COVID-19 FORECASTING

The purpose of the current project is to develop estimates for confirmed cases and fatalities between May 12 and June 7 by county in the United States as well as identifying factors that appear to impact the transmission rate of COVID-19.

> *Note:* conda install -c plotly plotly

Packages to be Imported

In [1]:
```
#Data Visualization
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.express as px
import plotly.graph_objects as go

#Machine Learning
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score
```

General Info about the Dataset

In [2]:
```
pwd
```

Out[2]:
```
'/Users/admin/Desktop/python/covid19/forecasting'
```

In [3]:
```
train = pd.read_csv('/Users/admin/Desktop/python/covid19/forecasting/datasets/train.csv', sep=',')
train
```

Out[3]:

	Id	County	Province_State	Country_Region	Population	Weight	Date	Target	TargetValue
0	1	NaN	NaN	Afghanistan	27657145	0.058359	2020-01-23	ConfirmedCases	0.0
1	2	NaN	NaN	Afghanistan	27657145	0.583587	2020-01-23	Fatalities	0.0
2	3	NaN	NaN	Afghanistan	27657145	0.058359	2020-01-24	ConfirmedCases	0.0
3	4	NaN	NaN	Afghanistan	27657145	0.583587	2020-01-24	Fatalities	0.0
4	5	NaN	NaN	Afghanistan	27657145	0.058359	2020-01-25	ConfirmedCases	0.0
...	...	...	...	...	...	...	...	...	...
727225	969566	NaN	NaN	Zimbabwe	14240168	0.607106	2020-05-04	Fatalities	0.0
727226	969567	NaN	NaN	Zimbabwe	14240168	0.060711	2020-05-05	ConfirmedCases	0.0
727227	969568	NaN	NaN	Zimbabwe	14240168	0.607106	2020-05-05	Fatalities	0.0
727228	969569	NaN	NaN	Zimbabwe	14240168	0.060711	2020-05-06	ConfirmedCases	0.0
727229	969570	NaN	NaN	Zimbabwe	14240168	0.607106	2020-05-06	Fatalities	0.0

727230 rows × 9 columns

In [4]:
```
test = pd.read_csv('/Users/admin/Desktop/python/covid19/forecasting/datasets/test.csv', sep=',')
test
```

Out[4]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
0	1	NaN	NaN	Afghanistan	27657145	0.058359	2020-04-27	ConfirmedCases
1	2	NaN	NaN	Afghanistan	27657145	0.583587	2020-04-27	Fatalities
2	3	NaN	NaN	Afghanistan	27657145	0.058359	2020-04-28	ConfirmedCases
3	4	NaN	NaN	Afghanistan	27657145	0.583587	2020-04-28	Fatalities
4	5	NaN	NaN	Afghanistan	27657145	0.058359	2020-04-29	ConfirmedCases
...	...	...	...	...	...	...	...	...
311665	311666	NaN	NaN	Zimbabwe	14240168	0.607106	2020-06-08	Fatalities
311666	311667	NaN	NaN	Zimbabwe	14240168	0.060711	2020-06-09	ConfirmedCases
311667	311668	NaN	NaN	Zimbabwe	14240168	0.607106	2020-06-09	Fatalities
311668	311669	NaN	NaN	Zimbabwe	14240168	0.060711	2020-06-10	ConfirmedCases
311669	311670	NaN	NaN	Zimbabwe	14240168	0.607106	2020-06-10	Fatalities

311670 rows × 8 columns

Understanding and Cleaning Data

In [5]:
```
train.shape
```

Out[5]:
```
(727230, 9)
```

In [6]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 727230 entries, 0 to 727229
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Id              727230 non-null  int64  
 1   County          660030 non-null  object 
 2   Province_State  687960 non-null  object 
 3   Country_Region  727230 non-null  object 
 4   Population      727230 non-null  int64  
 5   Weight          727230 non-null  float64
 6   Date            727230 non-null  object 
 7   Target          727230 non-null  object 
 8   TargetValue     727230 non-null  float64
dtypes: float64(2), int64(2), object(5)
memory usage: 49.9+ MB
```

In [7]:
```
train.dtypes
```

Out[7]:
```
Id                 int64
County            object
Province_State    object
Country_Region    object
Population         int64
Weight           float64
Date              object
Target            object
TargetValue      float64
dtype: object
```

In [8]:
```
train.describe()
```

Out[8]:

	Id	Population	Weight	TargetValue
count	727230.000000	7.272300e+05	727230.000000	727230.000000
mean	484785.500000	2.719395e+06	0.530872	9.316042
std	279911.144854	3.477762e+07	0.451909	256.630911
min	1.000000	8.600000e+01	0.047491	-10034.000000
25%	242358.250000	1.213300e+04	0.096838	0.000000
50%	484785.500000	3.053100e+04	0.349413	0.000000
75%	727212.750000	1.056120e+05	0.968379	0.000000
max	969570.000000	1.395773e+09	2.239186	36163.000000

In [9]:
```
#Data Description for String Columns
train.describe(include=[np.object])
```

Out[9]:

	County	Province_State	Country_Region	Date	Target
count	660030	687960	727230	727230	727230
unique	1840	133	187	105	2
top	Washington	Texas	US	2020-03-26	ConfirmedCases
freq	6510	53550	671580	6926	363615

Handling Null Values

In [10]:
```
train.isna().any()
```

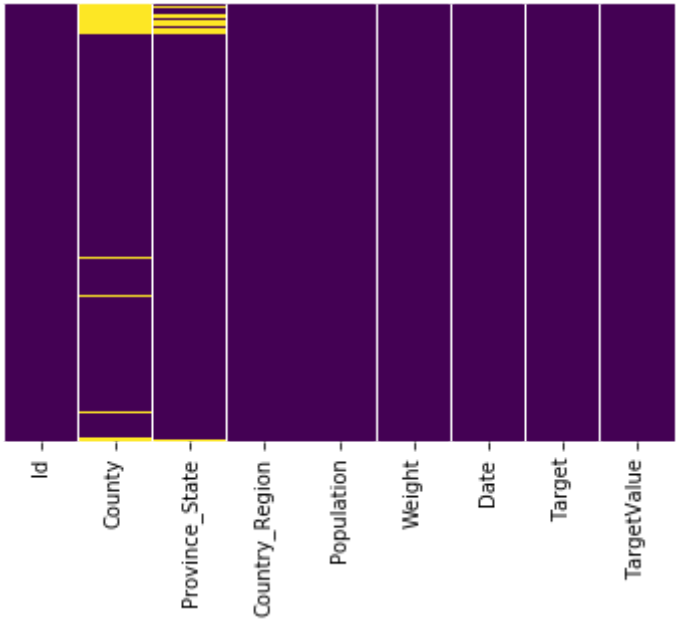
Out[10]:
```
Id                False
County             True
Province_State     True
Country_Region    False
Population        False
Weight            False
Date              False
Target            False
TargetValue       False
dtype: bool
```

In [11]:
```
train.isna().sum()
```

Out[11]:
```
Id                    0
County            67200
Province_State    39270
Country_Region        0
Population            0
Weight                0
Date                  0
Target                0
TargetValue           0
dtype: int64
```

In [12]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x104b33790>
```



In [13]:
```
#Subsetting US Data
train = train[train.Country_Region == 'US']
test = test[test.Country_Region == 'US']
```

In [14]:
```
train
```

Out[14]:

	Id	County	Province_State	Country_Region	Population	Weight	Date	Target	TargetValue
50820	67761	Autauga	Alabama	US	55869	0.091485	2020-01-23	ConfirmedCases	0.0
50821	67762	Autauga	Alabama	US	55869	0.914848	2020-01-23	Fatalities	0.0
50822	67763	Autauga	Alabama	US	55869	0.091485	2020-01-24	ConfirmedCases	0.0
50823	67764	Autauga	Alabama	US	55869	0.914848	2020-01-24	Fatalities	0.0
50824	67765	Autauga	Alabama	US	55869	0.091485	2020-01-25	ConfirmedCases	0.0
...	...	...	...	...	...	...	...	...	...
722395	963126	NaN	NaN	US	324141489	0.510290	2020-05-04	Fatalities	1240.0
722396	963127	NaN	NaN	US	324141489	0.051029	2020-05-05	ConfirmedCases	23976.0
722397	963128	NaN	NaN	US	324141489	0.510290	2020-05-05	Fatalities	2142.0
722398	963129	NaN	NaN	US	324141489	0.051029	2020-05-06	ConfirmedCases	24251.0
722399	963130	NaN	NaN	US	324141489	0.510290	2020-05-06	Fatalities	2367.0

671580 rows × 9 columns

In [15]:
```
test
```

Out[15]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21780	21781	Autauga	Alabama	US	55869	0.091485	2020-04-27	ConfirmedCases
21781	21782	Autauga	Alabama	US	55869	0.914848	2020-04-27	Fatalities
21782	21783	Autauga	Alabama	US	55869	0.091485	2020-04-28	ConfirmedCases
21783	21784	Autauga	Alabama	US	55869	0.914848	2020-04-28	Fatalities
21784	21785	Autauga	Alabama	US	55869	0.091485	2020-04-29	ConfirmedCases
...	...	...	...	...	...	...	...	...
309595	309596	NaN	NaN	US	324141489	0.510290	2020-06-08	Fatalities
309596	309597	NaN	NaN	US	324141489	0.051029	2020-06-09	ConfirmedCases
309597	309598	NaN	NaN	US	324141489	0.510290	2020-06-09	Fatalities
309598	309599	NaN	NaN	US	324141489	0.051029	2020-06-10	ConfirmedCases
309599	309600	NaN	NaN	US	324141489	0.510290	2020-06-10	Fatalities

287820 rows × 8 columns

In [16]:
```
#Droping of Unwanted Data to make model more Predictive
train = train.dropna()
test = test.dropna()
```

In [17]:
```
train
```

Out[17]:

	Id	County	Province_State	Country_Region	Population	Weight	Date	Target	TargetValue
50820	67761	Autauga	Alabama	US	55869	0.091485	2020-01-23	ConfirmedCases	0.0
50821	67762	Autauga	Alabama	US	55869	0.914848	2020-01-23	Fatalities	0.0
50822	67763	Autauga	Alabama	US	55869	0.091485	2020-01-24	ConfirmedCases	0.0
50823	67764	Autauga	Alabama	US	55869	0.914848	2020-01-24	Fatalities	0.0
50824	67765	Autauga	Alabama	US	55869	0.091485	2020-01-25	ConfirmedCases	0.0
...	...	...	...	...	...	...	...	...	...
721975	962566	Weston	Wyoming	US	6927	1.130796	2020-05-04	Fatalities	0.0
721976	962567	Weston	Wyoming	US	6927	0.113080	2020-05-05	ConfirmedCases	0.0
721977	962568	Weston	Wyoming	US	6927	1.130796	2020-05-05	Fatalities	0.0
721978	962569	Weston	Wyoming	US	6927	0.113080	2020-05-06	ConfirmedCases	0.0
721979	962570	Weston	Wyoming	US	6927	1.130796	2020-05-06	Fatalities	0.0

660030 rows × 9 columns

In [18]:
```
test
```

Out[18]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21780	21781	Autauga	Alabama	US	55869	0.091485	2020-04-27	ConfirmedCases
21781	21782	Autauga	Alabama	US	55869	0.914848	2020-04-27	Fatalities
21782	21783	Autauga	Alabama	US	55869	0.091485	2020-04-28	ConfirmedCases
21783	21784	Autauga	Alabama	US	55869	0.914848	2020-04-28	Fatalities
21784	21785	Autauga	Alabama	US	55869	0.091485	2020-04-29	ConfirmedCases
...	...	...	...	...	...	...	...	...
309415	309416	Weston	Wyoming	US	6927	1.130796	2020-06-08	Fatalities
309416	309417	Weston	Wyoming	US	6927	0.113080	2020-06-09	ConfirmedCases
309417	309418	Weston	Wyoming	US	6927	1.130796	2020-06-09	Fatalities
309418	309419	Weston	Wyoming	US	6927	0.113080	2020-06-10	ConfirmedCases
309419	309420	Weston	Wyoming	US	6927	1.130796	2020-06-10	Fatalities

282870 rows × 8 columns

In [19]:
```
test = test[test['Date'] > '2020-05-06']
test
```

Out[19]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21800	21801	Autauga	Alabama	US	55869	0.091485	2020-05-07	ConfirmedCases
21801	21802	Autauga	Alabama	US	55869	0.914848	2020-05-07	Fatalities
21802	21803	Autauga	Alabama	US	55869	0.091485	2020-05-08	ConfirmedCases
21803	21804	Autauga	Alabama	US	55869	0.914848	2020-05-08	Fatalities
21804	21805	Autauga	Alabama	US	55869	0.091485	2020-05-09	ConfirmedCases
...	...	...	...	...	...	...	...	...
309415	309416	Weston	Wyoming	US	6927	1.130796	2020-06-08	Fatalities
309416	309417	Weston	Wyoming	US	6927	0.113080	2020-06-09	ConfirmedCases
309417	309418	Weston	Wyoming	US	6927	1.130796	2020-06-09	Fatalities
309418	309419	Weston	Wyoming	US	6927	0.113080	2020-06-10	ConfirmedCases
309419	309420	Weston	Wyoming	US	6927	1.130796	2020-06-10	Fatalities

220010 rows × 8 columns

In [20]:
```
result = test.copy()
result
```

Out[20]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21800	21801	Autauga	Alabama	US	55869	0.091485	2020-05-07	ConfirmedCases
21801	21802	Autauga	Alabama	US	55869	0.914848	2020-05-07	Fatalities
21802	21803	Autauga	Alabama	US	55869	0.091485	2020-05-08	ConfirmedCases
21803	21804	Autauga	Alabama	US	55869	0.914848	2020-05-08	Fatalities
21804	21805	Autauga	Alabama	US	55869	0.091485	2020-05-09	ConfirmedCases
...	...	...	...	...	...	...	...	...
309415	309416	Weston	Wyoming	US	6927	1.130796	2020-06-08	Fatalities
309416	309417	Weston	Wyoming	US	6927	0.113080	2020-06-09	ConfirmedCases
309417	309418	Weston	Wyoming	US	6927	1.130796	2020-06-09	Fatalities
309418	309419	Weston	Wyoming	US	6927	0.113080	2020-06-10	ConfirmedCases
309419	309420	Weston	Wyoming	US	6927	1.130796	2020-06-10	Fatalities

220010 rows × 8 columns

Data Visualization

Performing Correlation Matrix for Train Data

In [21]:
```
np.corrcoef(train['TargetValue'], train['Weight'])
```

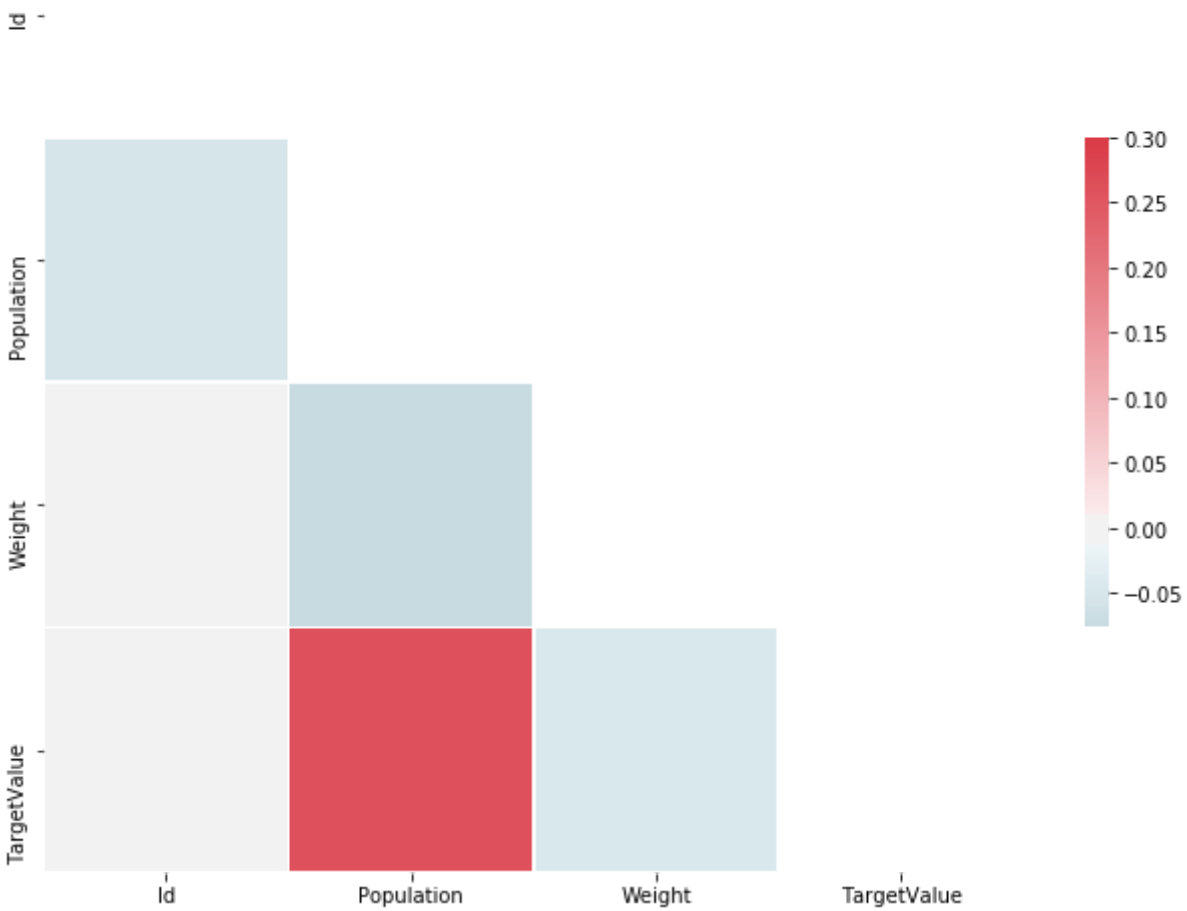
Out[21]:
```
array([[ 1.        , -0.04587647],
       [-0.04587647,  1.        ]])
```

In [22]:
```
np.corrcoef(train['TargetValue'], train['Population'])
```

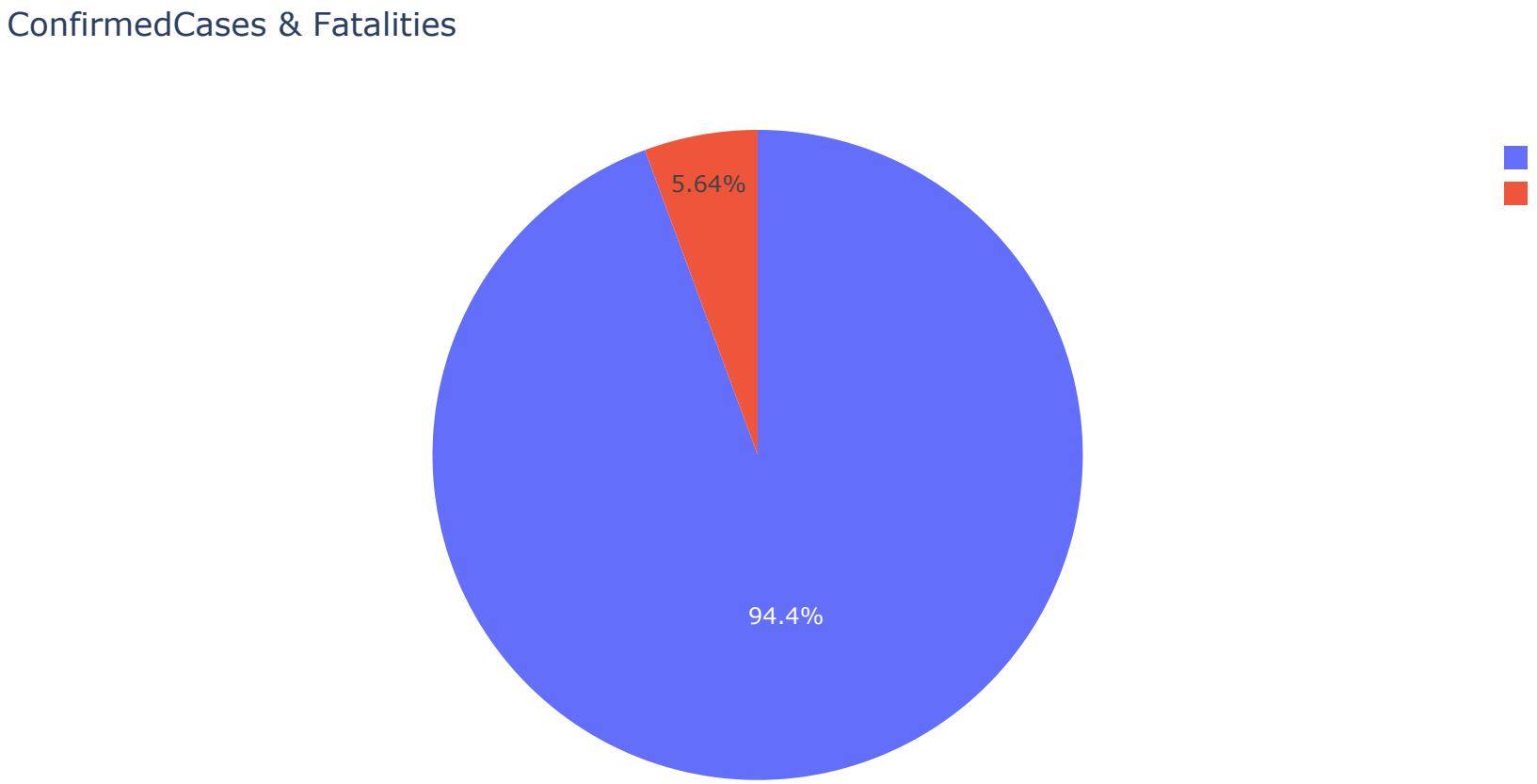
Out[22]:
```
array([[1.        , 0.25947227],
       [0.25947227, 1.        ]])
```

In [23]:
```
corr = train.corr()
mask = np.triu(np.ones_like(corr, dtype=np.bool))
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

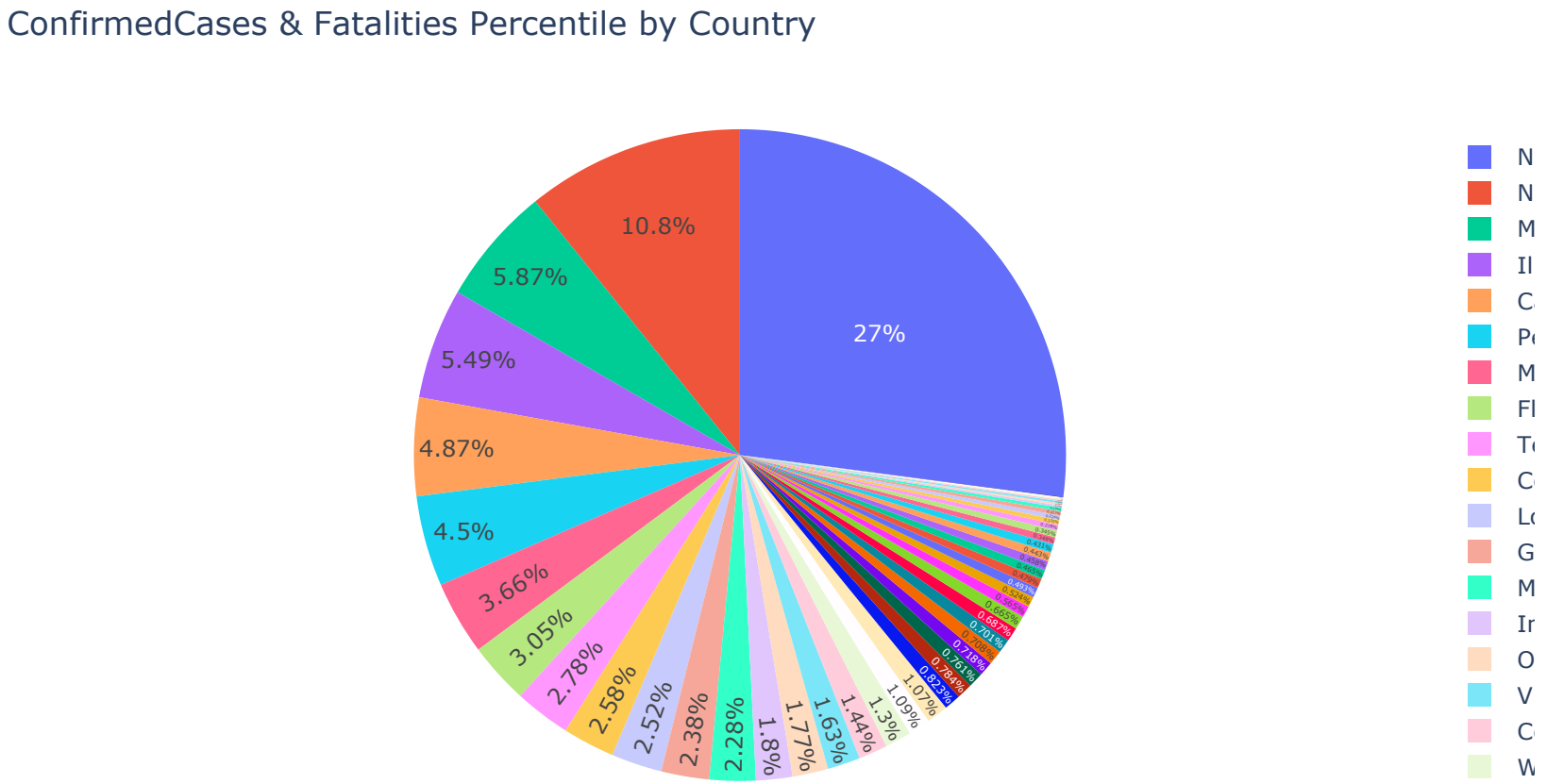
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1a6c6950>



In [24]:
```
fig = px.pie(train, values='TargetValue', names='Target', title='ConfirmedCases & Fatalities')
fig.show()
```



In [25]:
```
fig = px.pie(train, values='TargetValue', names='Province_State', title='ConfirmedCases & Fatalities Per
fig.update_traces(textposition='inside')
fig.show()
```



In [26]:
```
last_date=train.Date.max()
df=train[train["Date"]==last_date]
df
```

Out[26]:

	Id	County	Province_State	Country_Region	Population	Weight	Date	Target	TargetValue
51028	67969	Autauga	Alabama	US	55869	0.091485	2020-05-06	ConfirmedCases	5.0
51029	67970	Autauga	Alabama	US	55869	0.914848	2020-05-06	Fatalities	0.0
51238	68249	Baldwin	Alabama	US	223234	0.081195	2020-05-06	ConfirmedCases	7.0
51239	68250	Baldwin	Alabama	US	223234	0.811953	2020-05-06	Fatalities	0.0
51448	68529	Barbour	Alabama	US	24686	0.098873	2020-05-06	ConfirmedCases	0.0
...	...	...	...	...	...	...	...	...	...
721559	962010	Uinta	Wyoming	US	20226	1.008596	2020-05-06	Fatalities	0.0
721768	962289	Washakie	Wyoming	US	7805	0.111574	2020-05-06	ConfirmedCases	0.0
721769	962290	Washakie	Wyoming	US	7805	1.115742	2020-05-06	Fatalities	0.0
721978	962569	Weston	Wyoming	US	6927	0.113080	2020-05-06	ConfirmedCases	0.0
721979	962570	Weston	Wyoming	US	6927	1.130796	2020-05-06	Fatalities	0.0

6286 rows × 9 columns

In [27]:
```
df=df.groupby(by=["Province_State"],as_index=False)["TargetValue"].sum()
df
```

Out[27]:

	Province_State	TargetValue
0	Alabama	282.0
1	Alaska	2.0
2	Arizona	434.0
3	Arkansas	108.0
4	California	2243.0
5	Colorado	506.0
6	Connecticut	522.0
7	Delaware	402.0
8	District of Columbia	152.0
9	Florida	617.0
10	Georgia	960.0
11	Hawaii	1.0
12	Idaho	32.0
13	Illinois	2542.0
14	Indiana	890.0
15	Iowa	292.0
16	Kansas	348.0
17	Kentucky	136.0
18	Louisiana	454.0
19	Maine	29.0
20	Maryland	1102.0
21	Massachusetts	2232.0
22	Michigan	757.0

	Province_State	TargetValue
23	Minnesota	770.0
24	Mississippi	249.0
25	Missouri	221.0
26	Montana	0.0
27	Nebraska	316.0
28	Nevada	94.0
29	New Hampshire	100.0
30	New Jersey	1558.0
31	New Mexico	160.0
32	New York	3285.0
33	North Carolina	563.0
34	North Dakota	63.0
35	Ohio	697.0
36	Oklahoma	81.0
37	Oregon	79.0
38	Pennsylvania	1102.0
39	Rhode Island	0.0
40	South Carolina	104.0
41	South Dakota	59.0
42	Tennessee	285.0
43	Texas	1045.0
44	Utah	138.0
45	Vermont	1.0
46	Virginia	0.0
47	Washington	295.0
48	West Virginia	-2.0

	Province_State	TargetValue
49	Wisconsin	344.0
50	Wyoming	27.0

In [28]:
```
states=df.nlargest(5,"TargetValue")
states
```

Out[28]:

	Province_State	TargetValue
32	New York	3285.0
13	Illinois	2542.0
4	California	2243.0
21	Massachusetts	2232.0
30	New Jersey	1558.0

In [29]:
```
cases=train.groupby(by=["Date","Province_State"],as_index=False)["TargetValue"].sum()
cases
```

Out[29]:

	Date	Province_State	TargetValue
0	2020-01-23	Alabama	0.0
1	2020-01-23	Alaska	0.0
2	2020-01-23	Arizona	0.0
3	2020-01-23	Arkansas	0.0
4	2020-01-23	California	0.0
...	...	...	...
5350	2020-05-06	Virginia	0.0
5351	2020-05-06	Washington	295.0
5352	2020-05-06	West Virginia	-2.0
5353	2020-05-06	Wisconsin	344.0
5354	2020-05-06	Wyoming	27.0

5355 rows × 3 columns

In [30]:
```
cases=cases.merge(states,on="Province_State")
cases
```

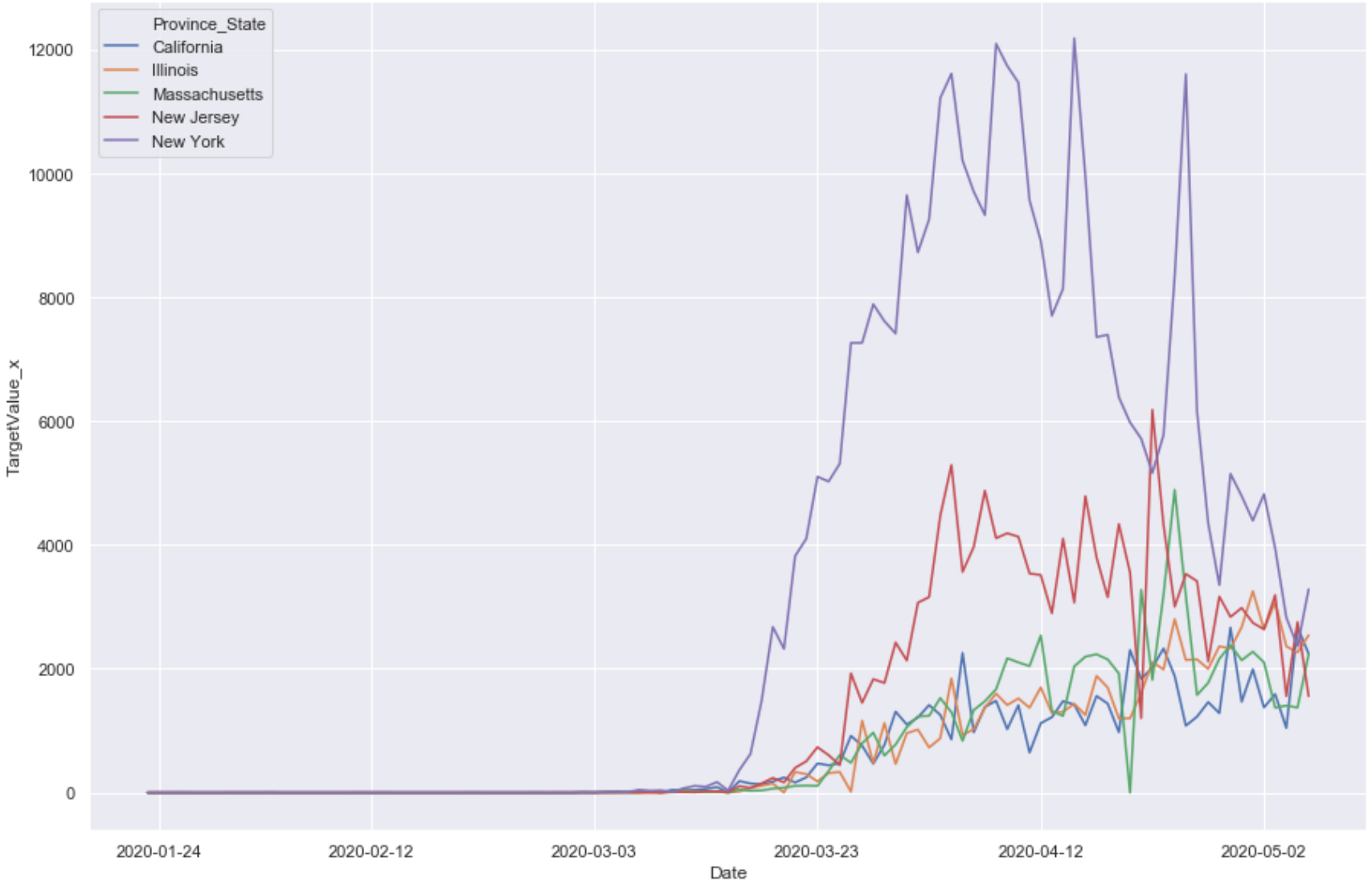
Out[30]:

	Date	Province_State	TargetValue_x	TargetValue_y
0	2020-01-23	California	0.0	2243.0
1	2020-01-24	California	0.0	2243.0
2	2020-01-25	California	0.0	2243.0
3	2020-01-26	California	2.0	2243.0
4	2020-01-27	California	0.0	2243.0
...	...	...	...	...
520	2020-05-02	New York	4822.0	3285.0
521	2020-05-03	New York	3948.0	3285.0
522	2020-05-04	New York	2829.0	3285.0
523	2020-05-05	New York	2364.0	3285.0
524	2020-05-06	New York	3285.0	3285.0

525 rows × 4 columns

In [31]:
```
plt.figure(figsize=(15,10))
sns.set(style="darkgrid")
sns.lineplot(x="Date",y="TargetValue_x",hue="Province_State",data=cases)
plt.xticks([1,20,40,60,80,100])
```

Out[31]:
```
([<matplotlib.axis.XTick at 0x1a1b0221d0>,
  <matplotlib.axis.XTick at 0x1a1b01a810>,
  <matplotlib.axis.XTick at 0x1a1aff2c10>,
  <matplotlib.axis.XTick at 0x1a3a107650>,
  <matplotlib.axis.XTick at 0x1a3a107410>,
  <matplotlib.axis.XTick at 0x1a4e5c1650>],
 <a list of 6 Text xticklabel objects>)
```



Preparing Data for Model

In [32]:
```
pd.options.mode.chained_assignment = None  # default='warn'
```

In [33]:
```
#Converting String Date into Integer for both Train and Test Datasets
train["Date"] = pd.to_datetime(train["Date"]).dt.strftime("%Y%m%d")
test["Date"] = pd.to_datetime(test["Date"]).dt.strftime("%Y%m%d")
```

In [34]:
```
#Applying Label Encoding for Categorial features
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

train['County']= le.fit_transform(train['County'])
train['Province_State']= le.fit_transform(train['Province_State'])
train['Target']= le.fit_transform(train['Target'])
test['County']= le.fit_transform(test['County'])
test['Province_State']= le.fit_transform(test['Province_State'])
test['Target']= le.fit_transform(test['Target'])
```

In [35]:
```
train
```

Out[35]:

	Id	County	Province_State	Country_Region	Population	Weight	Date	Target	TargetValue
50820	67761	81	0	US	55869	0.091485	20200123	0	0.0
50821	67762	81	0	US	55869	0.914848	20200123	1	0.0
50822	67763	81	0	US	55869	0.091485	20200124	0	0.0
50823	67764	81	0	US	55869	0.914848	20200124	1	0.0
50824	67765	81	0	US	55869	0.091485	20200125	0	0.0
...	...	...	...	...	...	...	...	...	...
721975	962566	1763	50	US	6927	1.130796	20200504	1	0.0
721976	962567	1763	50	US	6927	0.113080	20200505	0	0.0
721977	962568	1763	50	US	6927	1.130796	20200505	1	0.0
721978	962569	1763	50	US	6927	0.113080	20200506	0	0.0
721979	962570	1763	50	US	6927	1.130796	20200506	1	0.0

660030 rows × 9 columns

In [36]:
```
test
```

Out[36]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21800	21801	81	0	US	55869	0.091485	20200507	0
21801	21802	81	0	US	55869	0.914848	20200507	1
21802	21803	81	0	US	55869	0.091485	20200508	0
21803	21804	81	0	US	55869	0.914848	20200508	1
21804	21805	81	0	US	55869	0.091485	20200509	0
...	...	...	...	...	...	...	...	...
309415	309416	1763	50	US	6927	1.130796	20200608	1
309416	309417	1763	50	US	6927	0.113080	20200609	0
309417	309418	1763	50	US	6927	1.130796	20200609	1
309418	309419	1763	50	US	6927	0.113080	20200610	0
309419	309420	1763	50	US	6927	1.130796	20200610	1

220010 rows × 8 columns

In [37]:
```
predictors = train.drop(['TargetValue', 'Id', 'Country_Region'], axis=1)
target = train["TargetValue"]
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size = 0.3, random_state =
```

In [39]:
```
test.drop(['ForecastId', 'Country_Region'],axis=1,inplace=True)
test.index.name = 'Id'
test
```

Out[39]:

Id	County	Province_State	Population	Weight	Date	Target
21800	81	0	55869	0.091485	20200507	0
21801	81	0	55869	0.914848	20200507	1
21802	81	0	55869	0.091485	20200508	0
21803	81	0	55869	0.914848	20200508	1
21804	81	0	55869	0.091485	20200509	0
...	...	...	...	...	...	...
309415	1763	50	6927	1.130796	20200608	1
309416	1763	50	6927	0.113080	20200609	0
309417	1763	50	6927	1.130796	20200609	1
309418	1763	50	6927	0.113080	20200610	0
309419	1763	50	6927	1.130796	20200610	1

220010 rows × 6 columns

In [ ]:
```
```

Model Building

Random Forest

In [40]:
```
pipe_rfr = Pipeline([
    ('scaler', StandardScaler()),
    ('regression', RandomForestRegressor(n_estimators=150))
])
pipe_rfr.fit(X_train, y_train)
#MaxAbsScaler dene
```

Out[40]:
```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('regression',
                 RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                       criterion='mse', max_depth=None,
                                       max_features='auto', max_leaf_nodes=None,
                                       max_samples=None,
                                       min_impurity_decrease=0.0,
                                       min_impurity_split=None,
                                       min_samples_leaf=1, min_samples_split=2,
                                       min_weight_fraction_leaf=0.0,
                                       n_estimators=150, n_jobs=None,
                                       oob_score=False, random_state=None,
                                       verbose=0, warm_start=False))],
         verbose=False)
```

In [41]:
```
'''model = RandomForestRegressor(n_estimators=150, n_jobs=-1)
model.fit(X_train,y_train)
print(r2_score(y_test,model.predict(X_test)))'''
```

Out[41]:
```
'model = RandomForestRegressor(n_estimators=150, n_jobs=-1)\nmodel.fit(X_train,y_train)\nprint(r2_scor
e(y_test,model.predict(X_test)))'
```

In [42]:
```
'''#Fitting the model RandomForestRegressor
model = RandomForestRegressor(n_estimators=150, n_jobs=-1)
scores = []
model.fit(X_train, y_train)
scores.append(model.score(X_test, y_test))
score = model.score(X_test, y_test)
print(round(score*100,2))'''
```

Out[42]:
```
'#Fitting the model RandomForestRegressor\nmodel = RandomForestRegressor(n_estimators=150, n_jobs=-1)
\nscores = []\nmodel.fit(X_train, y_train)\nscores.append(model.score(X_test, y_test))\nscore = model.
score(X_test, y_test)\nprint(round(score*100,2))'
```

In [43]:
```
y_pred = pipe_rfr.predict(X_test)
y_pred
```

Out[43]:
```
array([ 0.33333333,  0.        , -0.14666667, ...,  0.        ,
        0.        ,  0.        ])
```

In [44]:
```
# Compute and print R^2 and RMSE
print("R^2: {}".format(pipe_rfr.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: 0.879338510635336
Root Mean Squared Error: 12.674922984691188
```

XGBoost

In [45]:
```
pipe_xgb = Pipeline([
    ('scaler', StandardScaler()),
    ('regression', GradientBoostingRegressor(loss="ls"))
])
pipe_xgb.fit(X_train, y_train)
#MaxAbsScaler dene
```

Out[45]:
```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('regression',
                 GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,
                                           criterion='friedman_mse', init=None,
                                           learning_rate=0.1, loss='ls',
                                           max_depth=3, max_features=None,
                                           max_leaf_nodes=None,
                                           min_impurity_decrease=0.0,
                                           min_impurity_split=None,
                                           min_samples_leaf=1,
                                           min_samples_split=2,
                                           min_weight_fraction_leaf=0.0,
                                           n_estimators=100,
                                           n_iter_no_change=None,
                                           presort='deprecated',
                                           random_state=None, subsample=1.0,
                                           tol=0.0001, validation_fraction=0.1,
                                           verbose=0, warm_start=False))],
         verbose=False)
```

In [46]:
```
'''#Fitting the model GradientBoostingRegressor
model = GradientBoostingRegressor(loss="ls")
scores = []
model.fit(X_train, y_train)
scores.append(model.score(X_test, y_test))
score = model.score(X_test, y_test)
print(round(score*100,2))'''
```

Out[46]:
```
'#Fitting the model GradientBoostingRegressor\nmodel = GradientBoostingRegressor(loss="ls")\nscores =
[]\nmodel.fit(X_train, y_train)\nscores.append(model.score(X_test, y_test))\nscore = model.score(X_tes
t, y_test)\nprint(round(score*100,2))'
```

In [47]:
```
y_pred = pipe_xgb.predict(X_test)
y_pred
```

Out[47]:
```
array([ 0.71492233, -0.0158903 ,  0.71492233, ..., -0.14123424,
        0.64018664, -0.05180918])
```

In [48]:
```
# Compute and print R^2 and RMSE
print("R^2: {}".format(pipe_xgb.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: 0.7787578474837777
Root Mean Squared Error: 17.16306810873646
```

In [49]:
```
#print(r2_score(y_test,model.predict(X_test)))
```

In [ ]:
```

```

Lasso Regression

In [50]:
```
pipe_lasso = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', Lasso(alpha=0.4, normalize=True))
])
pipe_lasso.fit(X_train, y_train)
#MaxAbsScaler dene
```

Out[50]:
```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('lasso',
                 Lasso(alpha=0.4, copy_X=True, fit_intercept=True,
                       max_iter=1000, normalize=True, positive=False,
                       precompute=False, random_state=None, selection='cyclic',
                       tol=0.0001, warm_start=False))],
         verbose=False)
```

In [51]:
```
y_pred = pipe_lasso.predict(X_test)
y_pred
```

Out[51]:
```
array([1.97824991, 1.97824991, 1.97824991, ..., 1.97824991, 1.97824991,
       1.97824991])
```

In [52]:
```
# Compute and print R^2 and RMSE
print("R^2: {}".format(pipe_lasso.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: -4.398674101846467e-06
Root Mean Squared Error: 36.48899848969968
```

In [ ]:
```

```

Ridge Regression

In [53]:
```
# Setup the array of alphas and lists to store scores
alpha_space = np.logspace(-4, 0, 50)
pipe_ridge_scores = []
pipe_ridge_scores_std = []
```

In [54]:
```
def display_plot(cv_scores, cv_scores_std):
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.plot(alpha_space, cv_scores)

    std_error = cv_scores_std / np.sqrt(10)

    ax.fill_between(alpha_space, cv_scores + std_error, cv_scores - std_error, alpha=0.2)
    ax.set_ylabel('CV Score +/- Std Error')
    ax.set_xlabel('Alpha')
    ax.axhline(np.max(cv_scores), linestyle='--', color='.5')
    ax.set_xlim([alpha_space[0], alpha_space[-1]])
    ax.set_xscale('log')
    plt.show()
```

In [ ]:
```
'''# Compute scores over range of alphas
for alpha in alpha_space:

    # Specify the alpha value to use: ridge.alpha
    pipe_ridge.alpha = alpha

    # Perform 10-fold CV: ridge_cv_scores
    pipe_ridge_cv_scores = cross_val_score(pipe_ridge, X_train, y_train, cv=10)

    # Append the mean of ridge_cv_scores to ridge_scores
    pipe_ridge_scores.append(np.mean(pipe_ridge_cv_scores))

    # Append the std of ridge_cv_scores to ridge_scores_std
    pipe_ridge_scores_std.append(np.std(pipe_ridge_cv_scores))

# Display the plot
display_plot(pipe_ridge_scores, pipe_ridge_scores_std)'''
```

In [56]:
```
pipe_ridge = Pipeline([
    ('scaler', StandardScaler()),
    ('ridge', Ridge(normalize=True))
])
pipe_ridge.fit(X_train, y_train)
#MaxAbsScaler dene
```

Out[56]:
```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('ridge',
                 Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
                       max_iter=None, normalize=True, random_state=None,
                       solver='auto', tol=0.001))],
         verbose=False)
```

In [57]:
```
y_pred = pipe_ridge.predict(X_test)
y_pred
```

Out[57]:
```
array([ 2.61109922, -0.65862268,  2.47545918, ..., -2.44094617,
        1.31116982, -1.4480018 ])
```

In [58]:
```
# Compute and print R^2 and RMSE
print("R^2: {}".format(pipe_ridge.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: 0.06232292278505036
Root Mean Squared Error: 35.3335796369288
```

In [ ]:
```

```

Model Tuning

In [59]:
```
max_range =10
```

In [60]:
```
'''for i in range(max_range):
    X_train,X_test,y_train,y_test = train_test_split(predictors,target,test_size =0.3,random_state =i)
    model = RandomForestRegressor(n_estimators=150)
    model.fit(X_train,y_train)
    print("Random state {}\n".format(i))
    print(r2_score(y_test,model.predict(X_test)))'''
```

Out[60]:
```
'for i in range(max_range):\n    X_train,X_test,y_train,y_test = train_test_split(predictors,target,te
st_size =0.3,random_state =i)\n    model = RandomForestRegressor(n_estimators=150)\n    model.fit(X_tr
ain,y_train)\n    print("Random state {}\n".format(i))\n    print(r2_score(y_test,model.predict(X_tes
t)))'
```

In [ ]:
```

```

In [61]:
```
'''param_grid = {'n_estimators':np.arange(50, 400, 100)}
clf = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, verbose=1, n_jobs=-1)
clf.fit(X_train, y_train)
print("Best Parameters: {}".format(clf.best_params_))
print("Best score is {}".format(clf.best_score_))'''
```

Out[61]:
```
'param_grid = {\'n_estimators\':np.arange(50, 400, 100)}\nclf = GridSearchCV(estimator=model, param_gr
id=param_grid, cv=3, verbose=1, n_jobs=-1)\nclf.fit(X_train, y_train)\nprint("Best Parameters: {}".for
mat(clf.best_params_)) \nprint("Best score is {}".format(clf.best_score_))'
```

In [ ]:

In [ ]:

Output

In [62]:
```
y_pred.shape
```

Out[62]:
```
(198009,)
```

In [63]:
```
y_pred = pipe_rfr.predict(test)

pred_list = [int(x) for x in y_pred]

output = pd.DataFrame({'Id': test.index, 'TargetValue': pred_list})
print(output)
```

```
            Id  TargetValue
0        21800            3
1        21801            0
2        21802            3
3        21803            0
4        21804            3
...        ...          ...
220005  309415            0
220006  309416            0
220007  309417            0
220008  309418            0
220009  309419            0

[220010 rows x 2 columns]
```

In [64]:
```
output
```

Out[64]:

	Id	TargetValue
0	21800	3
1	21801	0
2	21802	3
3	21803	0
4	21804	3
...	...	...
220005	309415	0
220006	309416	0
220007	309417	0
220008	309418	0
220009	309419	0

220010 rows × 2 columns

In [ ]:
```
```

Preparing Submission File

In [65]:
```
output.reset_index()
output
```

Out[65]:

	Id	TargetValue
0	21800	3
1	21801	0
2	21802	3
3	21803	0
4	21804	3
...	...	...
220005	309415	0
220006	309416	0
220007	309417	0
220008	309418	0
220009	309419	0

220010 rows × 2 columns

In [66]:
```
output['Id'] = output['Id'].apply(lambda x: x - 21799)
output
```

Out[66]:

	Id	TargetValue
0	1	3
1	2	0
2	3	3
3	4	0
4	5	3
...	...	...
220005	287616	0
220006	287617	0
220007	287618	0
220008	287619	0
220009	287620	0

220010 rows × 2 columns

In [67]:
```
result
```

Out[67]:

	ForecastId	County	Province_State	Country_Region	Population	Weight	Date	Target
21800	21801	Autauga	Alabama	US	55869	0.091485	2020-05-07	ConfirmedCases
21801	21802	Autauga	Alabama	US	55869	0.914848	2020-05-07	Fatalities
21802	21803	Autauga	Alabama	US	55869	0.091485	2020-05-08	ConfirmedCases
21803	21804	Autauga	Alabama	US	55869	0.914848	2020-05-08	Fatalities
21804	21805	Autauga	Alabama	US	55869	0.091485	2020-05-09	ConfirmedCases
...	...	...	...	...	...	...	...	...
309415	309416	Weston	Wyoming	US	6927	1.130796	2020-06-08	Fatalities
309416	309417	Weston	Wyoming	US	6927	0.113080	2020-06-09	ConfirmedCases
309417	309418	Weston	Wyoming	US	6927	1.130796	2020-06-09	Fatalities
309418	309419	Weston	Wyoming	US	6927	0.113080	2020-06-10	ConfirmedCases
309419	309420	Weston	Wyoming	US	6927	1.130796	2020-06-10	Fatalities

220010 rows × 8 columns

In [68]:
```
result['Id'] = result['ForecastId'].apply(lambda x: x - 21800)
result = result.drop(['ForecastId'], axis=1)
```

In [69]:
```
final = pd.merge(result, output, on='Id',  how='left')
final
```

Out[69]:

	County	Province_State	Country_Region	Population	Weight	Date	Target	Id	TargetValue
0	Autauga	Alabama	US	55869	0.091485	2020-05-07	ConfirmedCases	1	3
1	Autauga	Alabama	US	55869	0.914848	2020-05-07	Fatalities	2	0
2	Autauga	Alabama	US	55869	0.091485	2020-05-08	ConfirmedCases	3	3
3	Autauga	Alabama	US	55869	0.914848	2020-05-08	Fatalities	4	0
4	Autauga	Alabama	US	55869	0.091485	2020-05-09	ConfirmedCases	5	3
...	...	...	...	...	...	...	...	...	...
220005	Weston	Wyoming	US	6927	1.130796	2020-06-08	Fatalities	287616	0
220006	Weston	Wyoming	US	6927	0.113080	2020-06-09	ConfirmedCases	287617	0
220007	Weston	Wyoming	US	6927	1.130796	2020-06-09	Fatalities	287618	0
220008	Weston	Wyoming	US	6927	0.113080	2020-06-10	ConfirmedCases	287619	0
220009	Weston	Wyoming	US	6927	1.130796	2020-06-10	Fatalities	287620	0

220010 rows × 9 columns

In [70]:
```
final = final[['Id', 'Country_Region', 'Weight', 'Date', 'Target', 'TargetValue', 'County', 'Population'
final
```

Out[70]:

	Id	Country_Region	Weight	Date	Target	TargetValue	County	Population	Province_State
0	1	US	0.091485	2020-05-07	ConfirmedCases	3	Autauga	55869	Alabama
1	2	US	0.914848	2020-05-07	Fatalities	0	Autauga	55869	Alabama
2	3	US	0.091485	2020-05-08	ConfirmedCases	3	Autauga	55869	Alabama
3	4	US	0.914848	2020-05-08	Fatalities	0	Autauga	55869	Alabama
4	5	US	0.091485	2020-05-09	ConfirmedCases	3	Autauga	55869	Alabama
...	...	...	...	...	...	...	...	...	...
220005	287616	US	1.130796	2020-06-08	Fatalities	0	Weston	6927	Wyoming
220006	287617	US	0.113080	2020-06-09	ConfirmedCases	0	Weston	6927	Wyoming
220007	287618	US	1.130796	2020-06-09	Fatalities	0	Weston	6927	Wyoming
220008	287619	US	0.113080	2020-06-10	ConfirmedCases	0	Weston	6927	Wyoming
220009	287620	US	1.130796	2020-06-10	Fatalities	0	Weston	6927	Wyoming

220010 rows × 9 columns

In [71]:
```
print(final[final.County == 'Denton'][final.Date == '2020-06-10'])
```

```
            Id Country_Region    Weight        Date          Target  \
181018  236769             US  0.073015  2020-06-10  ConfirmedCases
181019  236770             US  0.730149  2020-06-10      Fatalities

        TargetValue  County  Population Province_State
181018           23  Denton      887207          Texas
181019            0  Denton      887207          Texas

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

In [72]:
```
print(final[final.Province_State == 'Texas'][final.Date == '2020-06-10'])
```

```
            Id Country_Region    Weight        Date          Target  \
176748  231279             US  0.091211  2020-06-10  ConfirmedCases
176749  231280             US  0.912106  2020-06-10      Fatalities
176818  231369             US  0.101661  2020-06-10  ConfirmedCases
176819  231370             US  1.016611  2020-06-10      Fatalities
176888  231459             US  0.087948  2020-06-10  ConfirmedCases
...        ...            ...       ...         ...             ...
194319  253870             US  1.020540  2020-06-10      Fatalities
194388  253959             US  0.104607  2020-06-10  ConfirmedCases
194389  253960             US  1.046070  2020-06-10      Fatalities
194458  254049             US  0.106617  2020-06-10  ConfirmedCases
194459  254050             US  1.066175  2020-06-10      Fatalities

        TargetValue    County  Population Province_State
176748            2  Anderson       57735          Texas
176749            0  Anderson       57735          Texas
176818            0   Andrews       18705          Texas
176819            0   Andrews       18705          Texas
176888           14  Angelina       86715          Texas
...             ...       ...         ...            ...
194319            0     Young       18010          Texas
194388            0    Zapata       14179          Texas
194389            0    Zapata       14179          Texas
194458            0    Zavala       11840          Texas
194459            0    Zavala       11840          Texas

[508 rows x 9 columns]

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

In [73]:
```
final_grouped_county = final.groupby(['Province_State', 'County', 'Target']).TargetValue.sum()
final_grouped_county = final_grouped_county.to_frame().reset_index()
final_grouped_county
```

Out[73]:

	Province_State	County	Target	TargetValue
0	Alabama	Autauga	ConfirmedCases	105
1	Alabama	Autauga	Fatalities	0
2	Alabama	Baldwin	ConfirmedCases	70
3	Alabama	Baldwin	Fatalities	0
4	Alabama	Barbour	ConfirmedCases	0
...	...	...	...	...
6281	Wyoming	Uinta	Fatalities	0
6282	Wyoming	Washakie	ConfirmedCases	0
6283	Wyoming	Washakie	Fatalities	0
6284	Wyoming	Weston	ConfirmedCases	0
6285	Wyoming	Weston	Fatalities	0

6286 rows × 4 columns

In [74]:
```
final_grouped_county[final_grouped_county.Province_State == 'Texas'][final_grouped_county.County == 'Dal
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

Out[74]:

	Province_State	County	Target	TargetValue
5160	Texas	Dallas	ConfirmedCases	8540
5161	Texas	Dallas	Fatalities	105

In [75]:
```
final_grouped_county[final_grouped_county.Province_State == 'New York']
```

Out[75]:

	Province_State	County	Target	TargetValue
3658	New York	Albany	ConfirmedCases	980
3659	New York	Albany	Fatalities	105
3660	New York	Allegany	ConfirmedCases	0
3661	New York	Allegany	Fatalities	0
3662	New York	Bronx	ConfirmedCases	0
...	...	...	...	...
3777	New York	Westchester	Fatalities	770
3778	New York	Wyoming	ConfirmedCases	35
3779	New York	Wyoming	Fatalities	0
3780	New York	Yates	ConfirmedCases	0
3781	New York	Yates	Fatalities	0

124 rows × 4 columns

In [76]:
```
final_grouped_state = final.groupby(['Province_State', 'Target']).TargetValue.sum()
final_grouped_state = final_grouped_state.to_frame().reset_index()
final_grouped_state
```

Out[76]:

	Province_State	Target	TargetValue
0	Alabama	ConfirmedCases	9940
1	Alabama	Fatalities	420
2	Alaska	ConfirmedCases	175
3	Alaska	Fatalities	0
4	Arizona	ConfirmedCases	14000
...	...	...	...
97	West Virginia	Fatalities	0
98	Wisconsin	ConfirmedCases	10290
99	Wisconsin	Fatalities	140
100	Wyoming	ConfirmedCases	875
101	Wyoming	Fatalities	0

102 rows × 3 columns

In [77]:
```
final.to_csv('final.csv', index=False)
```

In [78]:
```
final_grouped_county.to_csv('final_grouped_county.csv', index=False)
```

In [79]:
```
final_grouped_state.to_csv('final_grouped_state.csv', index=False)
```

In [ ]:
```

```

In [ ]:
```
'''last_date_final=final.Date.max()
df2=final[final["Date"]==last_date_final]
df2'''
```

In [81]:
```
final
```

Out[81]:

	Id	Country_Region	Weight	Date	Target	TargetValue	County	Population	Province_State
0	1	US	0.091485	2020-05-07	ConfirmedCases	3	Autauga	55869	Alabama
1	2	US	0.914848	2020-05-07	Fatalities	0	Autauga	55869	Alabama
2	3	US	0.091485	2020-05-08	ConfirmedCases	3	Autauga	55869	Alabama
3	4	US	0.914848	2020-05-08	Fatalities	0	Autauga	55869	Alabama
4	5	US	0.091485	2020-05-09	ConfirmedCases	3	Autauga	55869	Alabama
...	...	...	...	...	...	...	...	...	...
220005	287616	US	1.130796	2020-06-08	Fatalities	0	Weston	6927	Wyoming
220006	287617	US	0.113080	2020-06-09	ConfirmedCases	0	Weston	6927	Wyoming
220007	287618	US	1.130796	2020-06-09	Fatalities	0	Weston	6927	Wyoming
220008	287619	US	0.113080	2020-06-10	ConfirmedCases	0	Weston	6927	Wyoming
220009	287620	US	1.130796	2020-06-10	Fatalities	0	Weston	6927	Wyoming

220010 rows × 9 columns

In [82]:
```
df2=final.groupby(by=["Province_State", "Date"],as_index=False)["TargetValue"].sum()
df2
```

Out[82]:

	Province_State	Date	TargetValue
0	Alabama	2020-05-07	296
1	Alabama	2020-05-08	296
2	Alabama	2020-05-09	296
3	Alabama	2020-05-10	296
4	Alabama	2020-05-11	296
...	...	...	...
1780	Wyoming	2020-06-06	25
1781	Wyoming	2020-06-07	25
1782	Wyoming	2020-06-08	25
1783	Wyoming	2020-06-09	25
1784	Wyoming	2020-06-10	25

1785 rows × 3 columns

In [83]:
```
states_final=df2.nlargest(5,"TargetValue")
states_final
```

Out[83]:

	Province_State	Date	TargetValue
1120	New York	2020-05-07	3365
1121	New York	2020-05-08	3365
1122	New York	2020-05-09	3365
1123	New York	2020-05-10	3365
1124	New York	2020-05-11	3365

In [84]:
```
cases_final=final.groupby(by=["Date","Province_State"],as_index=False)["TargetValue"].sum()
cases_final
```

Out[84]:

	Date	Province_State	TargetValue
0	2020-05-07	Alabama	296
1	2020-05-07	Alaska	5
2	2020-05-07	Arizona	421
3	2020-05-07	Arkansas	90
4	2020-05-07	California	2583
...	...	...	...
1780	2020-06-10	Virginia	380
1781	2020-06-10	Washington	264
1782	2020-06-10	West Virginia	4
1783	2020-06-10	Wisconsin	298
1784	2020-06-10	Wyoming	25

1785 rows × 3 columns

In [85]:
```
cases_final=cases_final.merge(states_final,on="Province_State")
cases_final
```

Out[85]:

	Date_x	Province_State	TargetValue_x	Date_y	TargetValue_y
0	2020-05-07	New York	3365	2020-05-07	3365
1	2020-05-07	New York	3365	2020-05-08	3365
2	2020-05-07	New York	3365	2020-05-09	3365
3	2020-05-07	New York	3365	2020-05-10	3365
4	2020-05-07	New York	3365	2020-05-11	3365
...	...	...	...	...	...
170	2020-06-10	New York	3365	2020-05-07	3365
171	2020-06-10	New York	3365	2020-05-08	3365
172	2020-06-10	New York	3365	2020-05-09	3365
173	2020-06-10	New York	3365	2020-05-10	3365
174	2020-06-10	New York	3365	2020-05-11	3365

175 rows × 5 columns