```
import pandas as pd
# Read data.  This data represents the cumulative known cases to date (https://covidtracking.com/about-data/faq)
url = 'https://raw.githubusercontent.com/COVID19Tracking/covid-tracking-data/master/data/states_daily_4pm_et.csv'
df = pd.read_csv(url,index_col=0,parse_dates=[0])
```

```
df.head(5)
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	inIcuCurrently	inIcuCumulative	onVentilatorCu
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	NaN	NaN	
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	403.0	
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	NaN	NaN	
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	NaN	
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	282.0	NaN	

Double-click (or enter) to edit

```
# Drop total, posNeg, and hospitalized columns as they are redundant
# Drop other columns that will not be used
#df_drop = df.drop(columns = [6, 7, 8, 9, 11, 12, 14, 15, 17, 18, 19, 20, 21, 22, 23])
df_drop = df.drop(columns = ['inIcuCurrently', 'inIcuCumulative',
                             'onVentilatorCurrently', 'onVentilatorCumulative',
                             'hash', 'dateChecked', 'hospitalized', 'total',
                             'posNeg', 'fips', 'deathIncrease',
                             'hospitalizedIncrease', 'negativeIncrease',
                             'positiveIncrease', 'totalTestResultsIncrease'])
df_drop.head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0

```
# Create new features
# Divide positive by totalTestResults to get positive_percent
df_drop["percent_positive"] = ""
df_drop["percent_positive"] = 100*df_drop["positive"]/df_drop["totalTestResults"]
df_drop.head()
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	

```
# Divide hospitalized by positive to get hospitalized_percent
import numpy as np
df_drop["hospitalized_percent"] = ""
df_drop["hospitalized_percent"] = np.nanmax(df_drop[['hospitalizedCurrently','hospitalizedCumulative']], axis=1)
df_drop["hospitalized_percent"] = 100*df_drop["hospitalized_percent"]/df_drop["positive"]
df_drop.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: RuntimeWarning: All-NaN axis encountered
  This is separate from the ipykernel package so we can avoid doing imports until
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Divide recovered by positive to get recovered_percent
df_drop["recovered_percent"] = ""
df_drop["recovered_percent"] = 100*df_drop["recovered"]/df_drop["positive"]
df_drop.head()
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Divide death by positive to get death_percent
df_drop["death_percent"] = ""
df_drop["death_percent"] = 100*df_drop["death"]/df_drop["positive"]
df_drop.head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Fetch the latest state population data (nst-est2019-01.csv)
from google.colab import files
uploaded = files.upload()
```

Choose Files nst-est2019-01.csv

- **nst-est2019-01.csv**(application/vnd.ms-excel) - 676 bytes, last modified: 4/13/2020 - 100% done

```
Saving nst-est2019-01.csv to nst-est2019-01.csv
```

```
# Load latest state population data
import io
df_state_pop = pd.read_csv(io.StringIO(uploaded['nst-est2019-01.csv'].decode('utf-8')))
df_state_pop["Population"] = pd.to_numeric(df_state_pop["Population"])
df_state_pop.head()
```

	State	Population
0	AK	731545.0
1	AL	4903185.0
2	AR	3017804.0
3	AS	NaN
4	AZ	7278717.0

```
# Add column of state populations (population) to df_drop_total_posNeg
# Need to sort rows by state using index numbering from state_list

df_drop["population"] = ""

for i in range(len(df_drop)):
  for index in range(len(df_state_pop)):
    if df_drop.iloc[i, 0] == df_state_pop.iloc[index, 0]:
      df_drop.iloc[i, 13] = df_state_pop.iloc[index, 1]

df_drop[["population"]] = df_drop["population"].apply(pd.to_numeric)

df_drop.head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Normalize positive to state population
df_drop["positive_norm"] = ""
df_drop["positive_norm"] = df_drop["positive"]/df_drop["population"]
df_drop.head()
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos:
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7(
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0(
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Normalize hospitalized to state population
df_drop["hospitalized_norm"] = ""
df_drop["hospitalized_norm"] = np.nanmax(df_drop[['hospitalizedCurrently','hospitalizedCumulative']], axis=1)
df_drop["hospitalized_norm"] = df_drop["hospitalized_norm"]/df_drop["population"]
df_drop.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: RuntimeWarning: All-NaN axis encountered
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos:
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7(
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0(
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Normalize recovered to state population
df_drop["recovered_norm"] = ""
df_drop["recovered_norm"] = df_drop["recovered"]/df_drop["population"]
df_drop.head()
```

	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos:
date										
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7(
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0(
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
# Normalize death to state population
df_drop["death_norm"] = ""
df_drop["death_norm"] = df_drop["death"]/df_drop["population"]
df_drop.head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.7
2020-05-03	AL	7725.0	84775.0	NaN	NaN	1035.0	NaN	290.0	92500.0	8.3
2020-05-03	AR	3431.0	49459.0	NaN	100.0	427.0	1999.0	76.0	52890.0	6.4
2020-05-03	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.0
2020-05-03	AZ	8640.0	72479.0	NaN	732.0	1348.0	1597.0	362.0	81119.0	10.6

```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3321 entries, 2020-05-03 to 2020-01-22
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   state                  3321 non-null   object
 1   positive               3306 non-null   float64
 2   negative               3140 non-null   float64
 3   pending                677 non-null    float64
 4   hospitalizedCurrently  1191 non-null   float64
 5   hospitalizedCumulative 1239 non-null   float64
 6   recovered              1037 non-null   float64
 7   death                  2594 non-null   float64
 8   totalTestResults       3319 non-null   float64
 9   percent_positive       3275 non-null   float64
 10  hospitalized_percent   1870 non-null   float64
 11  recovered_percent      1037 non-null   float64
 12  death_percent          2541 non-null   float64
 13  population             3125 non-null   float64
 14  positive_norm          3125 non-null   float64
 15  hospitalized_norm      1831 non-null   float64
 16  recovered_norm         950 non-null    float64
 17  death_norm             2447 non-null   float64
dtypes: float64(17), object(1)
memory usage: 573.0+ KB
```

```
# Get the unique values of 'state' column
state_list = df.state.unique()
state_list
```

```
array(['AK', 'AL', 'AR', 'AS', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL',
       'GA', 'GU', 'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA',
       'MD', 'ME', 'MI', 'MN', 'MO', 'MP', 'MS', 'MT', 'NC', 'ND', 'NE',
       'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'PR', 'RI',
       'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VI', 'VT', 'WA', 'WI', 'WV',
       'WY'], dtype=object)
```

```
#create a data frame dictionary to store the state data frames
df_state_dict = {elem : pd.DataFrame for elem in state_list}

for key in df_state_dict.keys():
    df_state_dict[key] = df_drop[:][df_drop.state == key]
```

```
df_state_dict['AK'].head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
2020-05-03	AK	368.0	21210.0	NaN	12.0	NaN	262.0	9.0	21578.0	1.70
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.70
2020-05-01	AK	364.0	19961.0	NaN	25.0	NaN	254.0	9.0	20325.0	1.79
2020-04-30	AK	355.0	18764.0	NaN	19.0	NaN	252.0	9.0	19119.0	1.85
2020-04-29	AK	355.0	18764.0	NaN	14.0	NaN	240.0	9.0	19119.0	1.85

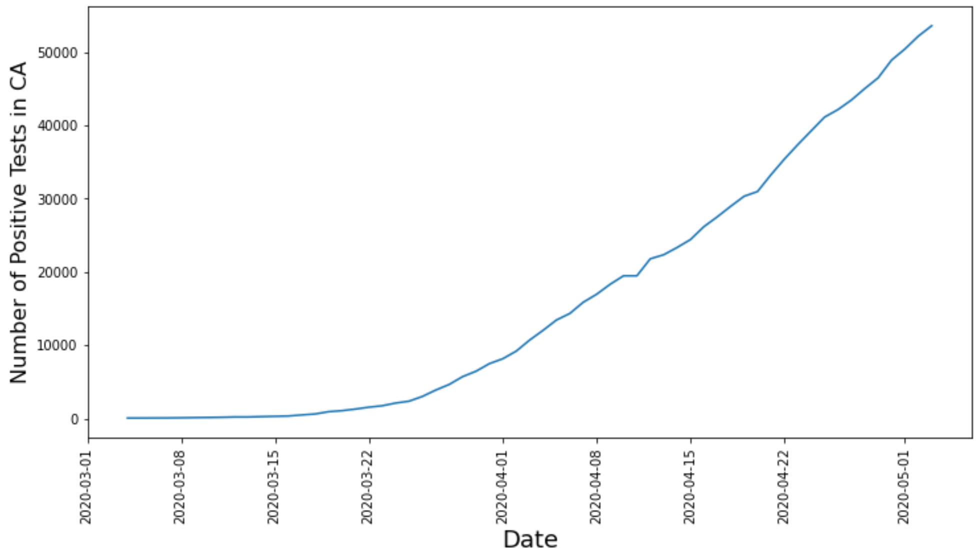
```
df_state_dict['CA'].head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_pos
2020-05-03	CA	53616.0	662135.0	NaN	4734.0	NaN	NaN	2215.0	715751.0	7.4
2020-05-02	CA	52197.0	634606.0	NaN	4722.0	NaN	NaN	2171.0	686803.0	7.5
2020-05-01	CA	50442.0	604543.0	NaN	4706.0	NaN	NaN	2073.0	654985.0	7.7
2020-04-30	CA	48917.0	576420.0	NaN	4981.0	NaN	NaN	1982.0	625337.0	7.8
2020-04-29	CA	46500.0	556639.0	NaN	5011.0	NaN	NaN	1887.0	603139.0	7.7

```
from matplotlib import pyplot as plt
```

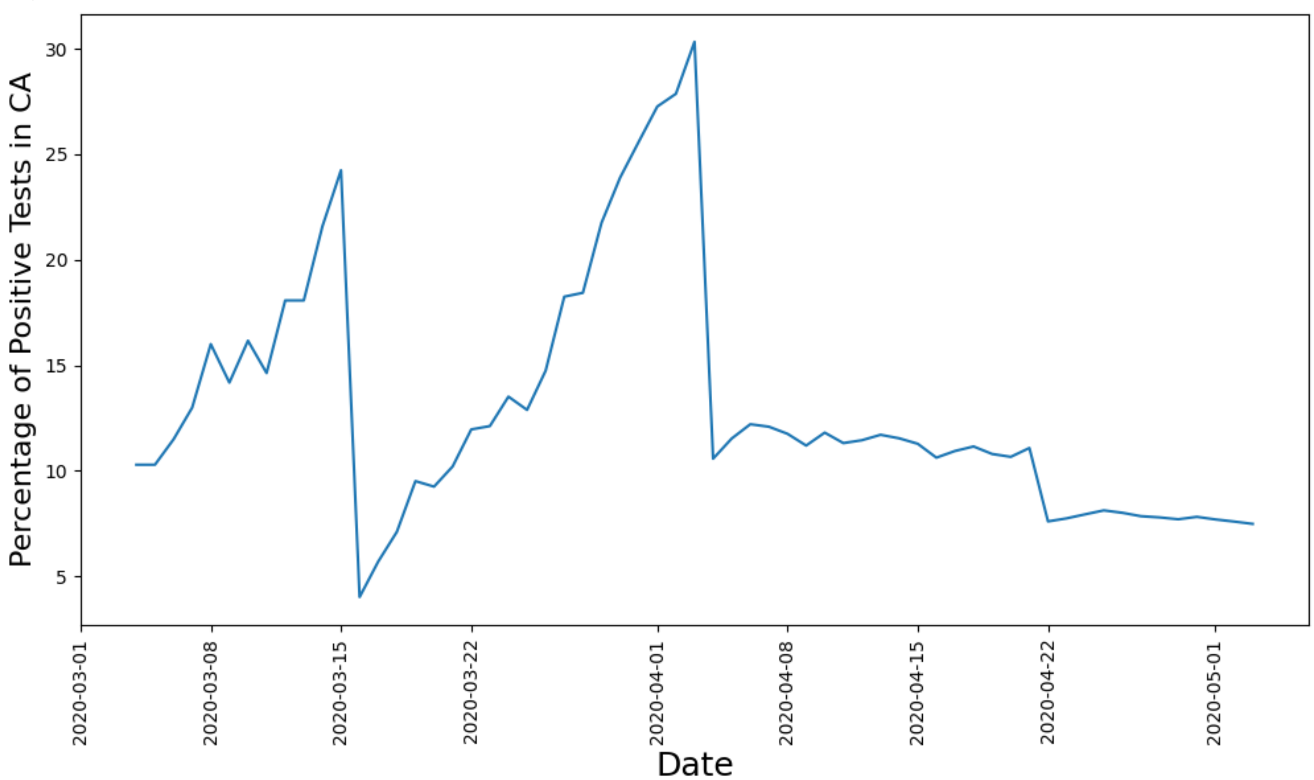
```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].positive)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Number of Positive Tests in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 432x288 with 0 Axes>
```




```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].percent_positive)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Percentage of Positive Tests in CA', fontsize=16)
plt.show()
```

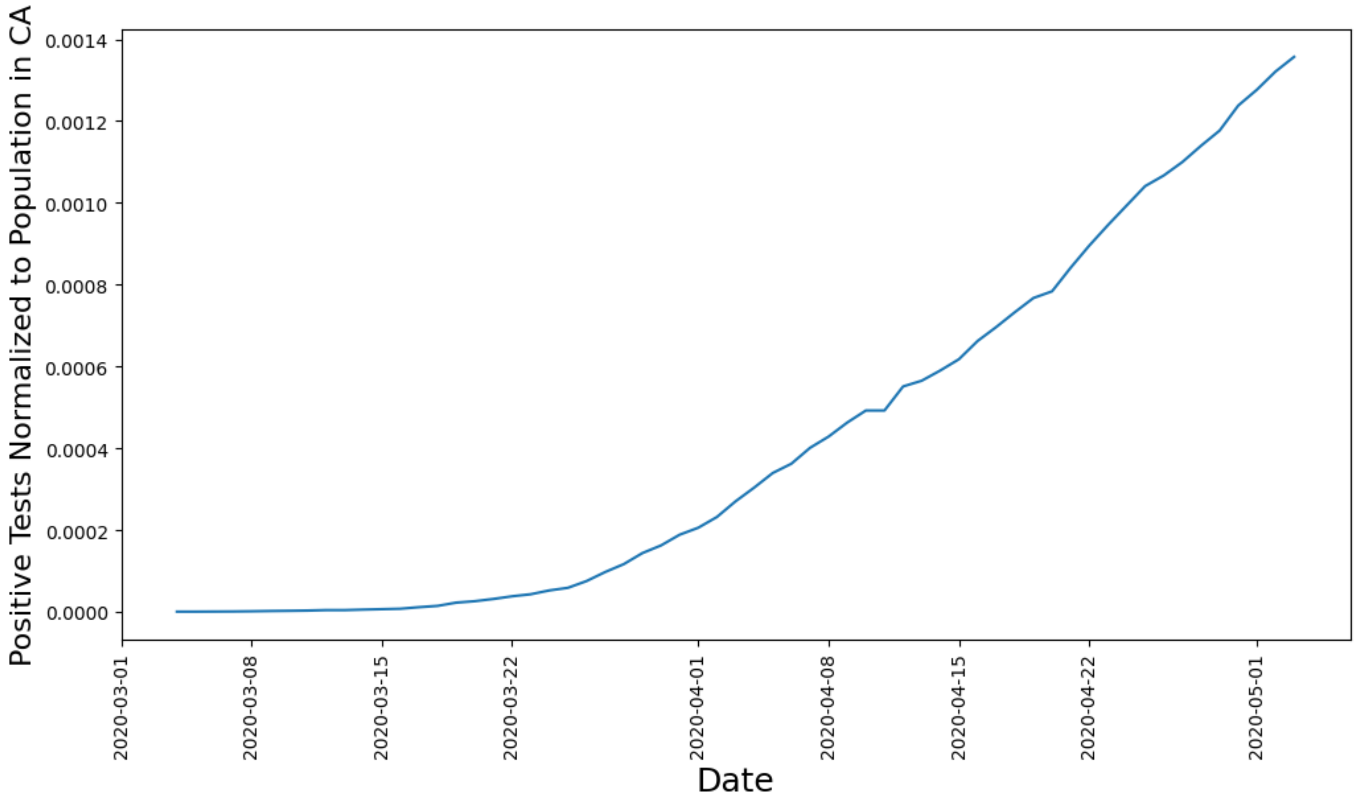
```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```




```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)
```

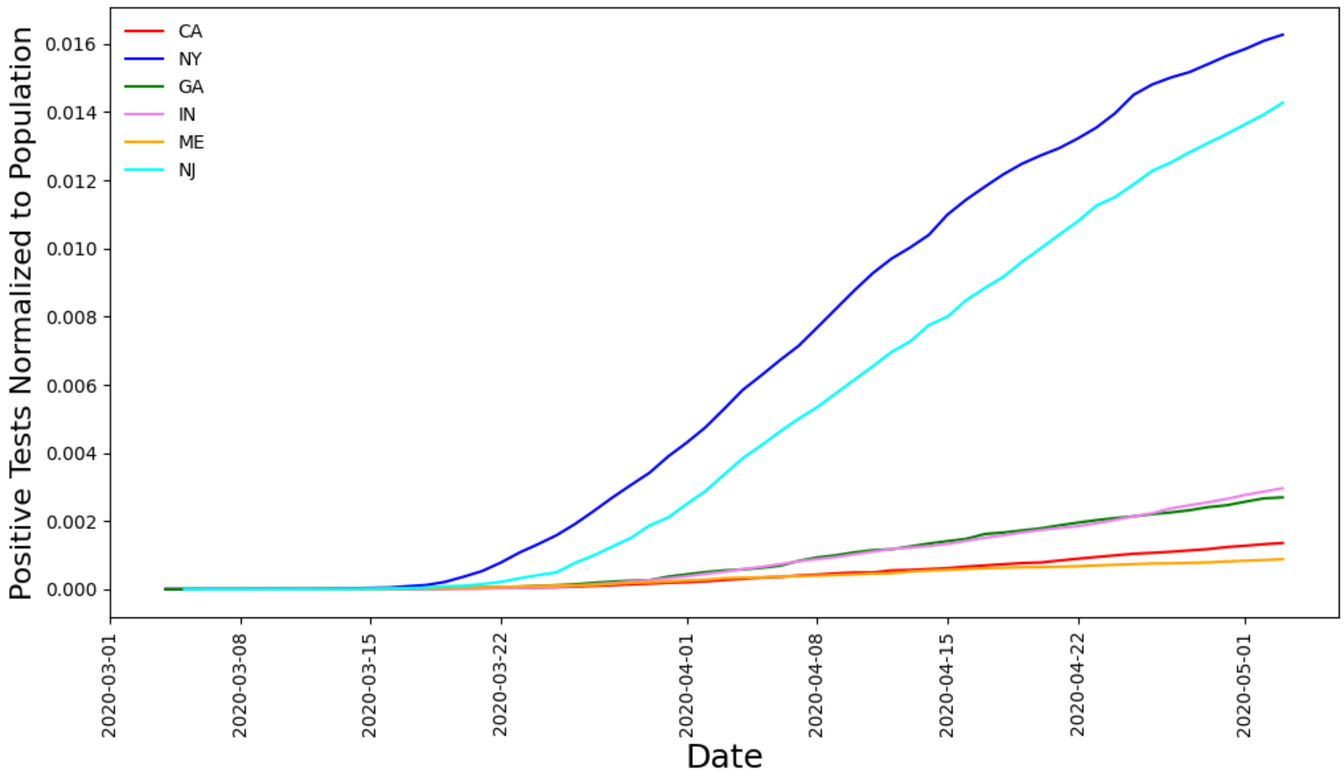
```
plt.plot(df_state_dict['CA'].positive_norm)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Positive Tests Normalized to Population in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```



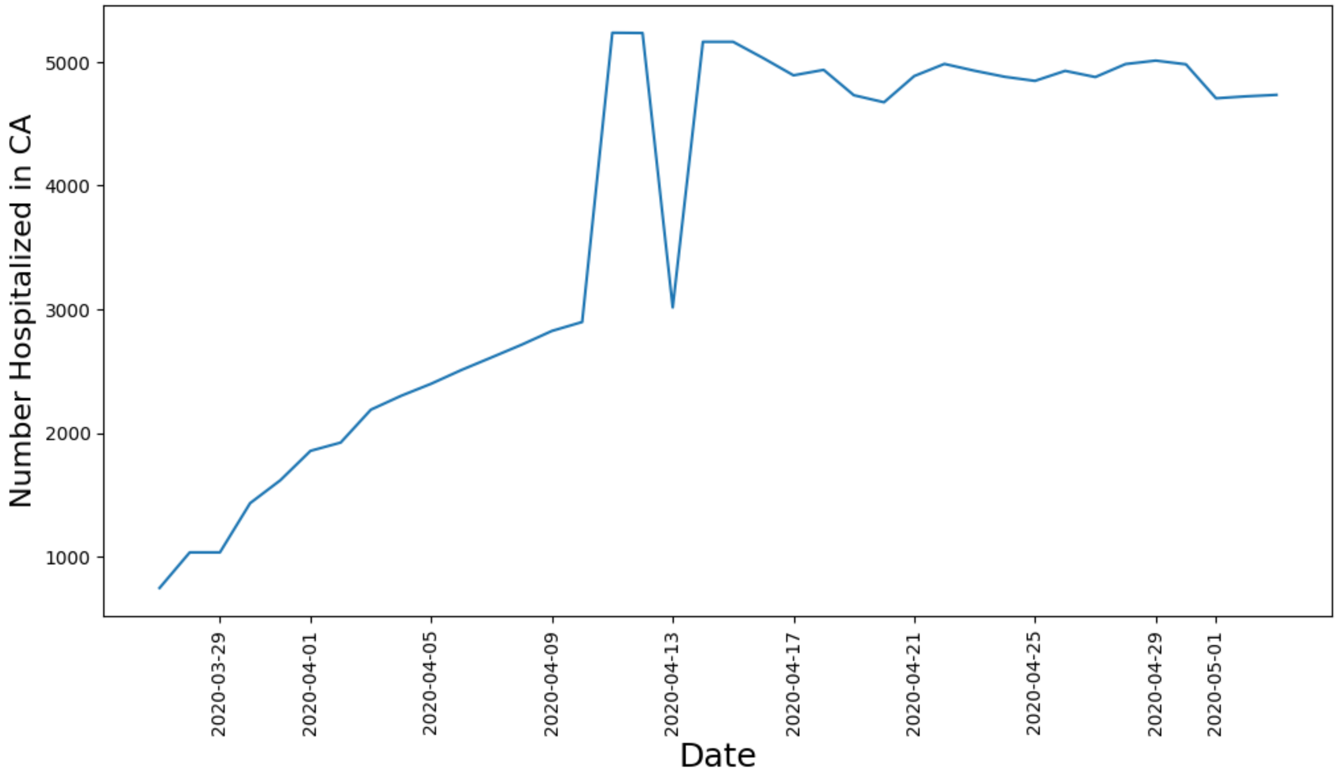

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].positive_norm, color="red", label="CA")
plt.plot(df_state_dict['NY'].positive_norm, color="blue", label="NY")
plt.plot(df_state_dict['GA'].positive_norm, color="green", label="GA")
plt.plot(df_state_dict['IN'].positive_norm, color="violet", label="IN")
plt.plot(df_state_dict['ME'].positive_norm, color="orange", label="ME")
plt.plot(df_state_dict['NJ'].positive_norm, color="cyan", label="NJ")
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Positive Tests Normalized to Population', fontsize=16)
plt.show()
```

<Figure size 640x480 with 0 Axes>



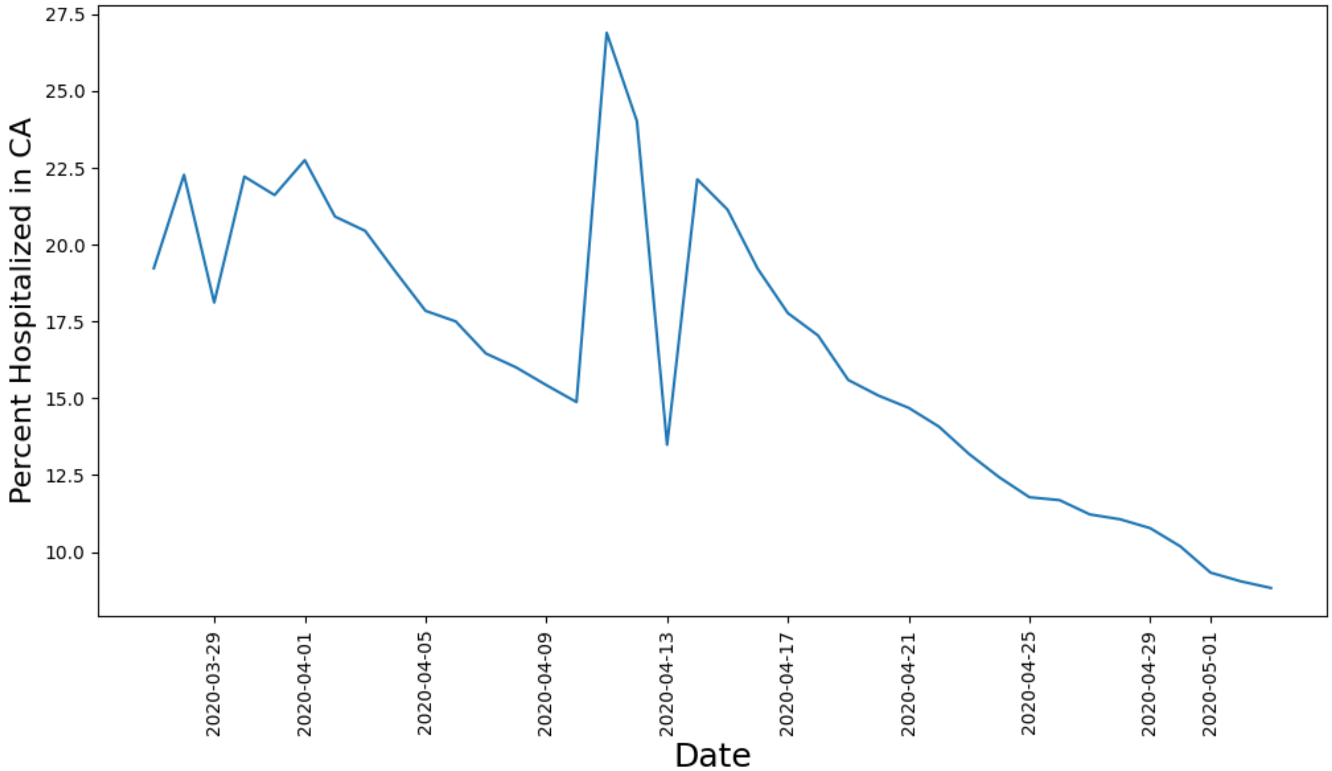

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].hospitalizedCurrently)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Number Hospitalized in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```



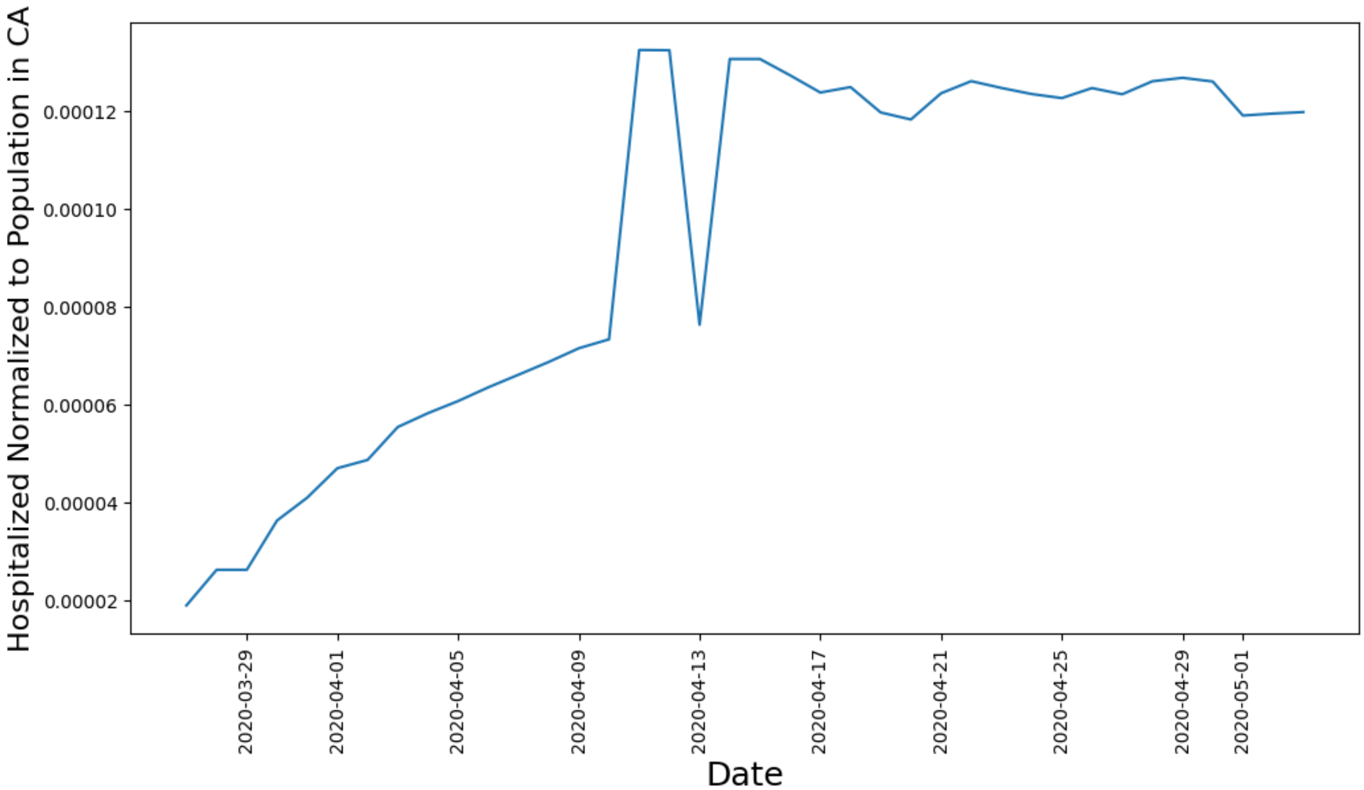

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].hospitalized_percent)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Percent Hospitalized in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```



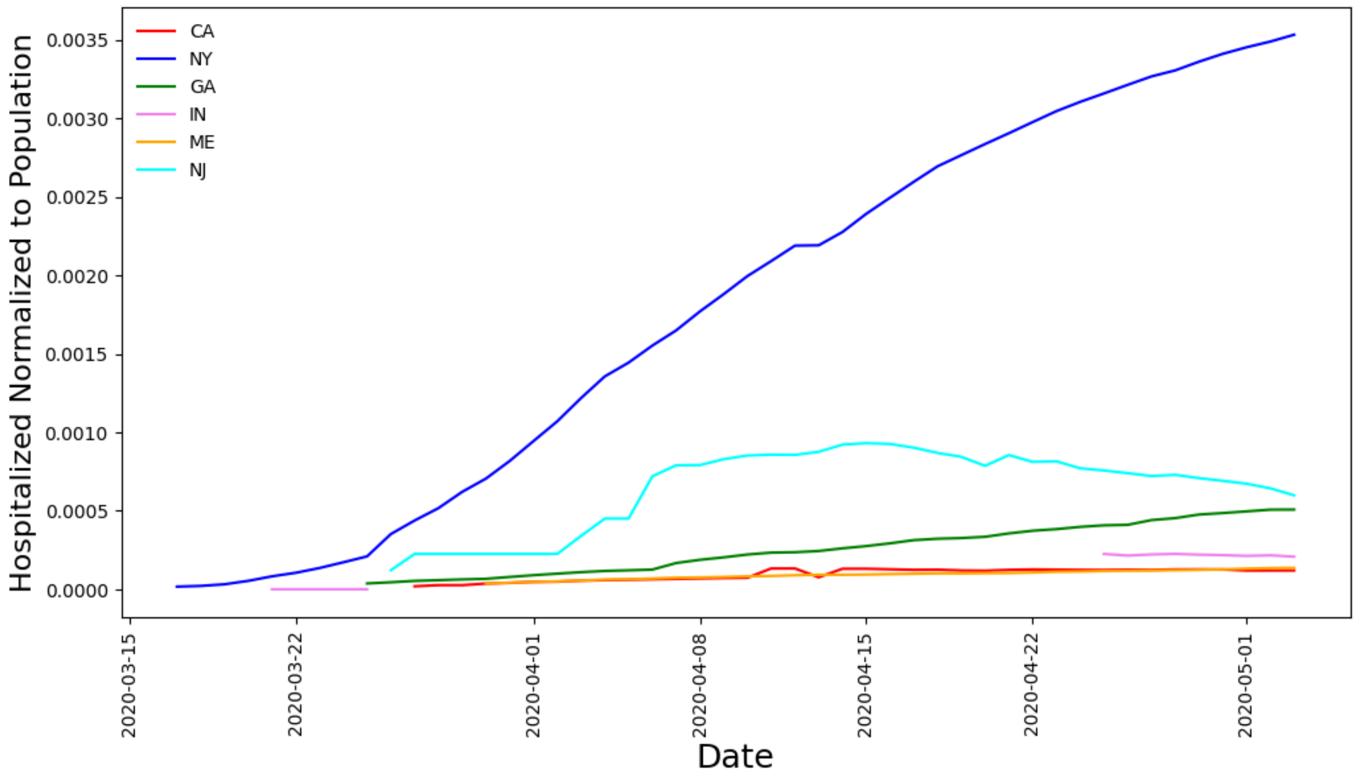

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].hospitalized_norm)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Hospitalized Normalized to Population in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```




```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].hospitalized_norm, color="red", label="CA")
plt.plot(df_state_dict['NY'].hospitalized_norm, color="blue", label="NY")
plt.plot(df_state_dict['GA'].hospitalized_norm, color="green", label="GA")
plt.plot(df_state_dict['IN'].hospitalized_norm, color="violet", label="IN")
plt.plot(df_state_dict['ME'].hospitalized_norm, color="orange", label="ME")
plt.plot(df_state_dict['NJ'].hospitalized_norm, color="cyan", label="NJ")
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Hospitalized Normalized to Population', fontsize=16)
plt.show()
```

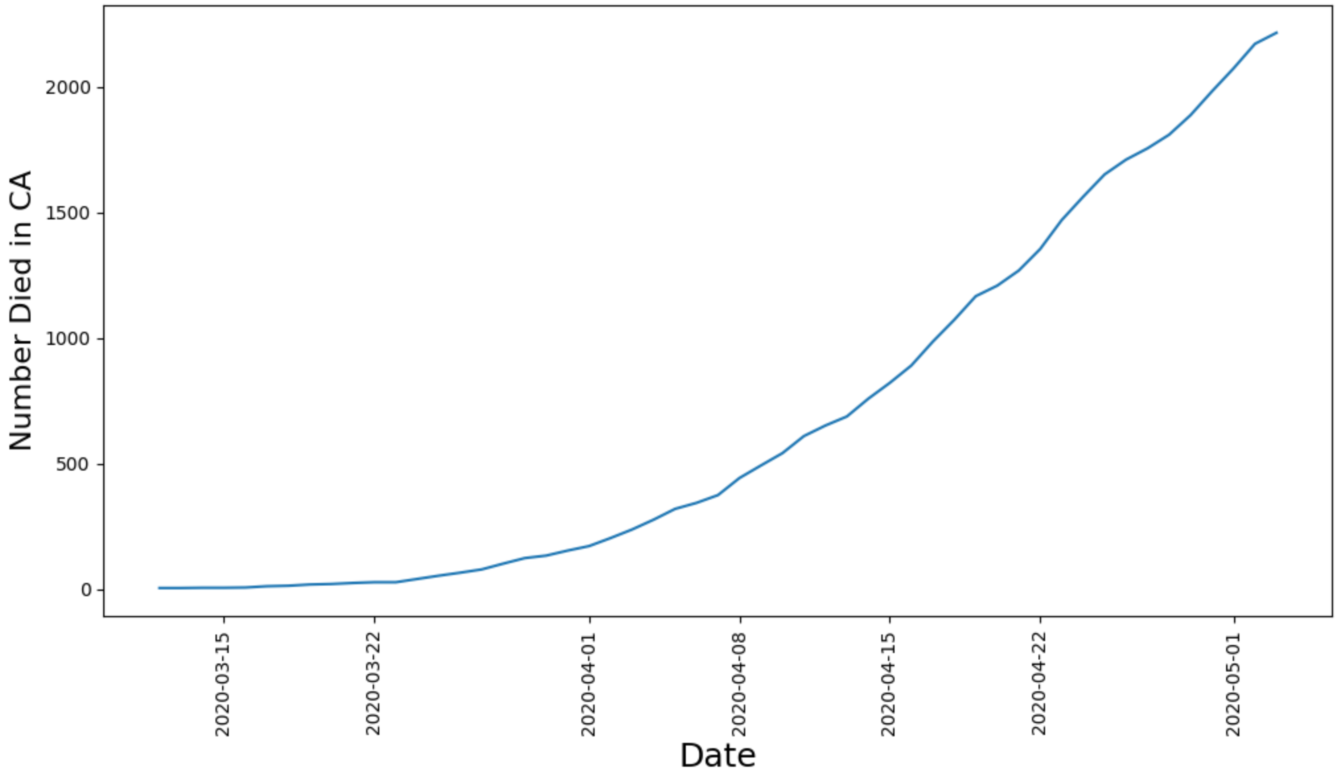
```
<Figure size 640x480 with 0 Axes>
```



In several states, population normalized hospitalizations plateau, although population normalized death rate continues to grow.

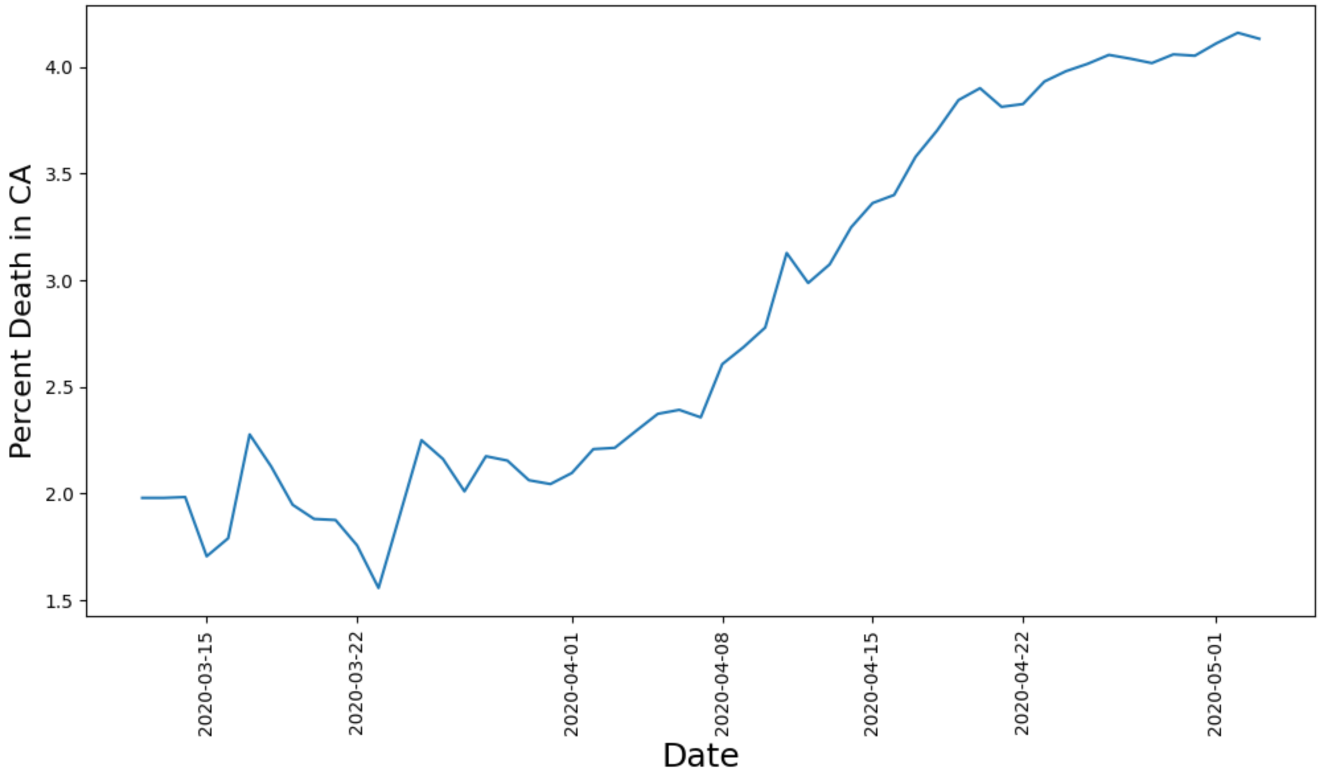
```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].death)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Number Died in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```



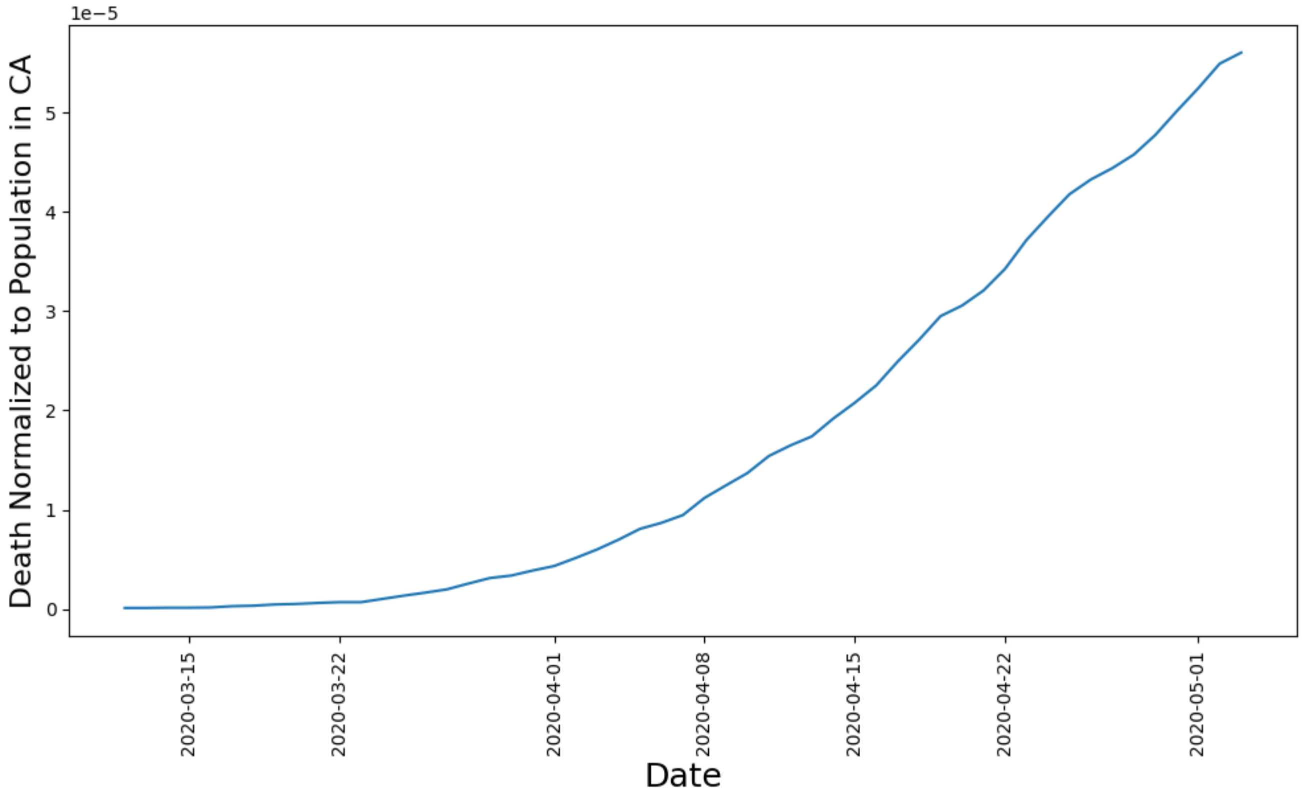

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].death_percent)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Percent Death in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```



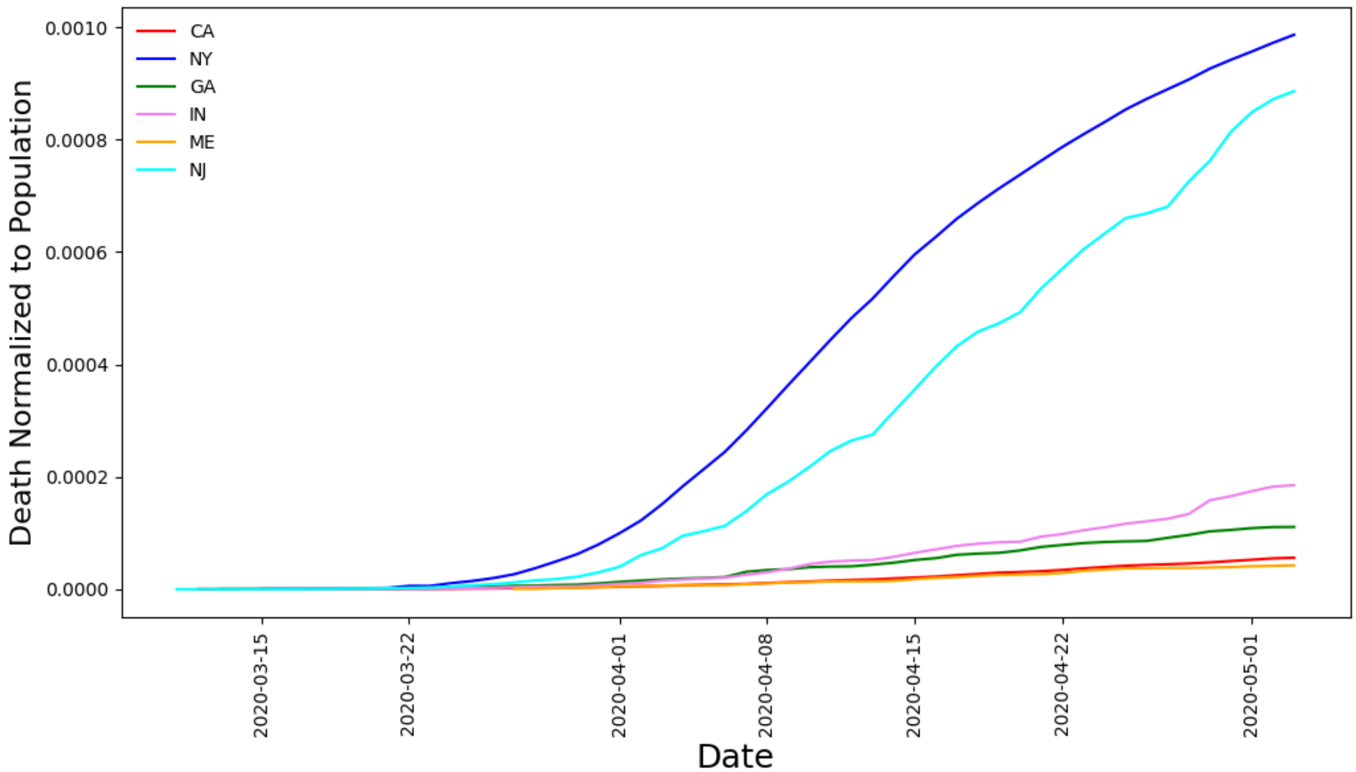

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].death_norm)
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Death Normalized to Population in CA', fontsize=16)
plt.show()
```

```
No handles with labels found to put in legend.
<Figure size 640x480 with 0 Axes>
```




```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

plt.plot(df_state_dict['CA'].death_norm, color="red", label="CA")
plt.plot(df_state_dict['NY'].death_norm, color="blue", label="NY")
plt.plot(df_state_dict['GA'].death_norm, color="green", label="GA")
plt.plot(df_state_dict['IN'].death_norm, color="violet", label="IN")
plt.plot(df_state_dict['ME'].death_norm, color="orange", label="ME")
plt.plot(df_state_dict['NJ'].death_norm, color="cyan", label="NJ")
plt.xticks(rotation='vertical')

plt.legend(frameon=False)
plt.xlabel('Date', fontsize=18)
plt.ylabel('Death Normalized to Population', fontsize=16)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



Note how the population normalized death curves relate closely to population normalized postive test curves

```
# Curve fitting done at:  http://www.xuru.org/rt/NLR.asp#CopyPaste
```

```
# Fetch the parameters for each state (AexpBx^-1.csv) that fit to positive_norm = a*exp(b/x)
# where x is the number of days from March 4, 2020
from google.colab import files
uploaded = files.upload()
```

Choose Files AexpBx^-1.csv

- **AexpBx^-1.csv**(application/vnd.ms-excel) - 2391 bytes, last modified: 5/3/2020 - 100% done

```
Saving AexpBx^-1.csv to AexpBx^-1.csv
```

```
# Load the parameters for each state (AexpBx^-1.csv) that fit to positive_norm = a*exp(b/x)
import io
df_state_params = pd.read_csv(io.StringIO(uploaded['AexpBx^-1.csv'].decode('utf-8')))
df_state_params.head()
```

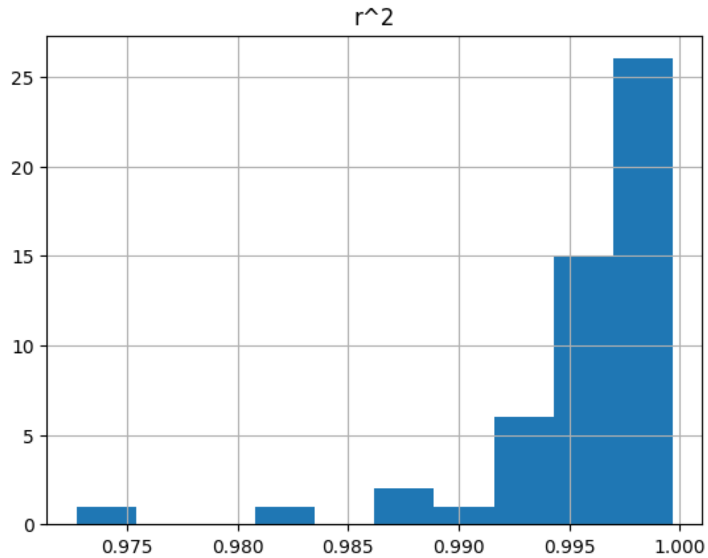
	State	a (10^-3)	b	fit rank	r^2
0	AK	2.593040	-75.366476	1.0	0.996906
1	AL	12.121593	-111.222242	2.0	0.997430
2	AR	2.941186	-75.356785	4.0	0.997586
3	AS	NaN	NaN	NaN	NaN
4	AZ	4.984063	-90.295019	1.0	0.998613

```
df_state_params.describe()
```

	a (10^-3)	b	fit rank	r^2
count	52.000000	52.000000	52.000000	52.000000
mean	16.215254	-100.951881	1.769231	0.995682
std	31.801661	25.545128	1.095720	0.004749
min	1.952592	-185.986576	1.000000	0.972728
25%	5.041013	-116.155268	1.000000	0.995399
50%	7.113788	-99.476492	1.000000	0.997030
75%	10.698133	-80.847333	2.000000	0.998087
max	190.553218	-49.104858	5.000000	0.999660

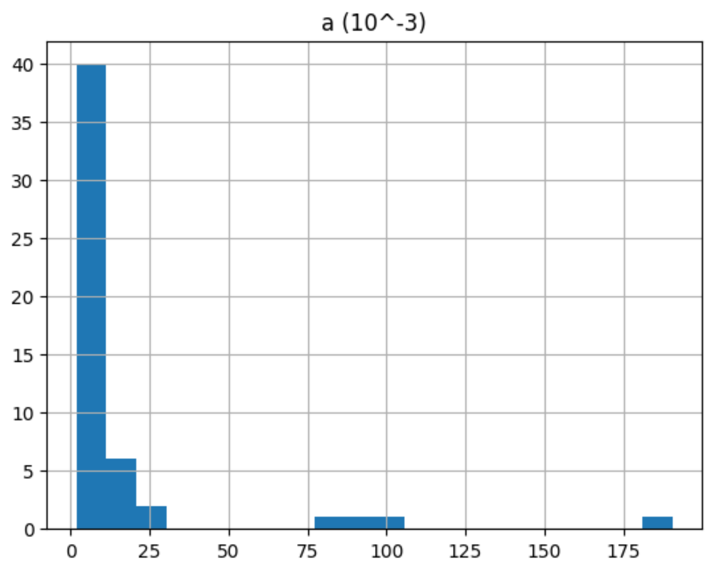
```
df_state_params.hist(column='r^2')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f093e578940>]],
      dtype=object)
```




```
df_state_params.hist(column='a (10^-3)', bins=20)
```

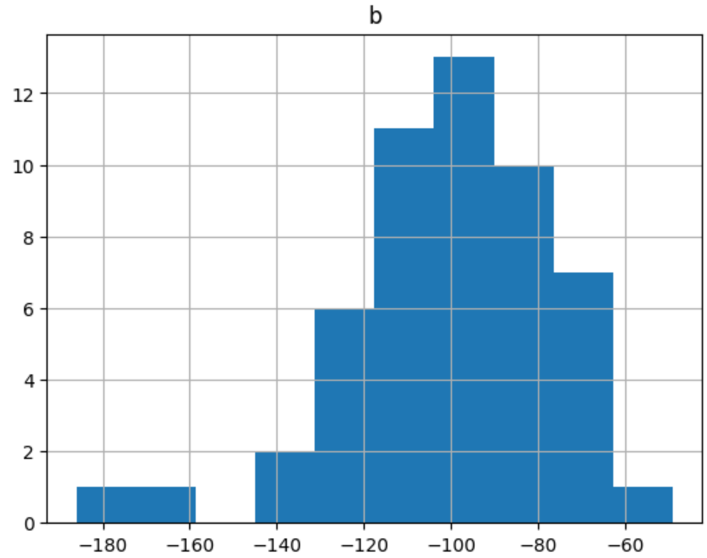
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f093e4ecc18>]],
      dtype=object)
```



High value outliers here are NJ (fit rank 1), NY, (fit rank 1), RI (fit rank 5), and SD (fit rank 4)

```
df_state_params.hist(column='b', bins=10)
```

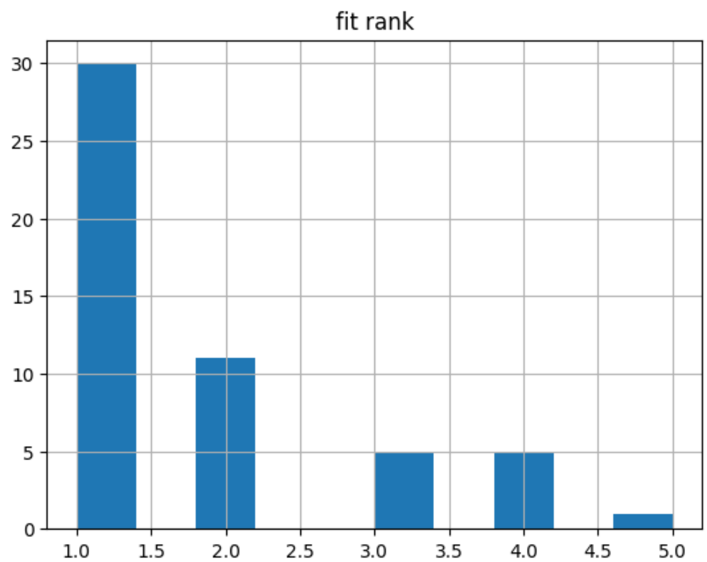
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f093e48b780>]],
      dtype=object)
```



Low value outliers here are RI (fit rank 5) and SD (fit rank 4).

```
df_state_params.hist(column='fit rank')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f093e433ba8>]],
      dtype=object)
```



The A*exp(B/x) functional form works extremely well for thirty of the 52 states (57.7%).

```
# Fetch static data for each state (CovidCompleteStateData.csv)
from google.colab import files
uploaded = files.upload()
```

Choose Files CovidCompl…teData.csv

- **CovidCompleteStateData.csv**(application/vnd.ms-excel) - 60510 bytes, last modified: 4/20/2020 - 100% done

```
Saving CovidCompleteStateData.csv to CovidCompleteStateData.csv
```

```
# Load static data for each state (CovidCurrentStateData.csv)
import io
df_state_data = pd.read_csv(io.StringIO(uploaded['CovidCompleteStateData.csv'].decode('utf-8')))
df_state_data.head()
```

	State	Sum of NUM_Medicare_BEN	Sum of NUM_BEN_Age_Less_65	Sum of NUM_BEN_Age_65_to_74	Sum of NUM_BEN_Age_75_to_84	Sum of NUM_BEN_Age_Greater_84	Sum of NUM_Female_BEN	NUM_
0	AK	1820384.0	270970.0	809516.0	468255.0	175296.0	1034762.0	
1	AL	10804823.0	2065353.0	4386595.0	2980828.0	1190504.0	6237445.0	∠
2	AR	15892716.0	2818665.0	6370265.0	4555468.0	1848506.0	9275039.0	(
3	AS	NaN	NaN	NaN	NaN	NaN	NaN	
4	AZ	10786064.0	886596.0	4861035.0	3377040.0	1294375.0	5944519.0	∠

5 rows × 116 columns

```
# Feature Engineering
# Land Area/Water Area
# df_state_data['State Area Ratio'] = df_state_data['Land Area']/df_state_data['Water Area']
df_state_data['State Area Ratio'] = df_state_data['Land Area'].divide(df_state_data['Water Area'], fill_value=0)

# Elevation Ratio = Highest Elevation/Mean Elevation
# df_state_data['Elevation Ratio'] = df_state_data['Highest Elevation']/df_state_data['Mean Elevation']
df_state_data['Elevation Ratio'] = df_state_data['Highest Elevation'].divide(df_state_data['Mean Elevation'], fill_v

# Capital Area Ratio = Capital Land Area/Capital Water Area
# df_state_data['Capital Area Ratio'] = df_state_data['Captial Land Area']/df_state_data['Capital Water Area']
df_state_data['Captial Land Area'] = df_state_data['Captial Land Area'].astype(float)
df_state_data['Capital Area Ratio'] = df_state_data['Captial Land Area'].divide(df_state_data['Capital Water Area'],

# Boundaries = Number of boarding states + On Coast + Borders Another Country
df_state_data['Boundaries'] = df_state_data['Number of bordering states'] + df_state_data['On Coast'] + df_state_dat

# Latitude Difference to State Capital = Latitude - Capital Latitude
df_state_data['Latitude Difference to State Capital'] = df_state_data['Latitude'] - df_state_data['Capital Latitude'

# Longitude Difference to State Capital = Capital Longitude - Longitude
df_state_data['Longitude Difference to State Capital'] = df_state_data['Capital Longitude'] - df_state_data['Longitu

# Latitude Difference to DC = Latitude - DC Latitude
df_state_data['Latitude Difference to DC'] = df_state_data['Latitude'] - 38.904722

# Longitude Difference to DC = DC Longitude - Longitude
df_state_data['Longitude Difference to DC'] = -77.016389 - df_state_data['Longitude']

# Latitude Difference to US Center = Latitude - Center Latitude
df_state_data['Latitude Difference to Center'] = df_state_data['Latitude'] - 39.833333

# Longitude Different to US Center = Center Longitude - Longitude
df_state_data['Longitude Difference to Center'] = -98.585522 - df_state_data['Longitude']
```

```
df_state_data.head()
```

	State	Sum of NUM_Medicare_BEN	Sum of NUM_BEN_Age_Less_65	Sum of NUM_BEN_Age_65_to_74	Sum of NUM_BEN_Age_75_to_84	Sum of NUM_BEN_Age_Greater_84	Sum of NUM_Female_BEN	NUM_
0	AK	1820384.0	270970.0	809516.0	468255.0	175296.0	1034762.0	
1	AL	10804823.0	2065353.0	4386595.0	2980828.0	1190504.0	6237445.0	∠
2	AR	15892716.0	2818665.0	6370265.0	4555468.0	1848506.0	9275039.0	(
3	AS	NaN	NaN	NaN	NaN	NaN	NaN	
4	AZ	10786064.0	886596.0	4861035.0	3377040.0	1294375.0	5944519.0	∠

5 rows × 126 columns

```
df_state_data.shape
```

```
(56, 126)
```

```
# Define variables for regression
df_temp1 = df_state_data.drop(df_state_data.index[[3, 12, 27, 42, 50]])
X = df_temp1.drop('State', axis = 1)
```

```
df_temp2 = df_state_params.drop(df_state_data.index[[3, 12, 27, 42, 50]])
y = df_temp2['a (10^-3)']
```

```
# Look at correlation coefficients
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 1000)
X.corr()
```

	Sum of NUM_Medicare_BEN	Sum of NUM_BEN_Age_Less_65	Sum of NUM_BEN_Age_65_to_74	Sum of NUM_BEN_Age_75_to_84	NU
Sum of NUM_Medicare_BEN	1.000000	0.981404	0.998624	0.998100	
Sum of NUM_BEN_Age_Less_65	0.981404	1.000000	0.978099	0.969440	
Sum of NUM_BEN_Age_65_to_74	0.998624	0.978099	1.000000	0.996374	
Sum of NUM_BEN_Age_75_to_84	0.998100	0.969440	0.996374	1.000000	
Sum of NUM_BEN_Age_Greater_84	0.989961	0.960650	0.982712	0.992601	
Sum of NUM_Female_BEN	0.999917	0.982576	0.998372	0.997916	
Sum of NUM_Male_BEN	0.999897	0.979741	0.998636	0.998296	
Sum of NUM_Black_or_African_American_BEN	0.896692	0.926091	0.895722	0.884218	
Sum of NUM_Asian_Pacific_Islander_BEN	0.525530	0.475021	0.517514	0.530001	
Sum of NUM_Hispanic_BEN	0.893302	0.827878	0.902298	0.899556	
Sum of NUM_American_IndianAlaska_Native_BEN	0.082561	0.059858	0.091513	0.086836	
Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	0.823477	0.774080	0.803783	0.832225	
Sum of NUM_Non-Hispanic_White_BEN	0.996838	0.978894	0.994391	0.996119	
Sum of NUM_Minorities	0.958442	0.925721	0.961095	0.957721	
Sum of Average_Age_of_BEN	0.682483	0.730359	0.686432	0.663590	
Sum of NUM_BEN_Atrial_Fibrillation	0.990425	0.969550	0.985604	0.991418	
Sum of NUM_BEN_Asthma	0.995532	0.979588	0.991583	0.992903	
Sum of NUM_BEN_Cancer	0.994765	0.972149	0.992903	0.994874	
Sum of NUM_BEN_Heart_Failure	0.997133	0.985150	0.995371	0.993915	
Sum of NUM_BEN_Chronic_Kidney_Disease	0.997501	0.980301	0.997095	0.995430	
Sum of NUM_BEN_Chronic_Obstructive_Pulmonary_Disease	0.986234	0.980624	0.981625	0.983999	
Sum of NUM_BEN_Hyperlipidemia	0.996237	0.974348	0.994742	0.996423	
Sum of NUM_BEN_Diabetes	0.997754	0.981227	0.996544	0.995687	
Sum of NUM_BEN_Hypertension	0.998856	0.982300	0.998079	0.996943	
Sum of NUM_BEN_Ischemic_Heart_Disease	0.994006	0.975145	0.991547	0.994105	
Sum of NUM_BEN_Stroke	0.990547	0.972081	0.988818	0.990024	
Sum of PCT_MEDICARE	0.713702	0.762102	0.716971	0.696228	
% Urban Pop	0.246412	0.181090	0.240984	0.259055	
Density (P/mi2)	-0.095479	-0.105571	-0.096280	-0.092015	
Children 0-18	0.886252	0.846604	0.876226	0.888481	
Adults 19-25	0.865749	0.826231	0.852680	0.868860	
Adults 26-34	0.848661	0.804492	0.835397	0.852982	
Adults 35-54	0.861684	0.820010	0.848035	0.865769	
Adults 55-64	0.840536	0.802214	0.822003	0.845654	
65+	0.842520	0.796154	0.822919	0.852028	
Latitude	-0.400391	-0.397373	-0.403138	-0.407192	
Longitude	0.046601	0.092974	0.034115	0.040031	
Land Area	0.229013	0.193883	0.242084	0.230058	
Water Area	0.042895	0.056385	0.036723	0.038782	
Mean Elevation	-0.163276	-0.224740	-0.147730	-0.155029	
Highest Elevation	-0.059881	-0.137582	-0.040603	-0.049835	
Lowest elevation	-0.354394	-0.352655	-0.344053	-0.355481	
Number of bordering states	0.077790	0.135863	0.075964	0.059448	
On Coast	0.471024	0.505115	0.442960	0.461862	
Borders Another Country	[illegible]	[illegible]	[illegible]	[illegible]	

Borders Another Country	0.358143	0.310618	0.363869	0.356815
Capital Latitude	-0.388663	-0.393979	-0.394070	-0.392266
Capital Longitude	0.027375	0.076949	0.015075	0.019608
Captial Land Area	0.008902	-0.002410	0.018688	0.009403
Capital Water Area	-0.087670	-0.096352	-0.083610	-0.087193
Capital Mean Elevation	-0.194009	-0.217624	-0.182931	-0.190725
Capital is the Largest City	-0.171080	-0.147972	-0.165860	-0.173283
Largest City Latitude	-0.421170	-0.421496	-0.425109	-0.424938
Largest City Longitude	0.057094	0.102104	0.044423	0.050338
Number of Counties	0.663716	0.710105	0.670375	0.645677
Became a State	-0.140415	-0.200801	-0.128869	-0.126557
DaysSinceStayatHomeOrder	-0.020651	-0.019693	-0.030343	-0.027347
DaysSinceFirstPositive	0.368252	0.319941	0.366229	0.374653
DaysSinceTestStart	0.290649	0.242592	0.289428	0.297948
15-49yearsAllcauses	0.888203	0.856564	0.874919	0.889982
15-49yearsAsthma	0.824682	0.787879	0.807656	0.827220
15-49yearsChronickidneydisease	0.918864	0.893772	0.909568	0.918825
15-49yearsChronicobstructivepulmonarydisease	0.896769	0.878089	0.880516	0.897303
15-49yearsDiabetesmellitus	0.912330	0.881654	0.900896	0.914260
15-49yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.881251	0.864222	0.866766	0.880121
15-49yearsIschemicheartdisease	0.928387	0.927789	0.916634	0.923405
15-49yearsNeoplasms	0.887461	0.860138	0.873067	0.888685
15-49yearsOtherchronicrespiratorydiseases	0.906636	0.885246	0.892415	0.906637
15-49yearsRheumaticheartdisease	0.903473	0.893269	0.893364	0.898792
15-49yearsStroke	0.919789	0.898558	0.910295	0.919449
50-69yearsAllcauses	0.880146	0.855617	0.863069	0.881923
50-69yearsAsthma	0.801803	0.765502	0.781306	0.805925
50-69yearsChronickidneydisease	0.917312	0.898401	0.905572	0.916416
50-69yearsChronicobstructivepulmonarydisease	0.879259	0.872843	0.860771	0.878641
50-69yearsDiabetesmellitus	0.882501	0.857522	0.865414	0.884673
50-69yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.863191	0.840683	0.846169	0.863950
50-69yearsIschemicheartdisease	0.905979	0.901073	0.890019	0.902683
50-69yearsNeoplasms	0.872500	0.853408	0.854035	0.873415
50-69yearsOtherchronicrespiratorydiseases	0.885021	0.875159	0.867604	0.883457
50-69yearsRheumaticheartdisease	0.892519	0.890373	0.880528	0.886667
50-69yearsStroke	0.907993	0.892290	0.895108	0.907388
70+yearsAllcauses	0.849263	0.819400	0.828488	0.854118
70+yearsAsthma	0.791486	0.748032	0.769602	0.799338
70+yearsChronickidneydisease	0.877077	0.858325	0.859219	0.877628
70+yearsChronicobstructivepulmonarydisease	0.866728	0.842585	0.846829	0.871197
70+yearsDiabetesmellitus	0.845276	0.815442	0.823824	0.850785
70+yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.833832	0.799993	0.814083	0.839080
70+yearsIschemicheartdisease	0.841243	0.819787	0.818295	0.844148
70+yearsNeoplasms	0.837485	0.808405	0.816021	0.842466
70+yearsOtherchronicrespiratorydiseases	0.875916	0.859545	0.858192	0.875620
70+yearsRheumaticheartdisease	0.844465	0.839621	0.826738	0.839480
70+yearsStroke	0.871562	0.849583	0.854254	0.873252

AllAgesAllcauses	0.880003	0.851293	0.863399	0.882592
AllAgesAsthma	0.833253	0.794917	0.815812	0.836910
AllAgesChronickidneydisease	0.905462	0.885503	0.891504	0.905307
AllAgesChronicobstructivepulmonarydisease	0.877214	0.860833	0.858127	0.879265
AllAgesDiabetesmellitus	0.879728	0.852121	0.862209	0.882908
AllAgesInterstitiallungdiseaseandpulmonarysarcoidosis	0.853912	0.826093	0.835749	0.856759
AllAgesIschemicheartdisease	0.883535	0.870954	0.864460	0.883069
AllAgesNeoplasms	0.865325	0.841450	0.846325	0.867726
AllAgesOtherchronicrespiratorydiseases	0.903592	0.885967	0.888445	0.902970
AllAgesRheumaticheartdisease	0.880357	0.875286	0.866144	0.875089
AllAgesStroke	0.895398	0.875735	0.880674	0.895978
AllAgesTotal	0.880507	0.853923	0.863463	0.882813
Airpollution	0.889229	0.888442	0.875092	0.882964
Highbody-massindex	0.893797	0.872739	0.877133	0.894624
Highfastingplasmaglucose	0.886795	0.870124	0.868909	0.887417
HighLDLcholesterol	0.893215	0.882483	0.875398	0.892023
Highsystolicbloodpressure	0.897453	0.882631	0.880346	0.897131
Impairedkidneyfunction	0.889934	0.872693	0.873173	0.890034
Noaccesstohandwashingfacility	0.877603	0.857781	0.862453	0.876519
Smoking	0.881579	0.866726	0.862831	0.882612
Log10Pop	0.728494	0.737902	0.714057	0.722320
DaysSinceInfection	0.422525	0.373010	0.419727	0.431233
Children0-18	0.167133	0.180823	0.181296	0.159580
Allriskfactors	0.882815	0.860944	0.865530	0.884217
State Area Ratio	-0.141342	-0.180449	-0.126323	-0.134563
Elevation Ratio	0.020332	0.007311	0.029598	0.023691
Capital Area Ratio	-0.119284	-0.151665	-0.109968	-0.112407
Boundaries	0.499356	0.556393	0.479330	0.477960
Latitude Difference to State Capital	-0.268652	-0.211068	-0.252026	-0.293417
Longitude Difference to State Capital	-0.143646	-0.133106	-0.139285	-0.150250
Latitude Difference to DC	-0.400391	-0.397373	-0.403138	-0.407192
Longitude Difference to DC	-0.046601	-0.092974	-0.034115	-0.040031
Latitude Difference to Center	-0.400391	-0.397373	-0.403138	-0.407192
Longitude Difference to Center	-0.046601	-0.092974	-0.034115	-0.040031

```
#  Note that there are many highly correlated features which need to be dropped
# Create absolute value correlation matrix
corr_matrix = X.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

# Drop features by index which were identified as being highly correlated
X = X.drop(X[to_drop], axis=1)
```

```
X.head()
```

	Sum of NUM_Medicare_BEN	Sum of NUM_Black_or_African_American_BEN	Sum of NUM_Asian_Pacific_Islander_BEN	Sum of NUM_Hispanic_BEN	Sum of NUM_American_IndianAlaska_Nativ
0	1820384.0	62311.0	76773.0	46525.0	14
1	10804823.0	1549811.0	30624.0	65500.0	
2	15892716.0	1334245.0	19642.0	108428.0	6
4	10786064.0	221183.0	61840.0	689880.0	17
5	42579588.0	2072012.0	3276415.0	5674776.0	11

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51 entries, 0 to 55
Data columns (total 38 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   Sum of NUM_Medicare_BEN                     51 non-null     float64
 1   Sum of NUM_Black_or_African_American_BEN    51 non-null     float64
 2   Sum of NUM_Asian_Pacific_Islander_BEN       51 non-null     float64
 3   Sum of NUM_Hispanic_BEN                     51 non-null     float64
 4   Sum of NUM_American_IndianAlaska_Native_BEN 51 non-null     float64
 5   Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified  51 non-null     float64
 6   Sum of Average_Age_of_BEN                   51 non-null     float64
 7   Sum of PCT_MEDICARE                         51 non-null     float64
 8   % Urban Pop                                 51 non-null     float64
 9   Density (P/mi2)                             51 non-null     float64
 10  Children 0-18                               51 non-null     float64
 11  Latitude                                    51 non-null     float64
 12  Longitude                                   51 non-null     float64
 13  Land Area                                   51 non-null     float64
 14  Water Area                                  51 non-null     float64
 15  Mean Elevation                              51 non-null     float64
 16  Highest Elevation                           51 non-null     float64
 17  Lowest elevation                            51 non-null     float64
 18  Number of bordering states                  51 non-null     float64
 19  On Coast                                    51 non-null     float64
 20  Borders Another Country                     51 non-null     float64
 21  Captial Land Area                           51 non-null     float64
 22  Capital Water Area                          51 non-null     float64
 23  Capital Mean Elevation                      51 non-null     float64
 24  Capital is the Largest City                 51 non-null     float64
 25  Became a State                              51 non-null     float64
 26  DaysSinceStayatHomeOrder                    51 non-null     float64
 27  DaysSinceFirstPositive                      51 non-null     float64
 28  DaysSinceTestStart                          51 non-null     float64
 29  Log10Pop                                    51 non-null     float64
 30  DaysSinceInfection                          51 non-null     float64
 31  Children0-18                                51 non-null     float64
 32  State Area Ratio                            51 non-null     float64
 33  Elevation Ratio                             51 non-null     float64
 34  Capital Area Ratio                          51 non-null     float64
 35  Boundaries                                  51 non-null     float64
 36  Latitude Difference to State Capital        51 non-null     float64
 37  Longitude Difference to State Capital       51 non-null     float64
dtypes: float64(38)
memory usage: 15.5 KB
```

```
X.describe()
```

	Sum of NUM_Medicare_BEN	Sum of NUM_Black_or_African_American_BEN	Sum of NUM_Asian_Pacific_Islander_BEN	Sum of NUM_Hispanic_BEN	NUM_American_IndianAlaska_I
count	5.100000e+01	5.100000e+01	5.100000e+01	5.100000e+01	
mean	1.038431e+07	9.464777e+05	1.411691e+05	5.310095e+05	38
std	1.311026e+07	1.274593e+06	4.722330e+05	1.629961e+06	87
min	1.655870e+05	2.960000e+02	1.660000e+02	4.130000e+02	
25%	2.252305e+06	5.366600e+04	6.445500e+03	3.101950e+04	2
50%	6.272609e+06	3.156040e+05	2.579200e+04	1.042170e+05	7
75%	1.471830e+07	1.547566e+06	7.063400e+04	2.005865e+05	28
max	7.644909e+07	7.011107e+06	3.276415e+06	1.007620e+07	560

Double-click (or enter) to edit

```
# Train/validate split: random 75/25% train/validate split.
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.25, random_state = 42)

X_train.shape, y_train.shape, X_val.shape, y_val.shape
```

```
((38, 38), (38,), (13, 38), (13,))
```

```
X_train.describe()
```

	Sum of NUM_Medicare_BEN	Sum of NUM_Black_or_African_American_BEN	Sum of NUM_Asian_Pacific_Islander_BEN	Sum of NUM_Hispanic_BEN	NUM_American_IndianAlaska_I
count	3.800000e+01	3.800000e+01	3.800000e+01	3.800000e+01	
mean	1.014125e+07	9.685705e+05	1.623107e+05	3.942231e+05	37
std	9.963253e+06	1.001560e+06	5.333709e+05	1.021129e+06	93
min	3.472690e+05	2.689000e+03	4.580000e+02	2.622000e+03	
25%	2.518838e+06	4.934350e+04	1.427175e+04	3.676725e+04	4
50%	7.473651e+06	5.120990e+05	3.068000e+04	1.071920e+05	9
75%	1.563758e+07	1.560497e+06	9.455175e+04	1.983508e+05	27
max	4.257959e+07	3.265865e+06	3.276415e+06	5.674776e+06	560

```
# Optimizing Hyperparameters for Random Forest Regressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

# Define classifier
forest = RandomForestRegressor(random_state = 1)

# Parameters to fit

max_depth = [2, 3, 4]
n_estimators = [35, 36, 37]
min_samples_split = [1.5, 2, 2.5]
min_samples_leaf = [3.5, 4, 4.5]
max_leaf_nodes = [None]
max_features = ['auto']
ccp_alpha = [0.0, 0.00625, 0.0125]
min_weight_fraction_leaf = [0.0, 0.00625, 0.0125]

hyperF = dict(n_estimators = n_estimators, max_depth = max_depth,
              min_samples_split = min_samples_split,
              min_samples_leaf = min_samples_leaf,
              max_leaf_nodes = max_leaf_nodes,
              max_features = max_features,
```

```
             ccp_alpha=ccp_alpha,
             min_weight_fraction_leaf=min_weight_fraction_leaf)

gridF = GridSearchCV(forest, hyperF, cv = 3, verbose = 10,
                     scoring='r2', return_train_score=True,
                     n_jobs = -1)
bestF = gridF.fit(X_train, y_train)

# Output best accuracy and best parameters
print('The score achieved with the best parameters = ', gridF.best_score_, '\n')
print('The parameters are:', gridF.best_params_)
```

```
Fitting 3 folds for each of 729 candidates, totalling 2187 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1862s.) Setting batch_size=2.
[Parallel(n_jobs=-1)]: Done  22 tasks      | elapsed:    1.8s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1010s.) Setting batch_size=4.
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1858s.) Setting batch_size=8.
[Parallel(n_jobs=-1)]: Done  44 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed:    5.3s
[Parallel(n_jobs=-1)]: Done 276 tasks      | elapsed:    6.7s
[Parallel(n_jobs=-1)]: Done 364 tasks      | elapsed:    8.5s
[Parallel(n_jobs=-1)]: Done 468 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 572 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 692 tasks      | elapsed:   14.8s
[Parallel(n_jobs=-1)]: Done 812 tasks      | elapsed:   16.7s
[Parallel(n_jobs=-1)]: Done 948 tasks      | elapsed:   19.6s
[Parallel(n_jobs=-1)]: Done 1084 tasks      | elapsed:   22.1s
[Parallel(n_jobs=-1)]: Done 1236 tasks      | elapsed:   25.0s
[Parallel(n_jobs=-1)]: Done 1388 tasks      | elapsed:   28.3s
[Parallel(n_jobs=-1)]: Done 1556 tasks      | elapsed:   31.0s
[Parallel(n_jobs=-1)]: Done 1724 tasks      | elapsed:   34.5s
[Parallel(n_jobs=-1)]: Done 1908 tasks      | elapsed:   38.2s
[Parallel(n_jobs=-1)]: Done 2092 tasks      | elapsed:   41.9s
The score achieved with the best parameters =  -7.248154911069659

The parameters are: {'ccp_alpha': 0.0, 'max_depth': 3, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_sample
[Parallel(n_jobs=-1)]: Done 2187 out of 2187 | elapsed:   43.3s finished
```

```
!pip install category_encoders==2.0.0
```

```
Collecting category_encoders==2.0.0
  Downloading https://files.pythonhosted.org/packages/6e/a1/f7a22f144f33be78afeb06bfa78478e8284a64263a3c09b1ef54e673841e/category_encoder
     |████████████████████████████████| 92kB 2.4MB/s
Requirement already satisfied: patsy>=0.4.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.5.1)
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (1.0.3)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.22.2.pos
Requirement already satisfied: statsmodels>=0.6.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.10.2)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (1.4.1)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (1.18.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from patsy>=0.4.1->category_encoders==2.0.0) (1.12.0)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21.1->category_encoders==
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21.1->category_encoders==2.0.0) (20
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.20.0->category_encoders==2.0.
Installing collected packages: category-encoders
Successfully installed category-encoders-2.0.0
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
import category_encoders as ce
from sklearn.impute import SimpleImputer

pipeline1 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy='mean'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False))

pipeline1.fit(X_train, y_train)
```

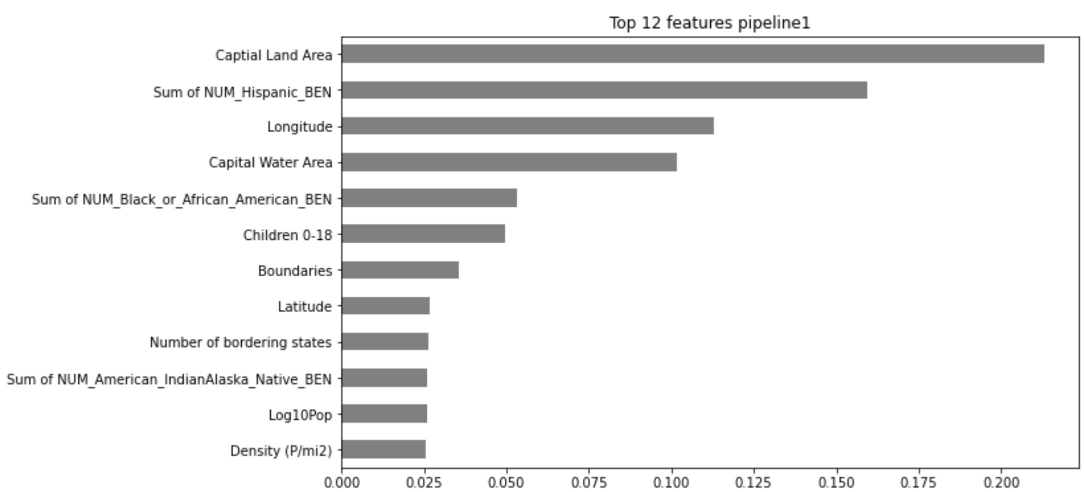
```
pipeline1.fit(X_train, y_train)

# Get the model's training accuracy
print("Training Accurary:  R^2 = ", pipeline1.score(X_train,y_train))

# Get the model's validation accuracy
print('Validation Accuracy:  R^2 = ', pipeline1.score(X_val, y_val))
```

```
Training Accurary:  R^2 =  0.3013954634238487
Validation Accuracy:  R^2 =  0.069388620867604
```

```
print("Feature Importances =")
#print(RandomForestRegressor.feature_importances_)
print(pipeline1.steps[2][1].feature_importances_)
```

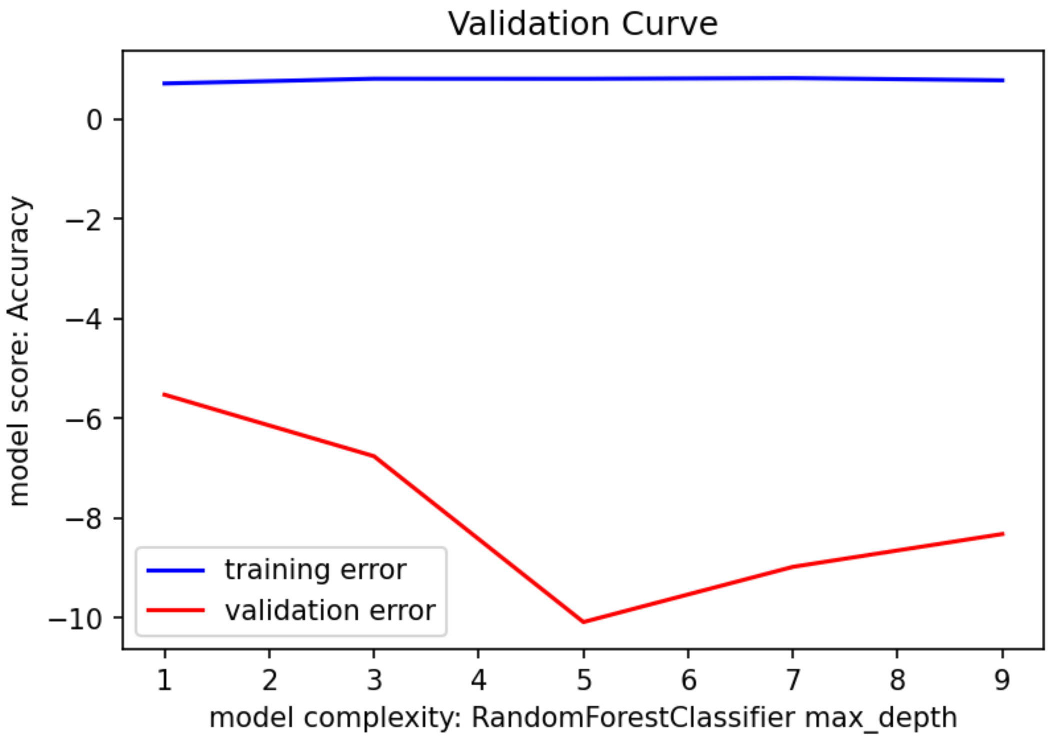
```
Feature Importances =
[1.58960229e-03 5.30566745e-02 4.17410911e-03 1.59284318e-01
 2.61004152e-02 2.17672192e-02 7.81822081e-03 2.43729475e-02
 2.90624300e-03 2.56504641e-02 4.95719120e-02 2.69244094e-02
 1.12768719e-01 1.23221013e-04 4.58989637e-04 2.49941071e-03
 8.43190668e-03 0.00000000e+00 2.63169045e-02 2.37762734e-02
 0.00000000e+00 2.13022103e-01 1.01510181e-01 1.46479728e-04
 0.00000000e+00 2.38233658e-02 5.38444596e-03 0.00000000e+00
 2.61738842e-04 2.59337047e-02 0.00000000e+00 0.00000000e+00
 2.95326741e-03 0.00000000e+00 3.98608558e-03 3.56252545e-02
 3.63475050e-03 6.12666322e-03]
```

```
# Plot of feature importances from pure Random Forest Regressor
%matplotlib inline
import matplotlib.pyplot as plt
# Get feature importances
encoder = pipeline1.named_steps['onehotencoder']
encoded = encoder.transform(X_train)
rf = pipeline1.named_steps['randomforestregressor']
importances1 = pd.Series(rf.feature_importances_, encoded.columns)
# Plot feature importances
n = 12
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features pipeline1')
importances1.sort_values()[-n:].plot.barh(color='grey');
```




```
# Generate validation curves
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import validation_curve
pipeline2 = make_pipeline(
    ce.OrdinalEncoder(),
    SimpleImputer(),
    RandomForestRegressor()
)

depth = range(1, 10, 2)
train_scores, val_scores = validation_curve(
    pipeline2, X_train, y_train,
    param_name='randomforestregressor__max_depth',
    param_range=depth,
    cv=3,
```

```
    n_jobs=-1
)

plt.figure(dpi=150)
plt.plot(depth, np.mean(train_scores, axis=1), color='blue', label='training error')
plt.plot(depth, np.mean(val_scores, axis=1), color='red', label='validation error')
plt.title('Validation Curve')
plt.xlabel('model complexity: RandomForestClassifier max_depth')
plt.ylabel('model score: Accuracy')
plt.legend();
```




```
# Get drop-column importances
column = 'Captial Land Area'

pipeline3 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False))

# Fit without column
pipeline3.fit(X_train.drop(columns=column), y_train)
score_without = pipeline3.score(X_val.drop(columns=column), y_val)
print(f'Validation Accuracy without {column}: {score_without}')

# Fit with column
pipeline3.fit(X_train, y_train)
score_with = pipeline3.score(X_val, y_val)
print(f'Validation Accuracy with {column}: {score_with}')

# Compare the error with & without column
print(f'Drop-Column Importance for {column}: {score_with - score_without}')
```

```
Validation Accuracy without Captial Land Area: 0.10771758680677601
Validation Accuracy with Captial Land Area: 0.069388620867604
Drop-Column Importance for Captial Land Area: -0.03832896593917201
```

```
# Using Eli5 library which does not work with pipelines
transformers = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy='most_frequent')
```

```
)

X_train_transformed = transformers.fit_transform(X_train)
X_val_transformed = transformers.transform(X_val)

model1 =     RandomForestRegressor(bootstrap=True, ccp_alpha=0, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)

model1.fit(X_train_transformed, y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

```
# Get permutation importances
! pip install eli5
from eli5.sklearn import PermutationImportance
import eli5

permuter = PermutationImportance(
    model1,
    scoring='r2',
    n_iter=2,
    random_state=42
)

permuter.fit(X_val_transformed, y_val)
feature_names = X_val.columns.tolist()

eli5.show_weights(
    permuter,
    top=None, # show permutation importances for all features
    feature_names=feature_names
)
```

```
Collecting eli5
  Downloading https://files.pythonhosted.org/packages/97/2f/c85c7d8f8548e460829971785347e14e45fa5c6617da374711dec8cb38cc/eli5-0.10.1-py2.
     |████████████████████████████████| 112kB 2.8MB/s
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (1.18.3)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.22.2.post1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from eli5) (1.12.0)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (from eli5) (0.10.1)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (19.3.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from eli5) (2.11.2)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.8.7)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from eli5) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.18->eli5) (0.14.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2->eli5) (1.1.1)
Installing collected packages: eli5
Successfully installed eli5-0.10.1
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  deprecated
  warnings.warn(message, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.feature_selection.base module is  dep
  warnings.warn(message, FutureWarning)
Using TensorFlow backend.
```

Weight	Feature
0.1883 ± 0.0854	Captial Land Area
0.0239 ± 0.2251	Longitude
0.0091 ± 0.0014	Density (P/mi2)
0.0063 ± 0.0098	Sum of NUM_Black_or_African_American_BEN
0.0049 ± 0.0032	Sum of NUM_Asian_Pacific_Islander_BEN
0.0045 ± 0.0043	Latitude
0.0044 ± 0.0104	Became a State
0.0024 ± 0.0001	Capital Area Ratio
0.0022 ± 0.0019	Sum of Average_Age_of_BEN
0.0021 ± 0.0014	Boundaries
0.0015 ± 0.0025	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified
0.0007 ± 0.0016	% Urban Pop
0.0005 ± 0.0008	On Coast
0.0003 ± 0.0014	Sum of NUM_Medicare_BEN
0 ± 0.0000	Capital is the Largest City
0 ± 0.0000	DaysSinceFirstPositive
0 ± 0.0000	Borders Another Country
0 ± 0.0000	DaysSinceInfection
0 ± 0.0000	Lowest elevation
0 ± 0.0000	Children0-18
0 ± 0.0000	Elevation Ratio
-0.0002 ± 0.0021	Latitude Difference to State Capital
-0.0002 ± 0.0001	Water Area
-0.0004 ± 0.0007	DaysSinceTestStart
-0.0004 ± 0.0004	Longitude Difference to State Capital
-0.0005 ± 0.0001	Capital Mean Elevation
-0.0005 ± 0.0003	Land Area
-0.0007 ± 0.0015	Mean Elevation
-0.0008 ± 0.0002	State Area Ratio
-0.0015 ± 0.0007	Highest Elevation
-0.0036 ± 0.0423	Sum of PCT_MEDICARE
-0.0069 ± 0.0009	DaysSinceStayatHomeOrder
-0.0077 ± 0.0167	Sum of NUM_American_IndianAlaska_Native_BEN
-0.0098 ± 0.0251	Number of bordering states
-0.0148 ± 0.0124	Capital Water Area
-0.0154 ± 0.0149	Children 0-18
-0.0157 ± 0.0063	Log10Pop
-0.0338 ± 0.1228	Sum of NUM_Hispanic_BEN

```
from sklearn.metrics import mean_squared_error, r2_score

# Coefficient of determination r2 for the training set
pipeline_score = permuter.score(X_train_transformed,y_train)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)

# Coefficient of determination r2 for the validation set
pipeline_score = permuter.score(X_val_transformed,y_val)
print("Coefficient of determination r2 for the validation set.: ", pipeline_score)

# The mean squared error
y_pred = permuter.predict(X_val_transformed)
print("Mean squared error: %.2f"% mean_squared_error(y_val, y_pred))
```

```
Coefficient of determination r2 for the training set.:  0.3013954634238487
Coefficient of determination r2 for the validation set.:  0.069388620867604
Mean squared error: 530.47
```

```
# Captial Land Area continues to be of importance
# Use importances for feature selection
print('Shape before removing features:', X_train.shape)
```

```
print('Shape before removing features:', X_train.shape)
```

```
Shape before removing features: (38, 38)
```

```
# Remove features of 0 importance
zero_importance = 0.0
mask = permuter.feature_importances_ > zero_importance
features1 = X_train.columns[mask]
X_train = X_train[features1]
print('Shape after removing features:', X_train.shape)
```

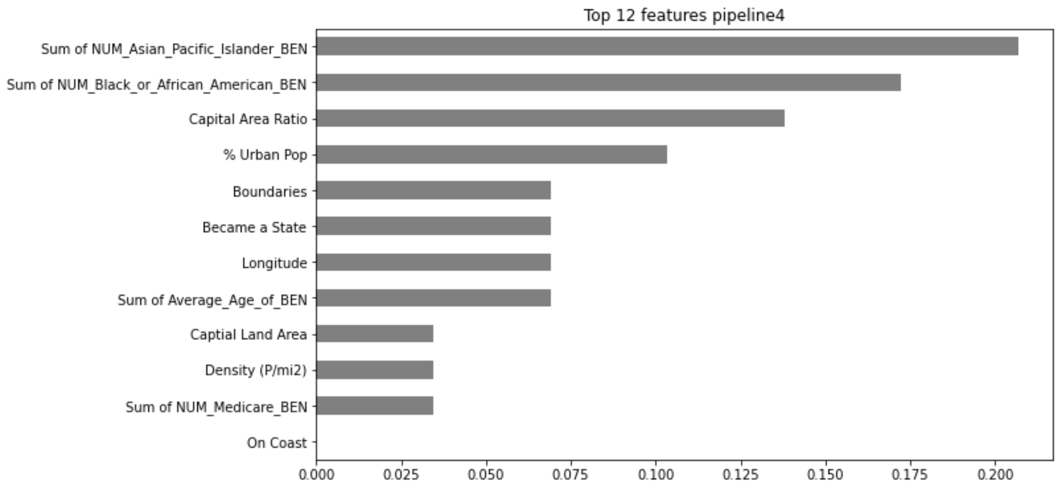
```
Shape after removing features: (38, 14)
```

```
# Random forest classifier with fourteen features
X_val = X_val[features1]
pipeline4 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                          max_depth=3, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=4,
                          min_samples_split=2, min_weight_fraction_leaf=0,
                          n_estimators=36, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False)
)

# Fit on train, score on val
pipeline4.fit(X_train, y_train);
```

```
from sklearn.metrics import mean_squared_error, r2_score

# Coefficient of determination r2 for the training set
pipeline_score = pipeline4.score(X_train,y_train)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)

# Coefficient of determination r2 for the validation set
pipeline_score = pipeline4.score(X_val,y_val)
print("Coefficient of determination r2 for the validation set.: ", pipeline_score)

# The mean squared error
y_pred = pipeline4.predict(X_val)
print("Mean squared error: %.2f"% mean_squared_error(y_val, y_pred))
```

```
Coefficient of determination r2 for the training set.:  0.26984125703310236
Coefficient of determination r2 for the validation set.:  0.11859396791756194
Mean squared error: 502.42
```

```
pipeline4.fit(X_val, y_val)
# Plot of features
%matplotlib inline
import matplotlib.pyplot as plt

# Get feature importances
encoder = pipeline4.named_steps['onehotencoder']
encoded = encoder.transform(X_val)
rf = pipeline4.named_steps['randomforestregressor']
importances2 = pd.Series(rf.feature_importances_, encoded.columns)

# Plot feature importances
#n = 4
n = 12
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features pipeline4')
importances2.sort_values()[-n:].plot.barh(color='grey');
```




```
!pip install pdpbox
```

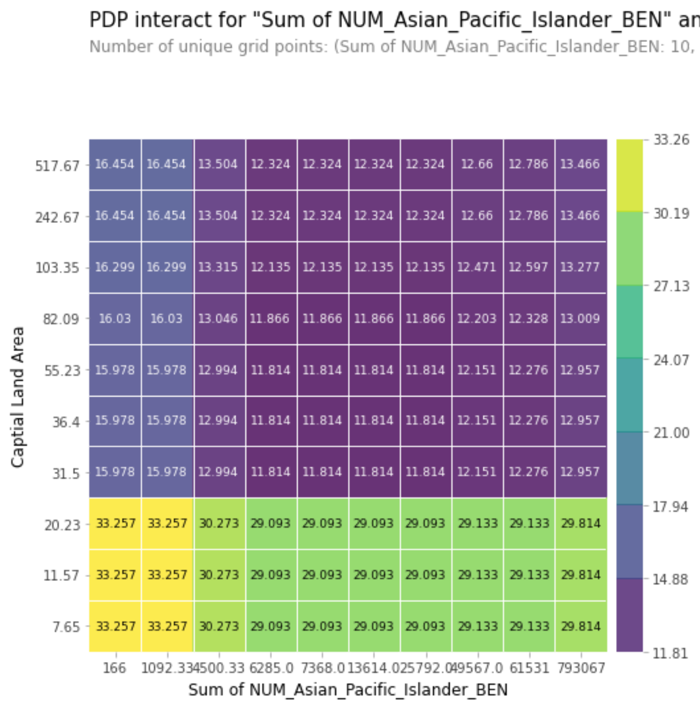
```
Collecting pdpbox
  Downloading https://files.pythonhosted.org/packages/87/23/ac7da5ba1c6c03a87c412e7e7b6e91a10d6ecf4474906c3e736f93940d49/PDPbox-0.2.0.tar
     |████████████████████████████████| 57.7MB 66kB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from pdpbox) (1.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from pdpbox) (1.18.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from pdpbox) (1.4.1)
Requirement already satisfied: matplotlib>=2.1.2 in /usr/local/lib/python3.6/dist-packages (from pdpbox) (3.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from pdpbox) (0.14.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.6/dist-packages (from pdpbox) (5.4.8)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from pdpbox) (0.22.2.post1)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas->pdpbox) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->pdpbox) (2018.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.1.2->pdpbox) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.1.2->pdpbox) (0.10.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->pandas->pdpbox) (1.12.0)
Building wheels for collected packages: pdpbox
  Building wheel for pdpbox (setup.py) ... done
  Created wheel for pdpbox: filename=PDPbox-0.2.0-cp36-none-any.whl size=57690722 sha256=2eb21b6a2e7bf0b40a562bf76671ac3e8c2da289d9100574
  Stored in directory: /root/.cache/pip/wheels/7d/08/51/63fd122b04a2c87d780464eeffb94867c75bd96a64d500a3fe
Successfully built pdpbox
Installing collected packages: pdpbox
Successfully installed pdpbox-0.2.0
```

```
model2 =     RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)

model2.fit(X_train, y_train)
```

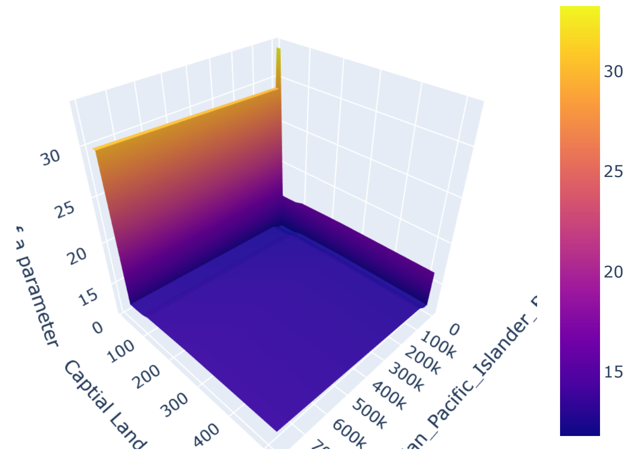
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=36, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

```
# Partial Dependence Plots with 2 features
from pdpbox.pdp import pdp_interact, pdp_interact_plot
#features2 = ['Density (P/mi2)', 'Captial Land Area']
features2 = ['Sum of NUM_Asian_Pacific_Islander_BEN', 'Captial Land Area']
interaction = pdp_interact(
#                           model=gb,
                           model=model2,
                           dataset=X_val,
                           model_features=X_val.columns,
                           features=features2
                           )
pdp_interact_plot(interaction, plot_type='grid', feature_names=features2);
```

```
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```




```
# A two feature partical dependence plot in 3D
pdp = interaction.pdp.pivot_table(
                                  values='preds',
                                  columns=features2[0],
                                  index=features2[1]
                                  )[::-1] # Slice notation to reverse index order so y axis is ascending
import plotly.graph_objs as go

target = 'Value of a parameter'

surface = go.Surface(x=pdp.columns,
                     y=pdp.index,
                     z=pdp.values)

layout = go.Layout(
                   scene=dict(
                              xaxis=dict(title=features2[0]),
                              yaxis=dict(title=features2[1]),
                              zaxis=dict(title=target)
                              )
)

fig = go.Figure(surface, layout)
fig.show()
```

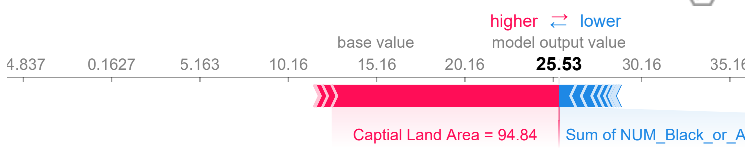



```
! pip install shap==0.23.0
! pip install -I shap
```

```
Collecting shap==0.23.0
  Downloading https://files.pythonhosted.org/packages/60/0d/8bd076821f7230edb2892ad982ea91ca25f2f925466563272e61eae891c6/shap-0.23.0.tar.
     |████████████████████████████████| 184kB 2.8MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (1.18.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (0.22.2.post1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (3.2.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (1.0.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (4.38.0)
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (5.5.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn->shap==0.23.0) (0.14.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap=
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (0.10.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->shap==0.23.0) (2018.9)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (2.1.3)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (0.8.1)
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.4.2)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.3.3)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (1.0.1
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (0.7.5)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (46.1.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.1->matplotlib->shap==0.23.0) (
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from pexpect; sys_platform != "win32"->ipython-
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2->ipython->shap==0.23.0) (0
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->shap==0.23.
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.23.0-cp36-cp36m-linux_x86_64.whl size=235674 sha256=888804ee0ed4369d5a0f68454e9a2429aa16412ebe8
  Stored in directory: /root/.cache/pip/wheels/c1/2c/aa/10d1782fe066536fcd564a2f8adea4dd05f57768236038855b
Successfully built shap
Installing collected packages: shap
Successfully installed shap-0.23.0
Collecting shap
  Downloading https://files.pythonhosted.org/packages/a8/77/b504e43e21a2ba543a1ac4696718beb500cfa708af2fb57cb54ce299045c/shap-0.35.0.tar.
     |████████████████████████████████| 276kB 2.8MB/s
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/03/27/e35e7c6e6a52fab9fcc64fc2b20c6b516eba930bb02b10ace3b38200d3ab/numpy-1.18.4-cp3
     |████████████████████████████████| 20.2MB 1.2MB/s
Collecting scipy
  Downloading https://files.pythonhosted.org/packages/dc/29/162476fd44203116e7980cfbd9352eef9db37c49445d1fec35509022f6aa/scipy-1.4.1-cp36
     |████████████████████████████████| 26.1MB 1.4MB/s
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/5e/d8/312e03adf4c78663e17d802fe2440072376fee46cada1404f1727ed77a32/scikit_learn-0.2
     |████████████████████████████████| 7.1MB 41.3MB/s
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/bb/71/8f53bdbcbc67c912b888b40def255767e475402e9df64050019149b1a943/pandas-1.0.3-cp3
     |████████████████████████████████| 10.0MB 174kB/s
Collecting tqdm>4.25.0
  Downloading https://files.pythonhosted.org/packages/c9/40/058b12e8ba10e35f89c9b1fdfc2d4c7f8c05947df2d5eb3c7b258019fda0/tqdm-4.46.0-py2.
     |████████████████████████████████| 71kB 8.5MB/s
Collecting joblib>=0.11
  Downloading https://files.pythonhosted.org/packages/28/5c/cf6a2b65a321c4a209efcdf64c2689efae2cb62661f8f6f4bb28547cf1bf/joblib-0.14.1-py
     |████████████████████████████████| 296kB 51.1MB/s
Collecting python-dateutil>=2.6.1
  Downloading https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b306af2bce5d134d78615cb/python_dateutil-
     |████████████████████████████████| 235kB 41.7MB/s
Collecting pytz>=2017.2
  Downloading https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e59d7d4efda580b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.
     |████████████████████████████████| 512kB 50.4MB/s
Collecting six>=1.5
  Downloading https://files.pythonhosted.org/packages/65/eb/1f97cb97bfc2390a276969c6fae16075da282f5058082d4cb10c6c5c1dba/six-1.14.0-py2.p
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.35.0-cp36-cp36m-linux_x86_64.whl size=394123 sha256=5ad6ad0eec64fad88c98344a30ed50224018eb684f8
  Stored in directory: /root/.cache/pip/wheels/e7/f7/0f/b57055080cf8894906b3bd3616d2fc2bfd0b12d5161bcb24ac
Successfully built shap
ERROR: google-colab 1.0.0 has requirement six~=1.12.0, but you'll have six 1.14.0 which is incompatible.
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
ERROR: convertdate 2.2.0 has requirement pytz<2020,>=2014.10, but you'll have pytz 2020.1 which is incompatible.
ERROR: albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have imgaug 0.2.9 which is incompatible.
Installing collected packages: numpy, scipy, joblib, scikit-learn, six, python-dateutil, pytz, pandas, tqdm, shap
Successfully installed joblib-0.14.1 numpy-1.18.4 pandas-1.0.3 python-dateutil-2.8.1 pytz-2020.1 scikit-learn-0.22.2.post1 scipy-1.4.1 sh
WARNING: The following packages were previously imported in this runtime:
  [dateutil,joblib,numpy,pandas,pytz,scipy,six,sklearn,tqdm]
You must restart the runtime in order to use newly installed versions.
```

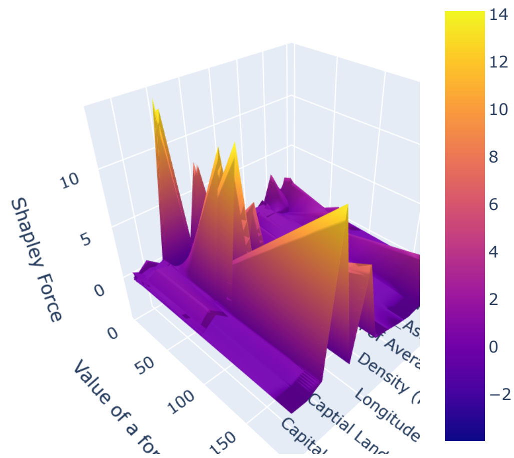
RESTART RUNTIME

```
# Local Interpretation using SHAP (for prediction at State # = 4, row 32)
import shap
shap.initjs()
explainer = shap.TreeExplainer(model2)
shap_values = explainer.shap_values(X_train)
i = 32
shap.force_plot(explainer.expected_value, shap_values[i], features=X_train.loc[i], feature_names=X_train.columns)
```




```
# Find Shapley Forces across the training sample i (i = 0 - 37)
processor = make_pipeline(
                          ce.OrdinalEncoder(),
                          SimpleImputer(strategy='median')
                          )

X_train_processed = processor.fit_transform(X_train)
column_names = X_train.columns
shap_values_array = pd.DataFrame(columns = column_names)

for i in range(len(y_train)):
        row = X_train.iloc[[i]]
        explainer = shap.TreeExplainer(model2)
        row_processed = processor.transform(row)
        shap_values_input = explainer.shap_values(row_processed)
        shap_values_array = np.concatenate((shap_values_array, shap_values_input), axis=0)
```

```
# Create a 3D plot of force as a function of state curve displacement from mean curve and features for validation sa
# A two feature partical dependence plot in 3D
import plotly.graph_objs as go
surface = go.Surface(x=column_names,
                     y=y_train,
                     z=shap_values_array)

layout = go.Layout(
        scene=dict(
                xaxis=dict(title= ''),
                yaxis=dict(title= 'Value of a for state'),
                zaxis=dict(title= 'Shapley Force')
)
)
fig = go.Figure(surface, layout)
fig.show()
```




```
# Recursive Feature Elimination
from sklearn.feature_selection import RFE, f_regression
from sklearn.model_selection import StratifiedKFold

rfr =     RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                          max_depth=3, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=4,
                          min_samples_split=2, min_weight_fraction_leaf=0,
                          n_estimators=36, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False)

#Selecting 7 features turns out to give maximum validation accuracy
number_selected_features = 7
rfe = RFE(rfr, n_features_to_select=number_selected_features, verbose =3)
rfe.fit(X_train,y_train)
```

```
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
RFE(estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                                    criterion='mse', max_depth=3,
                                    max_features='auto', max_leaf_nodes=None,
                                    max_samples=None, min_impurity_decrease=0.0,
                                    min_impurity_split=None, min_samples_leaf=4,
                                    min_samples_split=2,
                                    min_weight_fraction_leaf=0, n_estimators=36,
                                    n_jobs=None, oob_score=False,
                                    random_state=0, verbose=0,
                                    warm_start=False),
    n_features_to_select=7, step=1, verbose=3)
```

```
rfe_support = rfe.get_support()
rfe_feature = X_train.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
```

```
7 selected features
```

```
from sklearn.metrics import mean_squared_error, r2_score

# Coefficient of determination r2 for the training set
pipeline_score = rfe.score(X_train,y_train)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)
```

```
# Coefficient of determination r2 for the validation set
pipeline_score = rfe.score(X_val,y_val)
print("Coefficient of determination r2 for the validation set.: ", pipeline_score)

# The mean squared error
y_pred = rfe.predict(X_val)
print("Mean squared error: %.2f"% mean_squared_error(y_val, y_pred))
```

```
Coefficient of determination r2 for the training set.:  0.27619181813440996
Coefficient of determination r2 for the validation set.:  0.14111881952867034
Mean squared error: 489.58
```

```
# Retain only features with highest importance from RFE
X_train_rfe_select = X_train[rfe_feature]
X_val_rfe_select = X_val[rfe_feature]
print('Shape after removing features:', X_train_rfe_select.shape, X_val_rfe_select.shape)
```

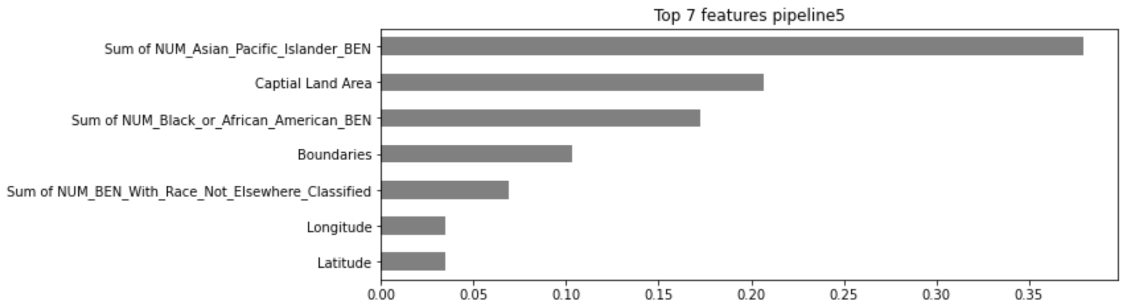
```
Shape after removing features: (38, 7) (13, 7)
```

```
# Random forest classifier after RFE Feature Selection on Reduced Feature Set

pipeline5 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                          max_depth=3, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=4,
                          min_samples_split=2, min_weight_fraction_leaf=0,
                          n_estimators=36, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False)
)

# Fit on train, score on val
pipeline5.fit(X_train_rfe_select, y_train);
```

```
# Coefficient of determination r2 for the training set
pipeline_score = pipeline5.score(X_train_rfe_select,y_train)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)

# Coefficient of determination r2 for the validation set
pipeline_score = pipeline5.score(X_val_rfe_select,y_val)
print("Coefficient of determination r2 for the validation set.: ", pipeline_score)

# The mean squared error
y_pred = pipeline5.predict(X_val_rfe_select)
print("Mean squared error: %.2f"% mean_squared_error(y_val, y_pred))
```

```
Coefficient of determination r2 for the training set.:  0.27619181813440996
Coefficient of determination r2 for the validation set.:  0.14111881952867034
Mean squared error: 489.58
```

```
pipeline5.fit(X_val_rfe_select, y_val)
# Plot of features
%matplotlib inline
import matplotlib.pyplot as plt

# Get feature importances
encoder = pipeline5.named_steps['onehotencoder']
encoded = encoder.transform(X_val_rfe_select)
rf = pipeline5.named_steps['randomforestregressor']
importances3 = pd.Series(rf.feature_importances_, encoded.columns)

# Plot feature importances
n = number_selected_features
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features pipeline5')
importances3.sort_values()[-n:].plot.barh(color='grey');
```


Top 7 features pipeline5
Sum of NUM_Asian_Pacific_Islander_BEN
Captial Land Area
Sum of NUM_Black_or_African_American_BEN
Boundaries
Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified
Longitude
Latitude
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35