Questions

Data Exploration

COMBINED DATASET

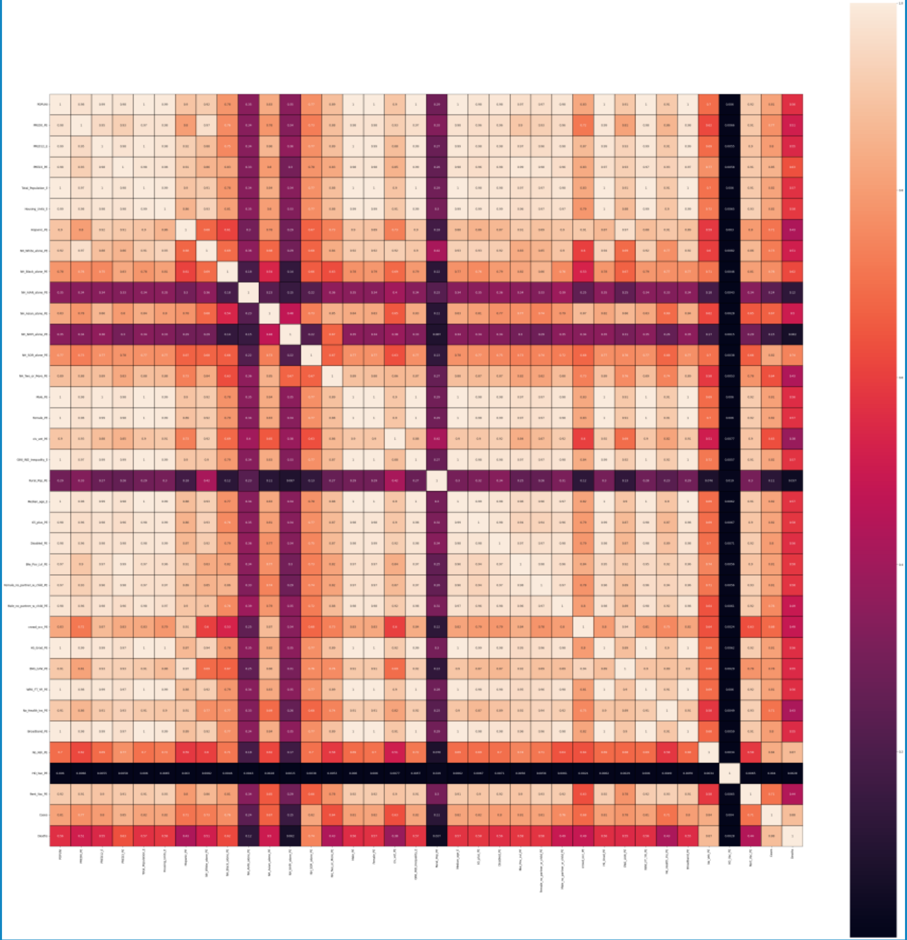
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	
1	fips	cases	deaths	lat	long	state	county	popuni	total_pop	zero_rf	one_two_	three_rf	housing_u	hispanic_	white_po	black_pop	native_po	asian_pop	pacific_isl	other_rac	bi_tri_rac	male_pop	female_p	veteran	gini_ind_i	rural_pop	median_a	elder_pop	disability_	be
2	1001	19732	230	32.53953	-86.6441	Alabama	Autauga C	55688	55380	20573	22750	12365	23493	1559	41543	10580	167	556	0	111	1169	27064	28623	7016	252	23388	21272	8353	10580	
3	1003	69641	724	30.72775	-87.7221	Alabama	Baldwin C	221898	212830	78622	90552	52724	114164	10207	184397	20414	1553	1997	0	443	3328	107842	114055	26183	1017	93818	95416	44379	31509	
4	1005	7451	103	31.86826	-85.3871	Alabama	Barbour C	22023	25361	5024	9171	7828	12013	969	10086	10438	66	110	0	88	264	11650	10372	1453	107	14929	8897	4096	4712	
5	1007	8067	109	32.99642	-87.1251	Alabama	Bibb Coun	20393	22493	6280	8986	5127	9185	530	15192	4506	20	20	0	0	101	11012	9380	1631	91	13938	8340	3242	3507	
6	1009	18616	261	33.98211	-86.5679	Alabama	Blount Co	57697	57681	18189	23950	15558	24323	5365	50138	865	57	230	0	230	865	28502	29194	4442	263	51898	23482	10327	8135	
7	1011	3020	54	32.10031	-85.7127	Alabama	Bullock Co	8600	10248	2074	3534	2992	4557	223	1840	6432	0	43	0	0	68	4678	3921	283	40	4417	3457	1376	1032	
8	1013	6518	132	31.753	-86.6806	Alabama	Butler Cou	19410	19828	5431	7643	6336	10089	271	10015	8734	19	58	0	0	310	8889	10520	1281	89	13825	7919	3823	3202	
9	1015	41228	675	33.77484	-85.8263	Alabama	Calhoun C	111694	114618	35169	44189	32336	53631	4244	80531	23120	223	1005	0	223	2457	53724	57969	12062	517	37640	44230	19211	23679	
10	1017	10812	170	32.9136	-85.3907	Alabama	Chambers	33117	33660	9457	12798	10862	16988	794	18379	13114	99	364	0	33	331	15829	17287	2616	140	16277	13909	6358	5994	
11	1019	6732	89	34.17806	-85.6064	Alabama	Cherokee	26116	25903	7374	11044	7698	16579	417	23948	1201	287	52	0	0	208	12875	13240	2376	117	22391	12143	5849	4648	
12	1021	12956	217	32.85044	-86.7173	Alabama	Chilton Co	44257	44055	12461	20344	11452	19781	3452	35449	4071	44	177	0	44	973	21553	22703	3363	208	38388	17127	7081	8585	
13	1023	2255	39	32.02227	-88.2656	Alabama	Choctaw C	12553	12925	2830	5421	4302	7359	50	7079	5347	12	12	0	12	50	5962	6590	1079	58	12553	5812	2836	3351	
14	1025	8529	107	31.681	-87.8355	Alabama	Clarke Cou	23522	24128	5797	10278	7447	12784	70	12396	10773	23	94	0	0	141	11125	12396	1458	121	17872	9996	4563	4281	
15	1027	5134	92	33.26984	-85.8584	Alabama	Clay Coun	13159	13337	3217	5728	4214	6799	407	10579	2039	0	26	0	13	92	6355	6803	789	60	13159	5697	2658	2289	
16	1029	4393	71	33.67679	-85.5201	Alabama	Cleburne	14819	14916	5228	6006	3585	6844	370	13737	400	14	0	0	14	281	7261	7557	903	67	14819	6386	2874	3245	
17	1031	17094	245	31.39933	-85.989	Alabama	Coffee Co	52203	51662	17942	21267	12994	23188	3810	36594	8770	626	730	0	52	1618	25736	26466	8561	225	24639	20515	8613	9292	
18	1033	21197	276	34.69847	-87.8017	Alabama	Colbert Co	55083	54771	18099	22141	14843	26588	1542	43074	8868	330	275	55	55	936	26439	28643	4902	246	24175	23355	10686	10190	
19	1035	3589	76	31.43402	-86.9932	Alabama	Conecuh C	12022	12394	3630	4119	4273	7182	84	6011	5746	24	60	0	0	96	5890	6131	709	51	9731	5385	2668	2692	
20	1037	3755	64	32.9369	-86.2485	Alabama	Coosa Cou	10511	10757	2829	4131	3551	6585	105	6811	3531	10	0	0	52	0	5339	5171	1093	47	10511	5087	2375	2407	
21	1039	11765	258	31.24779	-86.4505	Alabama	Covington	36877	37200	10089	15065	11723	18976	626	30718	4941	36	147	36	0	405	17811	19065	3687	169	25684	15967	7670	7781	
22	1041	4730	110	31.72942	-86.3159	Alabama	Crenshaw	13772	13844	3542	5703	4527	6815	110	9778	3181	96	179	13	0	426	6706	7065	908	67	13772	5742	2589	2823	
23	1043	31438	395	34.1302	-86.8689	Alabama	Cullman C	83277	82853	25997	36832	20448	37842	3664	76531	999	249	333	0	83	1498	40972	42304	6662	367	60992	33977	15156	14740	
24	1045	16436	245	31.43037	-85.611	Alabama	Dale Coun	49090	49277	14114	21082	13894	23103	3190	33675	9719	294	589	0	147	1472	24201	24888	8050	219	24981	18310	8050	10407	
25	1047	10935	258	32.32688	-87.1087	Alabama	Dallas Cou	36952	39149	9377	14920	12655	20419	406	10087	26014	36	221	0	0	184	17034	19917	2254	173	16864	14632	6429	6503	
26	1049	22466	345	34.45947	-85.8078	Alabama	DeKalb Co	71306	71310	22907	28714	19685	31309	10410	57330	998	784	71	142	0	1497	35296	36009	4492	336	64268	28165	11908	9626	
27	1051	29745	361	32.59785	-86.1442	Alabama	Elmore Co	77512	81144	29122	30362	18028	34165	2325	56738	16432	232	387	0	155	1240	37438	40073	8216	342	42003	29997	11549	13254	
28	1053	12347	179	31.12568	-87.1592	Alabama	Escambia	34309	37057	9184	14358	10767	16586	789	20654	10978	1372	102	34	34	308	17463	16845	2538	165	21789	13723	6072	6175	
29	1055	34137	690	34.04567	-86.0405	Alabama	Etowah Co	101051	102748	31759	41321	27971	47704	3940	78718	15561	404	707	0	101	1616	48706	52344	8488	464	37873	41835	18896	18593	
30	1057	5871	99	33.72077	-87.7389	Alabama	Fayette Co	16197	16494	4306	6815	5076	8505	291	13686	1814	0	80	0	0	323	7985	8211	1360	74	12994	7078	3368	4227	
31	1059	11996	152	34.44235	-87.8429	Alabama	Franklin C	31274	31466	8663	13728	8883	14052	5379	24018	1344	93	125	0	0	312	15574	15699	1751	133	22007	12071	5285	4691	
32	1061	7907	171	31.09389	-85.8357	Alabama	Geneva Co	26218	26417	8808	9731	7679	12869	1048	21996	2464	235	104	0	0	367	12741	13476	3041	115	23501	11326	5243	6187	
33	1063	2292	54	32.85504	-87.9568	Alabama	Greene Co	8111	8324	1533	3266	3312	5112	137	1395	6448	32	0	0	0	89	3747	4363	575	41	8111	3317	1727	2003	
34	1065	5705	110	32.76039	-87.6328	Alabama	Hale Coun	14597	14809	3431	6341	4825	7792	0	5765	8670	29	29	0	29	58	6933	7663	1036	72	13016	5911	2744	3094	
35	1067	5837	79	31.51148	-85.2427	Alabama	Henry Cou	17135	17133	4737	6961	5437	9155	445	11857	4557	34	68	17	0	171	8327	8807	1576	75	15035	7607	3786	3272	
36	1069	32514	529	31.15198	-85.2994	Alabama	Houston C	105267	104702	34095	42801	28371	47457	3473	70002	28211	315	947	0	315	2105	50422	54844	10316	508	35580	42212	18105	18105	
37	1071	18311	254	34.78144	-85.9975	Alabama	Jackson Co	51420	51852	15272	23252	12896	25041	1491	45866	1748	565	205	51	51	1336	25144	26275	3496	232	39603	22162	10026	9924	
38	1073	237792	2526	33.55555	-86.8951	Alabama	Jefferson	648693	659680	223785	259699	165209	307874	25299	323697	277640	1297	10379	0	1297	9730	306831	341861	46705	3249	63766	244557	99898	99898	

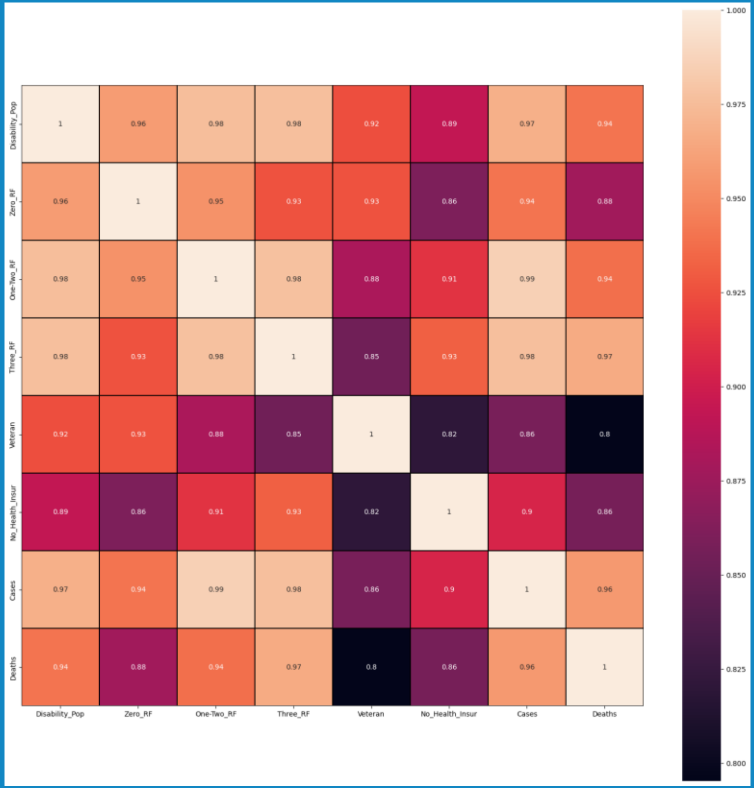
https://public.tableau.com/app/profile/erickson.vo/viz/Filterd_map_updated/Dashboard1?publish=yes



Interactive U.S. County Map

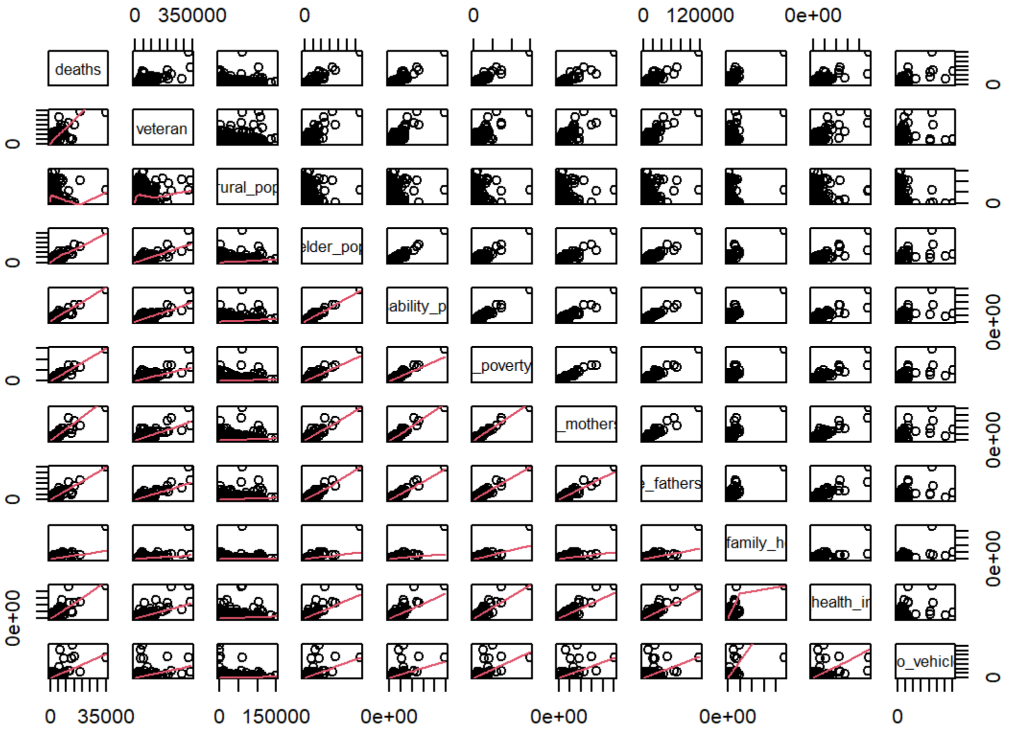
CRE VARIABLES HEATMAP





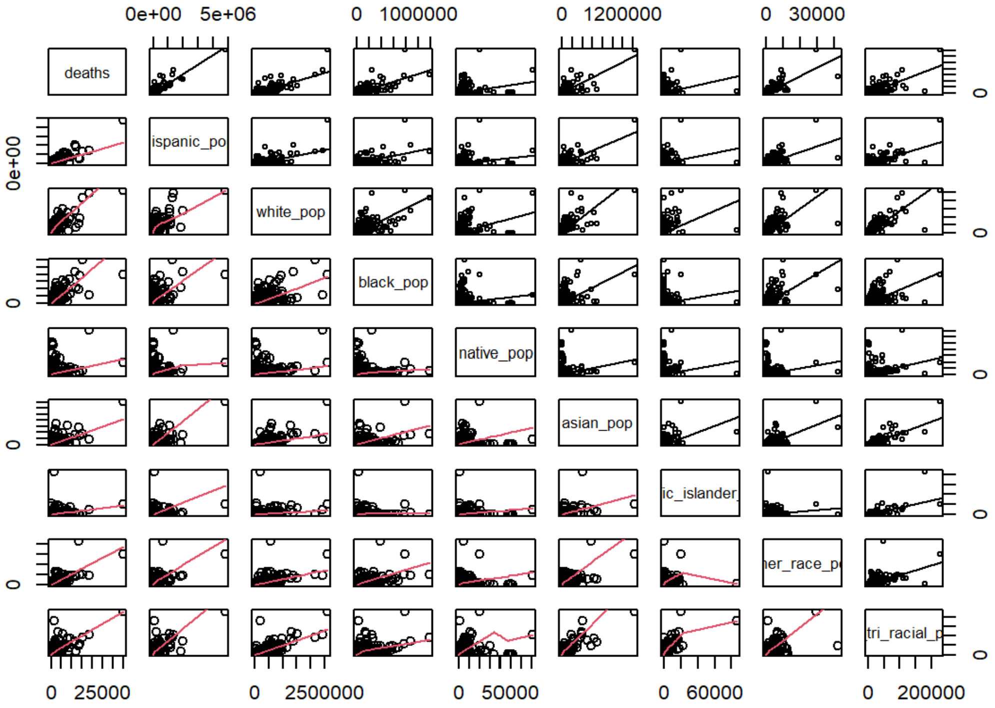
Vulnerable Populations

```
Call:
lm(formula = deaths ~ veteran + rural_pop + elder_pop + disability_pop +
    below_poverty_level + single_mothers_pop + single_fathers_pop +
    plus_family_homes + no_health_insur + no_vehicle, data = JHU_Cases_Deaths_with_CRE_Dat
Vtwo)

Residuals:
    Min      1Q  Median      3Q     Max
-2781.3   -16.8     2.8    25.1  5192.8

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         -2.2988148  6.4688549  -0.355 0.722339
veteran             -0.0103890  0.0010218 -10.167  < 2e-16 ***
rural_pop            0.0005486  0.0002924   1.877 0.060669 .
elder_pop            0.0084096  0.0006298  13.352  < 2e-16 ***
disability_pop       0.0139794  0.0016721   8.360  < 2e-16 ***
below_poverty_level  0.0033555  0.0008780   3.822 0.000135 ***
single_mothers_pop  -0.0015242  0.0019049  -0.800 0.423694
single_fathers_pop   0.0101712  0.0071879   1.415 0.157159
plus_family_homes    0.0065462  0.0011562   5.662 1.63e-08 ***
no_health_insur      0.0024688  0.0004402   5.608 2.22e-08 ***
no_vehicle           0.0024435  0.0001642  14.882  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 243.7 on 3131 degrees of freedom
Multiple R-squared:  0.9532,    Adjusted R-squared:  0.953
F-statistic:  6372 on 10 and 3131 DF,  p-value: < 2.2e-16
```



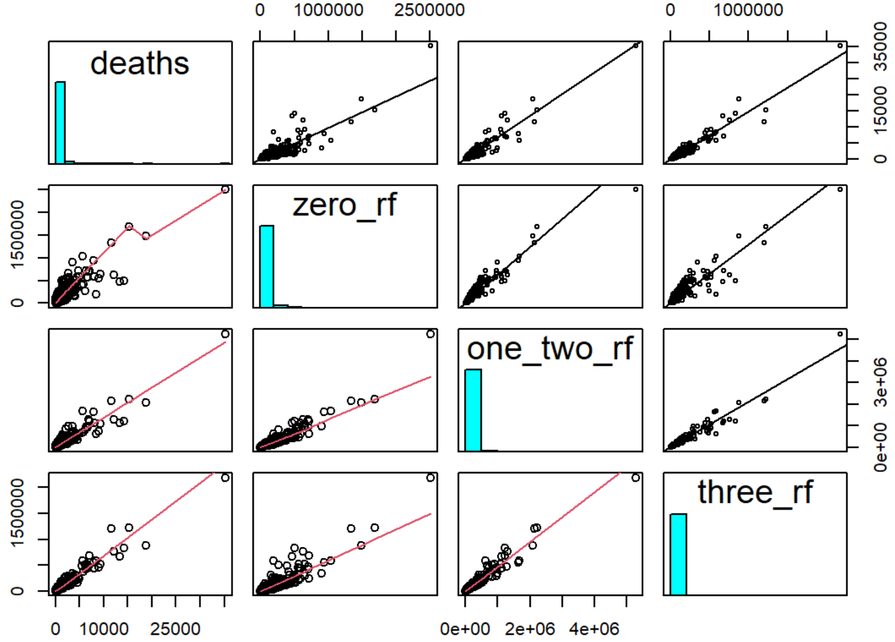
Racialized Populations

```
Call:
lm(formula = deaths ~ hispanic_pop + white_pop + black_pop +
    native_pop + asian_pop + pacific_islander_pop + other_race_pop +
    bi_tri_racial_pop, data = JHU_Cases_Deaths_with_CRE_DataVtwo)

Residuals:
    Min      1Q  Median      3Q     Max
-5050.2   -27.6    -6.4    34.3  5788.3

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           2.566e+01  5.625e+00   4.562 5.26e-06 ***
hispanic_pop          4.264e-03  7.071e-05  60.295  < 2e-16 ***
white_pop             3.229e-03  9.622e-05  33.561  < 2e-16 ***
black_pop             4.038e-03  1.508e-04  26.783  < 2e-16 ***
native_pop            2.269e-02  1.890e-03  12.005  < 2e-16 ***
asian_pop             5.346e-04  3.162e-04   1.691    0.091 .
pacific_islander_pop  4.867e-02  6.024e-03   8.079 9.23e-16 ***
other_race_pop        1.684e-01  6.519e-03  25.836  < 2e-16 ***
bi_tri_racial_pop    -3.042e-02  2.808e-03 -10.833  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 280.3 on 3133 degrees of freedom
Multiple R-squared:  0.938,	Adjusted R-squared:  0.9379
F-statistic:  5927 on 8 and 3133 DF,  p-value: < 2.2e-16
```

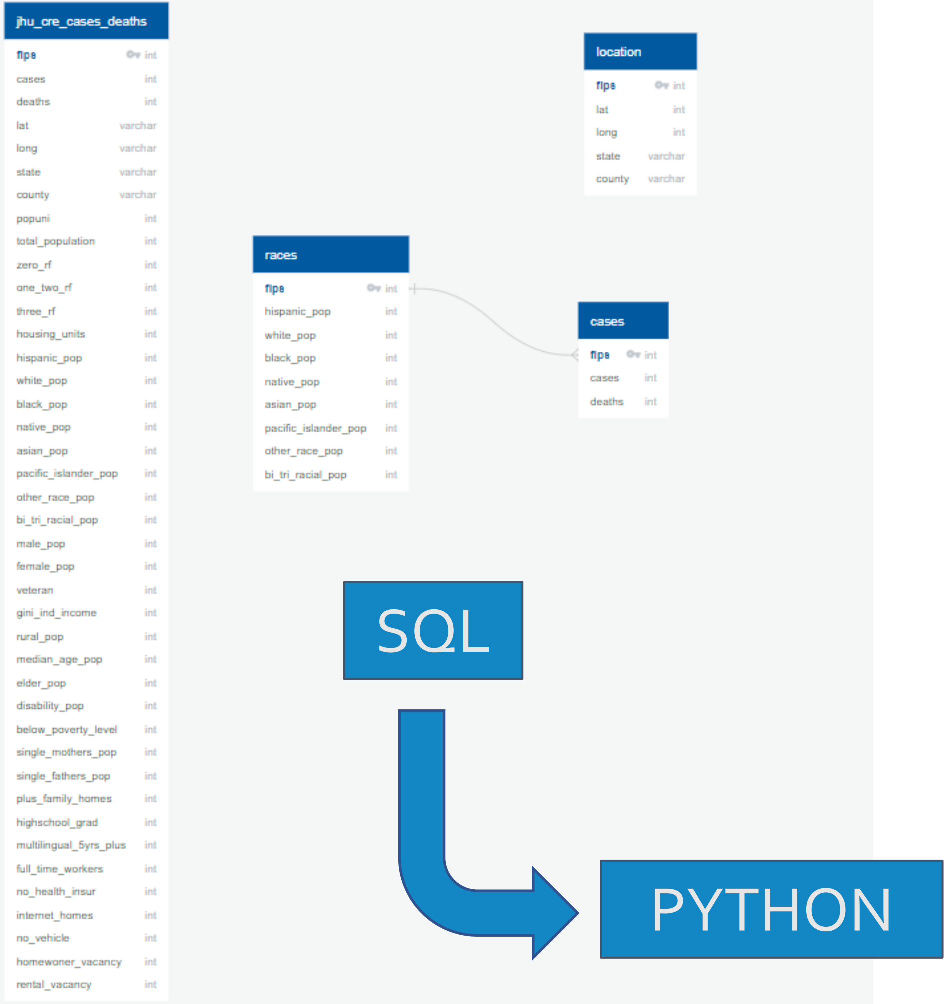


```
Call:
lm(formula = deaths ~ zero_rf + one_two_rf + three_rf, data = JHU_Cases_Deaths_with_CRE_DataVtw
o)

Residuals:
    Min      1Q  Median      3Q     Max
-6043.0   -30.4   -13.7    20.7  5750.3

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 26.9505705  5.1456937   5.237 1.74e-07 ***
zero_rf     -0.0006952  0.0001552  -4.480 7.72e-06 ***
one_two_rf   0.0010104  0.0001796   5.626 2.01e-08 ***
three_rf     0.0135762  0.0003077  44.122  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 268 on 3138 degrees of freedom
Multiple R-squared:  0.9432,	Adjusted R-squared:  0.9432
F-statistic: 1.738e+04 on 3 and 3138 DF,  p-value: < 2.2e-16

>
```

Combined Risk Factors



Data Analysis



In [1]:
```python
import pandas as pd
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
from flask import Flask, jsonify
from sqlalchemy.engine import url
import json
from sqlalchemy import extract
from sqlalchemy.engine import make_url
from sqlalchemy.orm import sessionmaker
from sqlalchemy.sql import text
import psycopg2
```

In [2]:
```python
# opening configuration file, saving into a variable, then establishing that variable as env which specifies the data as develop
# environment credentials
with open('sql/config.json') as datafile:
    data = json.load(datafile)

env = data['dev']
```

In [3]:
```python
# Instantiating the environment column values as the column names to hide sensitive data

db = env['db']
user = env['user']
password = env['pass']
port = env['port']
host = env['host']
```

In [4]:
```python
# Connection string

engine = sqlalchemy.create_engine(f'postgresql://{user}:{password}@{host}:{port}/{db}')
```

In [5]:
```python
# Reflecting an existing DB into a new model
Base = automap_base()
```

In [6]:
```python
Base.prepare(autoload_with = engine)
```

In [7]:
```python
# Seeing what tables there are
Base.classes.keys()
```

Out[7]:
```
['races', 'cases', 'jhu_cre_cases_deaths', 'location']
```

In [8]:
```python
# Saving the tables as references here if u want
```

In [9]:
```python
# Another table reference
jhu_data = Base.classes.jhu_cre_cases_deaths
```

In [10]:
```python
# Session object is the handler to the database, estb convo w db.
# Sessionmaker class creates a 'top level' session configuration that then can be used throughout the application without
# the need to repeat config arguments.'- Credit to rfkortekaas on Stack for explaining.
Session = sessionmaker(bind = engine)
session = Session()
```

In [11]:
```python
# Test query to make sure database connection works
sql = session.query(jhu_data)
```

In [12]:
```python
## An example of using sql commands:
sql = '''
    SELECT * FROM jhu_cre_cases_deaths;
'''
with engine.connect() as conn:
    query = conn.execute(text(sql))
df2 = pd.DataFrame(query.fetchall())
```

In [13]:
```python
df2
```

Out[13]:

	fips	cases	deaths	lat	long	state	county	popuni	total_population	zero_rf	...	single_fathers_pop	plus_family_homes	highsch
0	1001	19732	230	32.539527	-86.644082	Alabama	Autauga County	55688	55380	20573	...	723	111	
1	1003	69641	724	30.727750	-87.722071	Alabama	Baldwin County	221898	212830	78622	...	2218	887	
2	1005	7451	103	31.868263	-85.387129	Alabama	Barbour County	22023	25361	5024	...	220	132	
3	1007	8067	109	32.996421	-87.125115	Alabama	Bibb County	20393	22493	6280	...	346	163	
4	1009	18616	261	33.982109	-86.567906	Alabama	Blount County	57697	57681	18189	...	1038	115	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3137	56037	12484	139	41.659439	-108.882788	Wyoming	Sweetwater County	41888	43521	16977	...	502	209	
3138	56039	12123	16	43.935225	-110.589080	Wyoming	Teton County	23390	23280	7250	...	140	771	
3139	56041	6378	43	41.287818	-110.547578	Wyoming	Uinta County	20183	20479	7744	...	322	141	
3140	56043	2749	50	43.904516	-107.680187	Wyoming	Washakie County	7738	8027	2601	...	108	77	
3141	56045	1903	23	43.839612	-104.567488	Wyoming	Weston County	6664	7049	1857	...	179	26	

3142 rows × 41 columns

Beginning ML portion of project

In [14]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [22]:
```
dfd.deaths.describe()
```

Out[22]:
```
count     3142.00000
mean       349.12317
std       1124.50972
min          0.00000
25%         47.00000
50%        110.00000
75%        261.00000
max      35250.00000
Name: deaths, dtype: float64
```

In [23]:
```
## using lower quartiles, mean, upper quartile 47, 110, 261 ##

death_catlow = 47
death_catmed = 110
death_catintermed = 261
death_cathigh = 369

count = 0


for (column, columnData) in dfd.iterrows():
    deaths = dfd['deaths'].values[count]
    dc = dfd['death_cat']
    if deaths <= death_catlow:
        dc.values[count] = 'low'
    elif deaths > death_catlow and deaths <= death_catmed:
        dc.values[count] = 'med'
    elif deaths > death_catmed and deaths <= death_catintermed:
        dc.values[count] = 'intermed'
    else:
        dc.values[count] = 'high'


    count = count + 1
```

In [24]:
```
# Assigning numerical values and storing in another column

dfd['State_Cat'] = LabelEncoder().fit_transform(dfd['state'])
dfd['County_Cat'] = LabelEncoder().fit_transform(dfd['county'])
dfd['Death_Cat'] = LabelEncoder().fit_transform(dfd['death_cat'])
```

In [28]:
```
dfdd.describe()
```

Out[28]:

	cases	deaths	total_population	zero_rf	one_two_rf	three_rf	housing_units	hispanic_pop	white_pop	black_pop	...	mi
count	3.142000e+03	3142.00000	3.142000e+03	3.142000e+03	3.142000e+03	3.142000e+03	3.142000e+03	3.142000e+03	3.142000e+03	3.142000e+03	...	
mean	3.213884e+04	349.12317	1.033411e+05	3.555582e+04	4.511753e+04	2.219337e+04	4.373933e+04	1.859688e+04	6.241113e+04	1.260647e+04	...	
std	1.109969e+05	1124.50972	3.311701e+05	1.029070e+05	1.583837e+05	7.377844e+04	1.279317e+05	1.257639e+05	1.427950e+05	5.375823e+04	...	
min	0.000000e+00	0.00000	6.600000e+01	2.000000e+01	3.900000e+01	2.700000e+01	6.600000e+01	0.000000e+00	1.900000e+01	0.000000e+00	...	
25%	3.097750e+03	47.00000	1.095200e+04	3.045500e+03	4.517250e+03	2.867000e+03	5.505000e+03	3.412500e+02	7.806000e+03	1.060000e+02	...	
50%	7.899000e+03	110.00000	2.573950e+04	7.858500e+03	1.074350e+04	6.488500e+03	1.249650e+04	1.049500e+03	1.978100e+04	7.770000e+02	...	
75%	2.120975e+04	261.00000	6.786600e+04	2.261150e+04	2.859225e+04	1.540625e+04	3.148100e+04	5.027250e+03	5.261900e+04	5.303500e+03	...	
max	3.691301e+06	35250.00000	1.008157e+07	2.503900e+06	5.264628e+06	2.180574e+06	3.542800e+06	4.825314e+06	2.606664e+06	1.180274e+06	...	

8 rows × 38 columns

In [29]:
```
dfdd
```

Out[29]:

	cases	deaths	total_population	zero_rf	one_two_rf	three_rf	housing_units	hispanic_pop	white_pop	black_pop	...	multilingual_5yrs_plus	full_time_wo
0	19732	230	55380	20573	22750	12365	23493	1559	41543	10580	...	779	
1	69641	724	212830	78622	90552	52724	114164	10207	184397	20414	...	3994	1
2	7451	103	25361	5024	9171	7828	12013	969	10086	10438	...	572	
3	8067	109	22493	6280	8986	5127	9185	530	15192	4506	...	265	
4	18616	261	57681	18189	23950	15558	24323	5365	50138	865	...	1961	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3137	12484	139	43521	16977	17781	7130	19771	6660	33342	460	...	1298	
3138	12123	16	23280	7250	11567	4573	13848	3508	19016	280	...	1777	
3139	6378	43	20479	7744	9346	3093	9041	1836	17660	20	...	322	
3140	2749	50	8027	2601	3215	1922	3860	1098	6337	0	...	108	
3141	1903	23	7049	1857	3367	1440	3562	73	6424	13	...	13	

3142 rows × 38 columns

In [30]:
```
## Dropping homeowner vacancy as it has a suspicious outlier
dfdd = dfdd.drop(['homeowner_vacancy'], axis = 1)
```

Outcomes

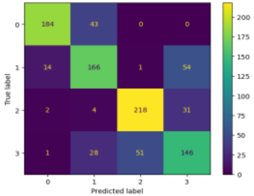
- Hispanic population has 88% correlation with deaths, 80% with zero risk factors, 92% with one to two risk factors and roughly 91% with three risk factors.
- White population has 85% correlation with deaths, a 97% correlation with zero risk factors, 88% with one to two risk factors, and 86% with three risk factors.
- Black population has a 77% correlation with deaths, a 76% correlation with zero risk factors, 75% with one to two risk factors, and an 83% with three risk factors.
- Asian population has a 76% correlation with deaths, a 78% percent correlation with zero risk factors, a 85% correlation with one to two risk factors, and a 80% correlation with three risk factors.
- The Bi and Tri racial populations have a 80% correlation with deaths, an 88% correlation with zero risk factors, an 89% correlation with one to two risk factors, and 83% correlation with three risk factors.
- Other Race population has a 80% correlation with the target, 73% correlation with zero risk factors, 77% with one to two risk factors and and and 78% with three risk factors.
- The native population has a 36% correlation with the target, 34% correlation with zero risk factors, 34% with one to two risk factors, and 33% with three risk factors.
- Pacific Islander population has a 26% correlation with the target, 34% correlation with zero risk factors, 36% with one to two risk factors, and 30% with three risk factors.
- **Zero Risk factors as a whole, has a 90% correlation with the target, One to Two risk factors has a 95% correlation with the target, and three risk factors has a 97% correlation with the target.

Classifier Models

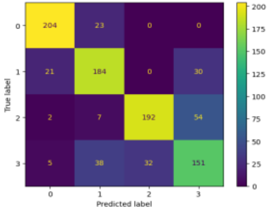
```
In [21]: from sklearn.decomposition import PCA
         # normalizing data

In [22]: # Creating features

         X = dfdd.drop(['total_population',
                        'gini_ind_income',
                        'State_Cat','County_Cat',
                        'plus_family_homes',
                        'rural_pop','rental_vacancy',
                        'Death_Cat'],
                       axis = 1)

         y = dfdd['Death_Cat']

In [ ]:

In [23]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(X,y,
                                                    test_size= .30,
                                                    train_size = .70,
                                                    random_state=49)

In [24]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()

         # Fit on the training set
         scaler.fit(x_train)

Out[24]: StandardScaler()

In [25]: # Apply to both the train set and the test set.
         X_train = scaler.transform(x_train)

In [26]: X_test = scaler.transform(x_test)

In [27]: # Apply PCA

In [28]: pca = PCA(n_components = 2)

In [29]: # Fit on the train set only
         pca.fit(X_train)

Out[29]: PCA(n_components=2)

In [30]: #Apply transformation on both train and test set
         X_train = pca.transform(X_train)
         X_test = pca.transform(X_test)

In [31]: var = pca.explained_variance_ratio_
         var.sum()

Out[31]: 0.8928846717018712

In [32]: # Storing methods as variables for formatting

         cr = classification_report
         ar = accuracy_score
         cm = confusion_matrix
```

```
In [37]: from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score

         lm = LinearRegression()
         lm.fit(X_train, y_train)
         predictlm = lm.predict(X_test)

         # Score the predictions with mse and r2
         mse = mean_squared_error(y_test, predictlm)
         r2 = r2_score(y_test, predictlm)
         print(f"mean squared error (MSE): {mse}")

         print(f"R-squared (R2): {r2}")
         lm.score(X_test, y_test)

         mean squared error (MSE): 1.060813458618592
         R-squared (R2): 0.12378872405194485

Out[37]: 0.12378872405194485
```

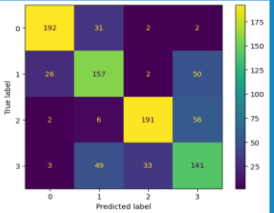
```
In [33]: from sklearn.linear_model import LogisticRegression

         LRM = LogisticRegression()
         LRM.fit(X_train,y_train)
         predLRM = LRM.predict(X_test)

         print(f"the accuracy score is:{ar(y_test,predLRM)}")

         print(f"confusion matrix:\n{cm(y_test,predLRM)}")

         the accuracy score is:0.7571580063626723
         confusion matrix:
         [[184  43   0   0]
          [ 14 166   1  54]
          [  2   4 218  31]
          [  1  28  51 146]]
```



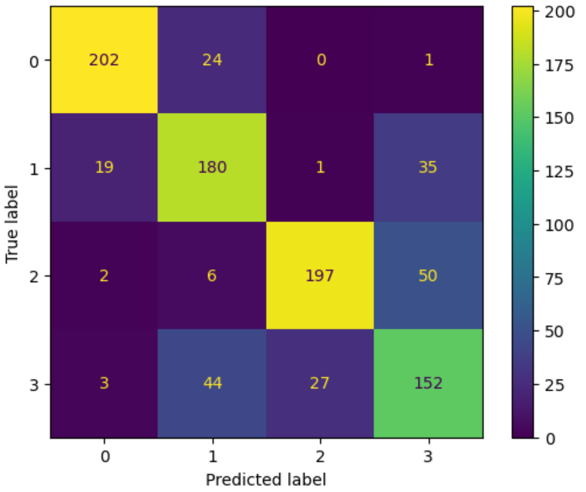

```
In [36]: from sklearn.neighbors import KNeighborsClassifier

         neighborC = KNeighborsClassifier()
         neighborC.fit(X_train, y_train)
         predneighbx = neighborC.predict(X_test)

         print(f"the accuracy score is:{ar(y_test, predneighbx)}")

         print(f"confusion matrix:\n{cm(y_test, predneighbx)}")

         the accuracy score is:0.7751855779427359
         confusion matrix:
         [[204  23   0   0]
          [ 21 184   0  30]
          [  2   7 192  54]
          [  5  38  32 151]]
```



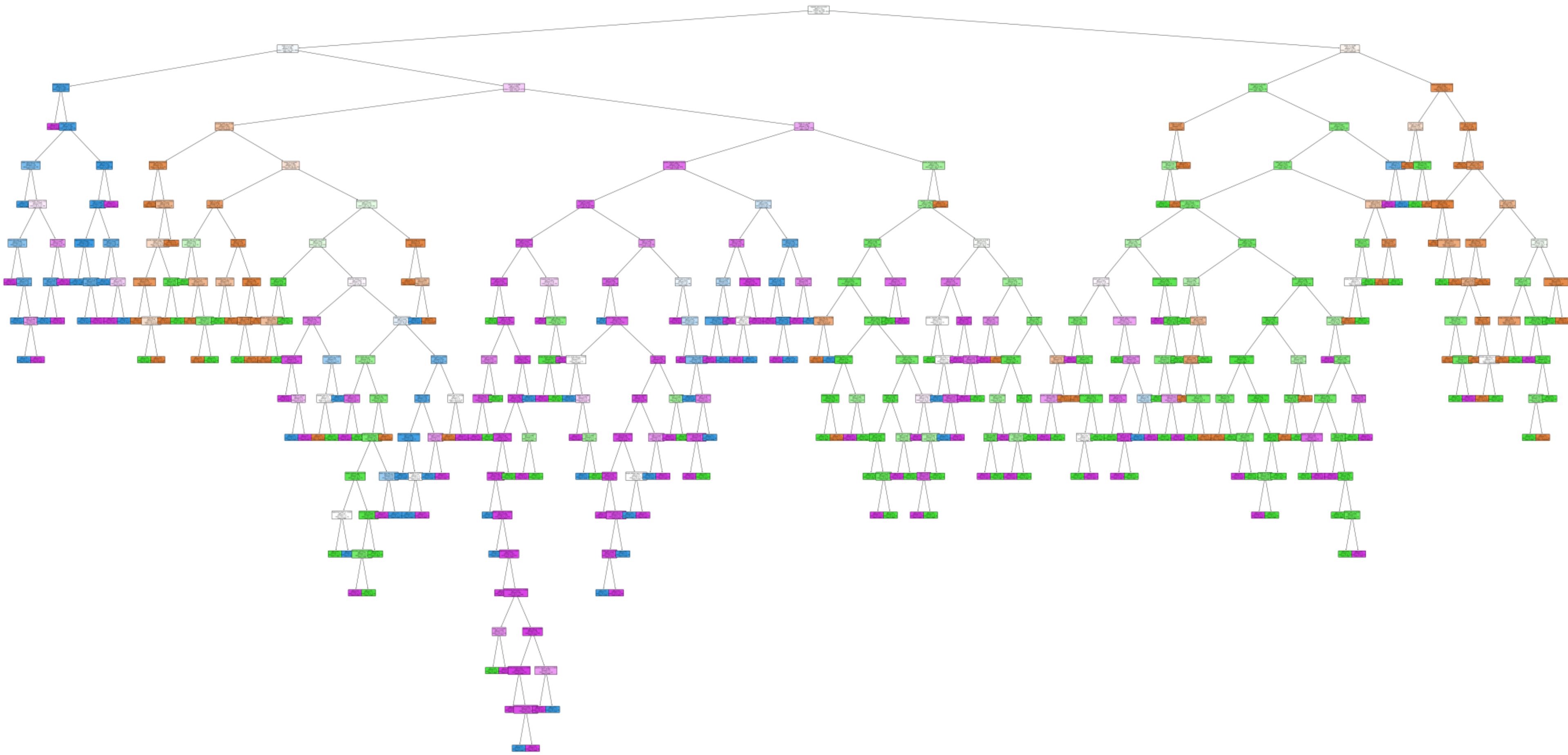

```
In [35]: from sklearn.tree import DecisionTreeClassifier

         DT = DecisionTreeClassifier()
         DT.fit(X_train, y_train)
         DTpredict = DT.predict(X_test)

         print(f"the accuracy score is:{ar(y_test, DTpredict)}")

         print(f"confusion matrix:\n{cm(y_test, DTpredict)}")

         the accuracy score is:0.7221633085896076
         confusion matrix:
         [[192  31   2   2]
          [ 26 157   2  50]
          [  2   6 191  56]
          [  3  49  33 141]]
```




```
In [34]: from sklearn.ensemble import RandomForestClassifier

         RFC = RandomForestClassifier()
         RFC.fit(X_train, y_train)
         predRFC = RFC.predict(X_test)


         print(f"the accuracy score is:{ar(y_test, predRFC)}")

         print(f"confusion_matrix:\n{cm(y_test, predRFC)}")

         the accuracy score is:0.7751855779427359
         confusion_matrix:
         [[202  24   0   1]
          [ 19 180   1  35]
          [  2   6 197  50]
          [  3  44  27 152]]
```



Decision Tree for Random Forest Classifier Model



QUESTIONS

&

ANSWERS

THANK YOU!