```
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	onVentilatorCurrently	onVentilatorCumulative	recovered	hash	dateChecked	death	hospitalized	total	totalTestResults
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	NaN	NaN	NaN	NaN	261.0	8915b2653b57fc004eaa5369881343ee1f984aa8	2020-05-02T20:00:00Z	9.0	NaN	21399.0	21399.0
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	335.0	NaN	195.0	NaN	884ad5ab757f3861f3c36c9eef2e2c0b64d855f1	2020-05-02T20:00:00Z	288.0	1023.0	92209.0	92209.0
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	NaN	NaN	20.0	85.0	1987.0	1899b014ba10b1308db7840e2478b1d9921c10af	2020-05-02T20:00:00Z	73.0	414.0	51582.0	51582.0
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	29106909f7aae107663c3942d63c79cd882acc89	2020-05-02T20:00:00Z	0.0	NaN	57.0	57.0
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	291.0	NaN	198.0	NaN	1565.0	0eed79b0025a0cc44b30dfb01e4b3bb5a1148a5c	2020-05-02T20:00:00Z	348.0	1339.0	77997.0	77997.0

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
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0

```
# 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()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489

```
# 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
```

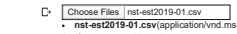
date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087

```
# 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()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent	recovered_percent
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143

```
# 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()
```

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



- **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_positive	hospitalized_percent	recovered_percent	death_percent	population
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN	3.874092	4903185.0
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453	2.164887	3017804.0
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143	4.160689	7278717.0

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

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0	0.000499
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN	3.874092	4903185.0	0.001516
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453	2.164887	3017804.0	0.001117
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN	NaN	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143	4.160689	7278717.0	0.001149

```
# 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
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm	hospitalized_norm
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0	0.000499	0.000014
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN	3.874092	4903185.0	0.001516	0.000209
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453	2.164887	3017804.0	0.001117	0.000137
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143	4.160689	7278717.0	0.001149	0.000184

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

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm	hospitalized_norm	recovered_norm
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0	0.000499	0.000014	0.000357
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN	3.874092	4903185.0	0.001516	0.000209	NaN
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453	2.164887	3017804.0	0.001117	0.000137	0.000658
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143	4.160689	7278717.0	0.001149	0.000184	0.000215

```
# 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_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm	hospitalized_norm	recovered_norm	death_norm
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0	0.000499	0.000014	0.000357	0.000012
2020-05-02	AL	7434.0	84775.0	NaN	NaN	1023.0	NaN	288.0	92209.0	8.062120	13.761098	NaN	3.874092	4903185.0	0.001516	0.000209	NaN	0.000059
2020-05-02	AR	3372.0	48210.0	NaN	95.0	414.0	1987.0	73.0	51582.0	6.537164	12.277580	58.926453	2.164887	3017804.0	0.001117	0.000137	0.000658	0.000024
2020-05-02	AS	0.0	57.0	NaN	NaN	NaN	NaN	0.0	57.0	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-05-02	AZ	8364.0	69633.0	NaN	718.0	1339.0	1565.0	348.0	77997.0	10.723489	16.009087	18.711143	4.160689	7278717.0	0.001149	0.000184	0.000215	0.000048

```
df_drop.info()
```

```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3265 entries, 2020-05-02 to 2020-01-22
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   state                   3265 non-null   object 
 1   positive                3250 non-null   float64
 2   negative                3084 non-null   float64
 3   pending                 671 non-null    float64
 4   hospitalizedCurrently   1152 non-null   float64
 5   hospitalizedCumulative  1207 non-null   float64
 6   recovered               997 non-null    float64
 7   death                   2538 non-null   float64
 8   totalTestResults        3263 non-null   float64
 9   percent_positive        3219 non-null   float64
 10  hospitalized_percent    1822 non-null   float64
 11  recovered_percent       997 non-null    float64
 12  death_percent           2486 non-null   float64
 13  population              3073 non-null   float64
 14  positive_norm           3073 non-null   float64
 15  hospitalized_norm       1783 non-null   float64
 16  recovered_norm          913 non-null    float64
 17  death_norm              2395 non-null   float64
dtypes: float64(17), object(1)
memory usage: 564.6+ 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_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm	hospitalized_norm	recovered_norm	death_norm
2020-05-02	AK	365.0	21034.0	NaN	10.0	NaN	261.0	9.0	21399.0	1.705687	2.739726	71.506849	2.465753	731545.0	0.000499	0.000014	0.000357	0.000012
2020-05-01	AK	364.0	19961.0	NaN	25.0	NaN	254.0	9.0	20325.0	1.790898	6.868132	69.780220	2.472527	731545.0	0.000498	0.000034	0.000347	0.000012
2020-04-30	AK	355.0	18764.0	NaN	19.0	NaN	252.0	9.0	19119.0	1.856792	5.352113	70.985915	2.535211	731545.0	0.000485	0.000026	0.000344	0.000012
2020-04-29	AK	355.0	18764.0	NaN	14.0	NaN	240.0	9.0	19119.0	1.856792	3.943662	67.605634	2.535211	731545.0	0.000485	0.000019	0.000328	0.000012
2020-04-28	AK	351.0	16738.0	NaN	16.0	NaN	228.0	9.0	17089.0	2.053953	4.558405	64.957265	2.564103	731545.0	0.000480	0.000022	0.000312	0.000012

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

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_positive	hospitalized_percent	recovered_percent	death_percent	population	positive_norm	hospitalized_norm	recovered_norm	death_norm
2020-05-02	CA	52197.0	634606.0	NaN	4722.0	NaN	NaN	2171.0	686803.0	7.599996	9.046497	NaN	4.159243	39512223.0	0.001321	0.000120	NaN	0.000055
2020-05-01	CA	50442.0	604543.0	NaN	4706.0	NaN	NaN	2073.0	654985.0	7.701245	9.329527	NaN	4.109671	39512223.0	0.001277	0.000119	NaN	0.000052
2020-04-30	CA	48917.0	576420.0	NaN	4981.0	NaN	NaN	1982.0	625337.0	7.822502	10.182554	NaN	4.051761	39512223.0	0.001238	0.000126	NaN	0.000050
2020-04-29	CA	46500.0	556639.0	NaN	5011.0	NaN	NaN	1887.0	603139.0	7.709666	10.776344	NaN	4.058065	39512223.0	0.001177	0.000127	NaN	0.000048
2020-04-28	CA	45031.0	532577.0	NaN	4983.0	NaN	NaN	1809.0	577608.0	7.796118	11.065710	NaN	4.017233	39512223.0	0.001140	0.000126	NaN	0.000046

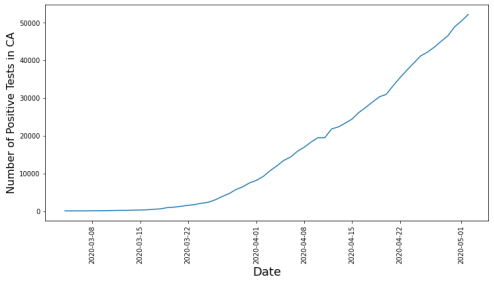
```
from matplotlib import pyplot as plt
```

```
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>
```

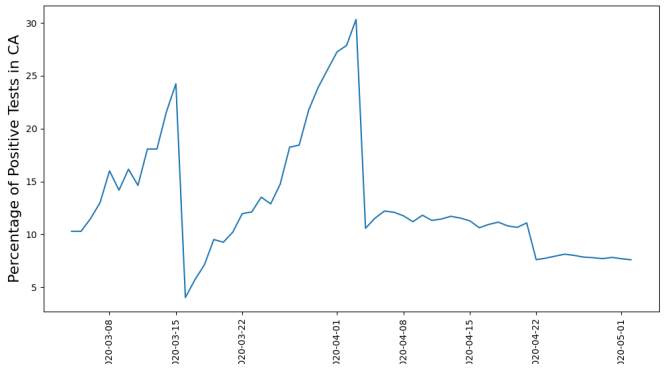



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

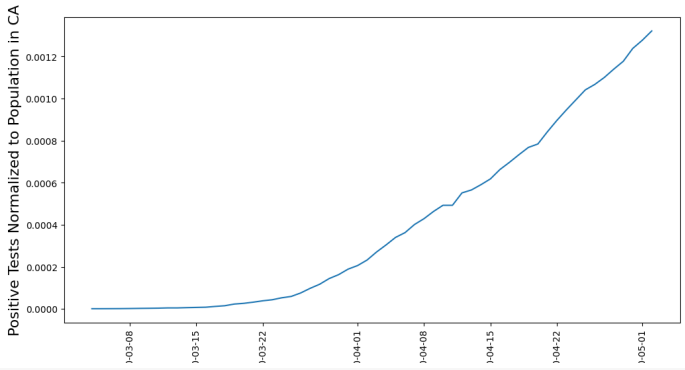



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

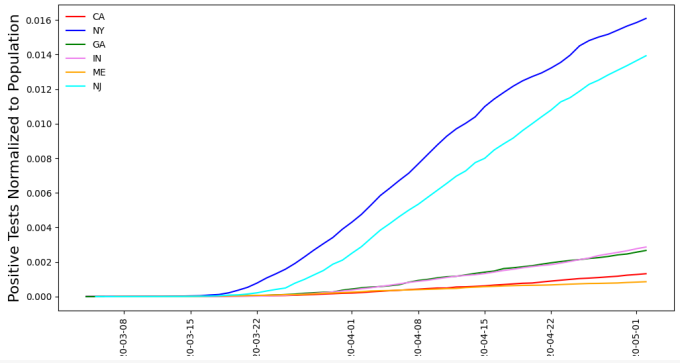



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

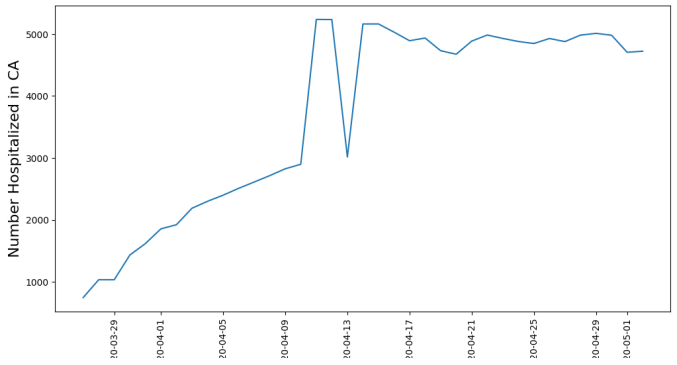



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

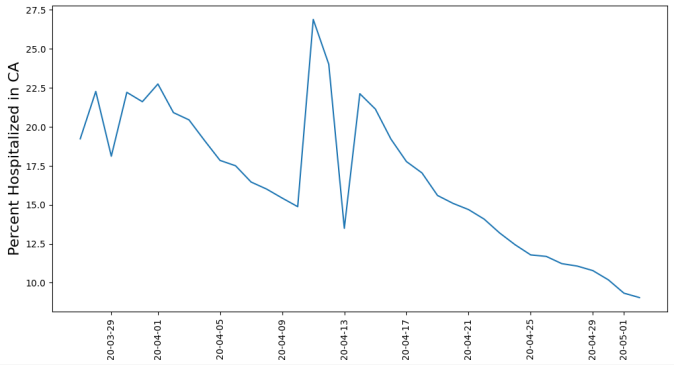



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

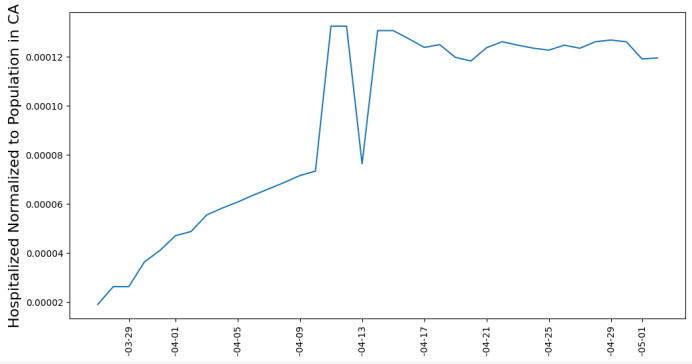



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

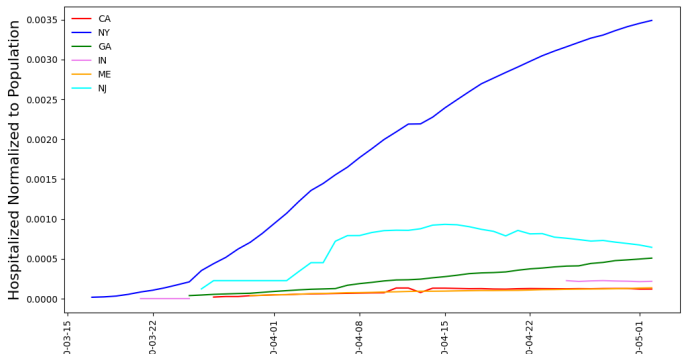



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



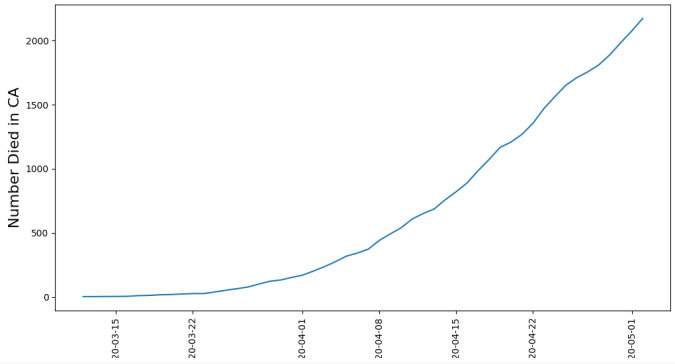
In several states, population normalized hospitalizations plateau, although population normalized death rate continues to grow.

```
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>
```

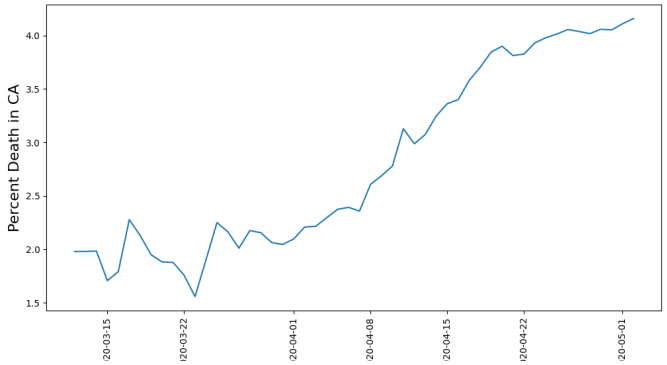



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

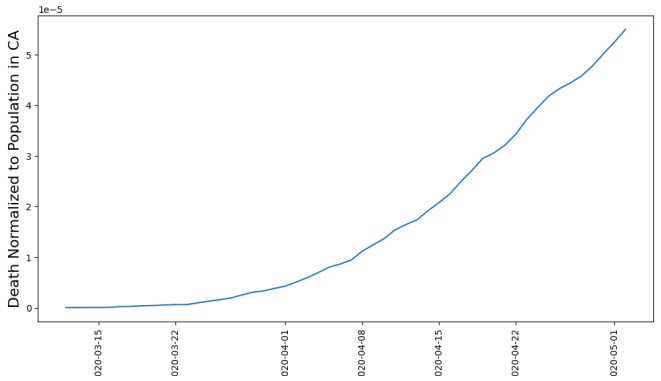



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```

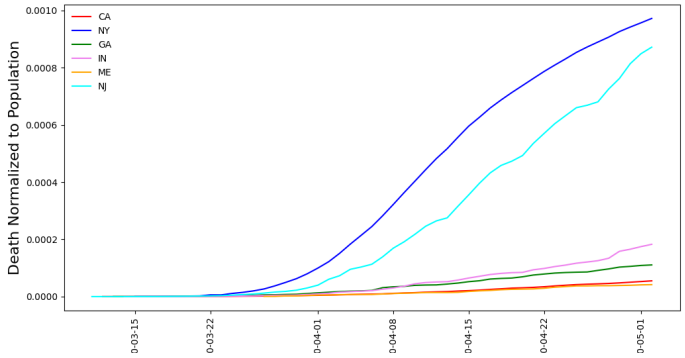



```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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) - 1695 bytes, last modified: 4/14/2020 - 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
0	AK	2.593040	-75.366476	1.0
1	AL	12.121593	-111.222242	2.0
2	AR	2.941186	-75.356785	4.0
3	AS	NaN	NaN	NaN
4	AZ	4.984063	-90.295019	1.0

```
df_state_params.describe()
```

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

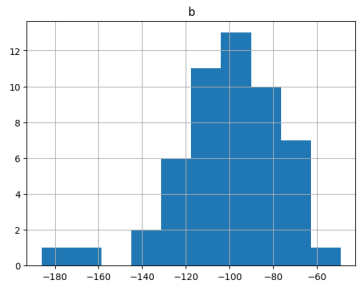
```
df_state_params.hist(column='a (10^-3)', bins=20)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7ce8cef748>]],
```

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)
```

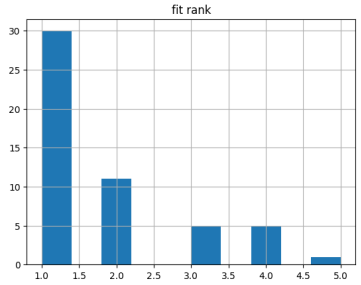
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7ce8ca6400>]],
      dtype=object)
```



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 0x7f7ce8bf2b70>]],
      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	Sum of NUM_Male_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_Native_BEN	NUM_BEN_With_
0	AK	1820384.0	270970.0	809516.0	468255.0	175296.0	1034762.0	760009.0	62311.0	76773.0	46525.0	147917.0	
1	AL	10804823.0	2065353.0	4386595.0	2980828.0	1190504.0	6237445.0	4514041.0	1549811.0	30624.0	65500.0	5556.0	
2	AR	15892716.0	2818665.0	6370265.0	4555468.0	1848506.0	9275039.0	6507151.0	1334245.0	19642.0	108428.0	62782.0	
3	AS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	AZ	10786064.0	886596.0	4861035.0	3377040.0	1294375.0	5944519.0	4747801.0	221183.0	61840.0	689880.0	179818.0	

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	Sum of NUM_Male_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_Native_BEN	NUM_BEN_With_
0	AK	1820384.0	270970.0	809516.0	468255.0	175296.0	1034762.0	760009.0	62311.0	76773.0	46525.0	147917.0	
1	AL	10804823.0	2065353.0	4386595.0	2980828.0	1190504.0	6237445.0	4514041.0	1549811.0	30624.0	65500.0	5556.0	
2	AR	15892716.0	2818665.0	6370265.0	4555468.0	1848506.0	9275039.0	6507151.0	1334245.0	19642.0	108428.0	62782.0	
3	AS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	AZ	10786064.0	886596.0	4861035.0	3377040.0	1294375.0	5944519.0	4747801.0	221183.0	61840.0	689880.0	179818.0	

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['b']
```

```
# 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	Sum of NUM_BEN_Age_Greater_84	Sum of NUM_Female_BEN	Sum of NUM_Male_BEN	Sum of NUM_Black_or_African_American_BEN	Sum of NUM_Asian_Pacific_Islander_BEN	Sum of NUM_Hispanic_BEN	NUM_Ameri
Sum of NUM_Medicare_BEN	1.000000	0.981404	0.998624	0.998100	0.989961	0.999917	0.999897	0.896692	0.525530	0.893302	
Sum of NUM_BEN_Age_Less_65	0.981404	1.000000	0.978099	0.969440	0.960650	0.982576	0.979741	0.926091	0.475021	0.827878	
Sum of NUM_BEN_Age_65_to_74	0.998624	0.978099	1.000000	0.996374	0.982712	0.998372	0.998636	0.895722	0.517514	0.902298	
Sum of NUM_BEN_Age_75_to_84	0.998100	0.969440	0.996374	1.000000	0.992601	0.997916	0.998296	0.884218	0.530001	0.899556	
Sum of NUM_BEN_Age_Greater_84	0.989961	0.960650	0.982712	0.992601	1.000000	0.989606	0.990404	0.864777	0.561253	0.879739	
Sum of NUM_Female_BEN	0.999917	0.982576	0.998372	0.997916	0.989606	1.000000	0.999658	0.899227	0.523618	0.889962	
Sum of NUM_Male_BEN	0.999897	0.979741	0.998636	0.998296	0.990404	0.999658	1.000000	0.893490	0.527005	0.895526	
Sum of NUM_Black_or_African_American_BEN	0.896692	0.926091	0.895722	0.884218	0.864777	0.899227	0.893490	1.000000	0.302985	0.726543	
Sum of NUM_Asian_Pacific_Islander_BEN	0.525530	0.475021	0.517514	0.530001	0.561253	0.523618	0.527005	0.302985	1.000000	0.633875	
Sum of NUM_Hispanic_BEN	0.893302	0.827878	0.902298	0.899556	0.879739	0.889962	0.895526	0.726543	0.633875	1.000000	
Sum of NUM_American_IndianAlaska_Native_BEN	0.082561	0.059858	0.091513	0.086836	0.065730	0.082194	0.083230	-0.035649	0.118158	0.130647	
Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	0.823477	0.774080	0.803783	0.832225	0.879096	0.821921	0.824850	0.638847	0.739901	0.734084	
Sum of NUM_Non-Hispanic_White_BEN	0.996838	0.978894	0.994391	0.996119	0.988883	0.997045	0.996745	0.888851	0.485602	0.865686	
Sum of NUM_Minorities	0.958442	0.925721	0.961095	0.957721	0.945115	0.957361	0.958590	0.868495	0.645767	0.959140	
Sum of Average_Age_of_BEN	0.682483	0.730359	0.686432	0.663590	0.637504	0.685316	0.678577	0.692434	0.132443	0.516973	
Sum of NUM_BEN_Atrial_Fibrillation	0.990425	0.969550	0.985604	0.991418	0.990376	0.990407	0.990816	0.889881	0.460369	0.851983	
Sum of NUM_BEN_Asthma	0.995532	0.979588	0.991583	0.992903	0.991762	0.995242	0.995600	0.893349	0.526980	0.880784	
Sum of NUM_BEN_Cancer	0.994765	0.972149	0.992903	0.994874	0.987401	0.994443	0.994921	0.900538	0.464158	0.883930	
Sum of NUM_BEN_Heart_Failure	0.997133	0.985150	0.995371	0.993915	0.984952	0.997171	0.996846	0.913138	0.484598	0.884681	
Sum of NUM_BEN_Chronic_Kidney_Disease	0.997501	0.980301	0.997095	0.995430	0.984274	0.997279	0.997616	0.907084	0.485414	0.893142	
Sum of NUM_BEN_Chronic_Obstructive_Pulmonary_Disease	0.986234	0.980624	0.981625	0.983999	0.978052	0.986989	0.985885	0.906585	0.429822	0.833395	
Sum of NUM_BEN_Hyperlipidemia	0.996237	0.974348	0.994742	0.996423	0.987588	0.996101	0.996491	0.903165	0.477347	0.884215	
Sum of NUM_BEN_Diabetes	0.997754	0.981227	0.996544	0.995687	0.985896	0.997745	0.997458	0.912776	0.494767	0.892805	
Sum of NUM_BEN_Hypertension	0.998856	0.982300	0.998079	0.996943	0.986018	0.998963	0.998633	0.908147	0.492704	0.886804	
Sum of NUM_BEN_Ischemic_Heart_Disease	0.994006	0.975145	0.991547	0.994105	0.985840	0.994115	0.993975	0.906294	0.457797	0.876921	
Sum of NUM_BEN_Stroke	0.990547	0.972081	0.988818	0.990024	0.980879	0.990462	0.990642	0.919103	0.447938	0.879610	
Sum of PCT_MEDICARE	0.713702	0.762102	0.716971	0.696228	0.671536	0.717742	0.709263	0.752793	0.141713	0.484719	
% Urban Pop	0.246412	0.181090	0.240984	0.259055	0.285836	0.242294	0.250798	0.181632	0.311802	0.283145	
Density (P/mi2)	-0.095479	-0.105571	-0.096280	-0.092015	-0.087655	-0.096111	-0.095250	-0.017993	-0.029294	-0.041712	
Children 0-18	0.886252	0.846604	0.876226	0.888481	0.912717	0.884787	0.887362	0.723346	0.775958	0.840004	
Adults 19-25	0.865749	0.826231	0.852680	0.868860	0.900273	0.864399	0.866800	0.698451	0.784342	0.808969	
Adults 26-34	0.848661	0.804492	0.835397	0.852982	0.886196	0.847034	0.850081	0.667717	0.811427	0.807784	
Adults 35-54	0.861684	0.820010	0.848035	0.865769	0.898830	0.860256	0.862945	0.695994	0.775924	0.803195	
Adults 55-64	0.840536	0.802214	0.822003	0.845654	0.889137	0.839342	0.841899	0.678395	0.734919	0.747404	
65+	0.842520	0.796154	0.822919	0.852028	0.896626	0.841354	0.844588	0.672577	0.691686	0.734185	
Latitude	-0.400391	-0.397373	-0.403138	-0.407192	-0.381151	-0.404042	-0.398907	-0.449230	-0.184344	-0.284522	
Longitude	0.046601	0.092974	0.034115	0.040031	0.057825	0.049875	0.043076	0.189118	-0.272166	-0.097547	
Land Area	0.229013	0.193883	0.242084	0.230058	0.205886	0.226251	0.230971	0.128422	0.201398	0.342072	
Water Area	0.042895	0.056385	0.036723	0.038782	0.050598	0.042948	0.042530	0.080106	0.048930	0.044540	
Mean Elevation	-0.163276	-0.224740	-0.147730	-0.155029	-0.169641	-0.168622	-0.157738	-0.314621	0.096312	0.038452	
Highest Elevation	-0.059881	-0.137582	-0.040603	-0.049835	-0.071530	-0.065559	-0.054634	-0.233143	0.289714	0.156542	
Lowest elevation	-0.354394	-0.352655	-0.344053	-0.355481	-0.373528	-0.356287	-0.352697	-0.312823	-0.524776	-0.283582	
Number of bordering states	0.077790	0.135863	0.075964	0.059448	0.056612	0.079630	0.074603	0.044175	-0.147225	-0.075000	
On Coast	0.471024	0.505115	0.442960	0.461862	0.518306	0.471588	0.470022	0.512059	0.172256	0.274148	
Borders Another Country	0.358143	0.310618	0.363869	0.356815	0.359742	0.351935	0.363168	0.188403	0.423258	0.500772	
Capital Latitude	-0.388663	-0.393979	-0.394070	-0.392266	-0.359329	-0.392545	-0.386636	-0.463654	-0.136997	-0.269834	
Capital Longitude	0.027375	0.076949	0.015075	0.019608	0.037346	0.030778	0.023488	0.181245	-0.297512	-0.116778	
Captial Land Area	0.008902	-0.002410	0.018688	0.009403	-0.012432	0.008939	0.008880	-0.020032	-0.012972	0.022939	
Capital Water Area	-0.087670	-0.096352	-0.083610	-0.087193	-0.092577	-0.088397	-0.087364	-0.097128	-0.020784	-0.040196	
Capital Mean Elevation	-0.194009	-0.217624	-0.182931	-0.190725	-0.206936	-0.197080	-0.190497	-0.249034	-0.121862	-0.049687	
Capital is the Largest City	-0.171080	-0.147972	-0.165860	-0.173283	-0.194417	-0.169162	-0.172323	-0.133938	-0.129514	-0.188909	
Largest City Latitude	-0.421170	-0.421496	-0.425109	-0.424938	-0.398170	-0.424688	-0.419431	-0.467601	-0.234938	-0.317480	
Largest City Longitude	0.057094	0.102104	0.044423	0.050338	0.069739	0.060248	0.053604	0.201824	-0.262204	-0.081857	
Number of Counties	0.663716	0.710105	0.670375	0.645677	0.611985	0.666592	0.659547	0.684930	0.100824	0.503011	
Became a State	-0.140415	-0.200801	-0.128869	-0.126557	-0.143131	-0.144064	-0.136075	-0.308218	0.076276	0.035901	
DaysSinceStayatHomeOrder	-0.020651	-0.019693	-0.030343	-0.027347	0.007693	-0.023631	-0.018913	-0.045827	0.221997	0.052471	
DaysSinceFirstPositive	0.368252	0.319941	0.366229	0.374653	0.390899	0.365580	0.370903	0.287102	0.259030	0.302869	
DaysSinceTestStart	0.290649	0.242592	0.289428	0.297948	0.312252	0.288578	0.292640	0.233275	0.191547	0.240467	
15-49yearsAllcauses	0.888203	0.856564	0.874919	0.889982	0.919415	0.887334	0.889058	0.739853	0.736873	0.795191	
15-49yearsAsthma	0.824682	0.787879	0.807656	0.827220	0.869382	0.823203	0.825395	0.667614	0.757078	0.750358	
15-49yearsChronickidneydisease	0.918864	0.893772	0.909568	0.918825	0.935419	0.918655	0.918932	0.805947	0.715267	0.828201	
15-49yearsChronicobstructivepulmonarydisease	0.896769	0.878089	0.880516	0.897303	0.928648	0.896643	0.896964	0.771591	0.635771	0.749901	
15-49yearsDiabetesmellitus	0.912330	0.881654	0.900896	0.914260	0.936863	0.911726	0.912809	0.781996	0.692768	0.812172	
15-49yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.881251	0.864222	0.866766	0.880121	0.909006	0.881289	0.881043	0.782863	0.644183	0.738019	
15-49yearsIschemicheartdisease	0.928387	0.927789	0.916634	0.923405	0.939571	0.929317	0.927199	0.849388	0.595981	0.764879	
15-49yearsNeoplasms	0.887461	0.860138	0.873067	0.888685	0.919369	0.886922	0.887968	0.748507	0.729958	0.785252	
15-49yearsOtherchronicrespiratorydiseases	0.906636	0.885246	0.892415	0.906637	0.934764	0.906287	0.906852	0.785146	0.652797	0.775052	
15-49yearsRheumaticheartdisease	0.903473	0.893269	0.893364	0.898792	0.916822	0.903373	0.903469	0.792144	0.690715	0.785070	
15-49yearsStroke	0.919789	0.898558	0.910295	0.919449	0.934870	0.919891	0.919744	0.808343	0.702539	0.815118	
50-69yearsAllcauses	0.880146	0.855617	0.863069	0.881923	0.918175	0.879688	0.880993	0.745024	0.677823	0.745280	
50-69yearsAsthma	0.801803	0.765502	0.781306	0.805925	0.855811	0.800503	0.802845	0.641040	0.741394	0.706390	
50-69yearsChronickidneydisease	0.917312	0.898401	0.905572	0.916416	0.938247	0.917261	0.917589	0.809479	0.675662	0.793827	
50-69yearsChronicobstructivepulmonarydisease	0.879259	0.872843	0.860771	0.878641	0.912096	0.879671	0.879770	0.765135	0.542692	0.678077	
50-69yearsDiabetesmellitus	0.882501	0.857522	0.865414	0.884673	0.920496	0.882047	0.883347	0.753933	0.653401	0.743092	
50-69yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.863191	0.840683	0.846169	0.863950	0.901043	0.862755	0.863919	0.739130	0.673635	0.725850	
50-69yearsIschemicheartdisease	0.905979	0.901073	0.890019	0.902683	0.931491	0.906505	0.905613	0.807307	0.618281	0.735980	
50-69yearsNeoplasms	0.872500	0.853408	0.854035	0.873415	0.912024	0.872273	0.873220	0.746019	0.650724	0.719312	
50-69yearsOtherchronicrespiratorydiseases	0.885021	0.875159	0.867604	0.883457	0.917418	0.885065	0.885343	0.780290	0.570425	0.701549	
50-69yearsRheumaticheartdisease	0.892519	0.890373	0.880528	0.886667	0.908250	0.892709	0.892705	0.793842	0.640749	0.737680	
50-69yearsStroke	0.907993	0.892290	0.895108	0.907388	0.930491	0.908246	0.908324	0.800720	0.656820	0.764139	
70+yearsAllcauses	0.849263	0.819400	0.828488	0.854118	0.899821	0.848669	0.850601	0.700955	0.653903	0.703444	
70+yearsAsthma	0.791486	0.748032	0.769602	0.799338	0.851826	0.790234	0.793100	0.612235	0.747127	0.697428	
70+yearsChronickidneydisease	0.877077	0.858325	0.859219	0.877628	0.915494	0.876848	0.877925	0.751777	0.643149	0.723413	
70+yearsChronicobstructivepulmonarydisease	0.866728	0.842585	0.846829	0.871197	0.911389	0.866460	0.868173	0.721373	0.599864	0.698688	
70+yearsDiabetesmellitus	0.845276	0.815442	0.823824	0.850785	0.897232	0.844780	0.846627	0.702717	0.638346	0.695439	
70+yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.833832	0.799993	0.814083	0.839080	0.884501	0.833012	0.835543	0.675650	0.689192	0.705929	
70+yearsIschemicheartdisease	0.841243	0.819787	0.818295	0.844148	0.893641	0.840980	0.841899	0.706345	0.634076	0.681143	
70+yearsNeoplasms	0.837485	0.808405	0.816021	0.842466	0.889888	0.836924	0.838922	0.689286	0.647903	0.686282	
70+yearsOtherchronicrespiratorydiseases	0.875916	0.859545	0.858192	0.875620	0.913137	0.875455	0.876936	0.752057	0.586677	0.704251	
70+yearsRheumaticheartdisease	0.844465	0.839621	0.826738	0.839480	0.879184	0.843959	0.845500	0.720717	0.626171	0.679448	
70+yearsStroke	0.871562	0.849583	0.854254	0.873252	0.910139	0.871322	0.872668	0.732812	0.659665	0.720810	

AllAgesAllcauses	0.880003	0.851293	0.863399	0.882592	0.918634	0.879356	0.880977	0.736553	0.695184	0.757560
AllAgesAsthma	0.833253	0.794917	0.815812	0.836910	0.879310	0.831811	0.834191	0.674054	0.748697	0.754660
AllAgesChronickidneydisease	0.905462	0.885503	0.891504	0.905307	0.933093	0.905311	0.905943	0.788948	0.671941	0.774120
AllAgesChronicobstructivepulmonarydisease	0.877214	0.860833	0.858127	0.879265	0.916074	0.877240	0.878179	0.745966	0.580626	0.696350
AllAgesDiabetesmellitus	0.879728	0.852121	0.862209	0.882908	0.919854	0.879232	0.880670	0.745944	0.658313	0.743480
AllAgesInterstitiallungdiseaseandpulmonarysarcoidosis	0.853912	0.826093	0.835749	0.856759	0.897256	0.853326	0.855050	0.714595	0.681169	0.720490
AllAgesIschemicheartdisease	0.883535	0.870954	0.864460	0.883069	0.920941	0.883720	0.883584	0.768383	0.628518	0.717687
AllAgesNeoplasms	0.865325	0.841450	0.846325	0.867726	0.908429	0.864932	0.866273	0.728920	0.662908	0.719608
AllAgesOtherchronicrespiratorydiseases	0.903592	0.885967	0.888445	0.902970	0.932994	0.903300	0.904011	0.785664	0.622884	0.752887
AllAgesRheumaticheartdisease	0.880357	0.875286	0.866144	0.875089	0.904080	0.880168	0.880883	0.767398	0.647527	0.727462
AllAgesStroke	0.895398	0.875735	0.880674	0.895978	0.925120	0.895381	0.896074	0.771183	0.666743	0.752509
AllAgesTotal	0.880507	0.853923	0.863463	0.882813	0.919294	0.879978	0.881411	0.740657	0.683664	0.750268
Airpollution	0.889229	0.888442	0.875092	0.882964	0.910652	0.889624	0.888991	0.780168	0.654775	0.729374
Highbody-massindex	0.893797	0.872739	0.877133	0.894624	0.928715	0.893524	0.894332	0.771258	0.660813	0.753636
Highfastingplasmaglucose	0.886795	0.870124	0.868909	0.887417	0.922985	0.886779	0.887281	0.773198	0.617042	0.724166
HighLDLcholesterol	0.893215	0.882483	0.875398	0.892023	0.926304	0.893527	0.893124	0.783354	0.627247	0.727214
Highsystolicbloodpressure	0.897453	0.882631	0.880346	0.897131	0.930464	0.897547	0.897670	0.788283	0.638519	0.741135
Impairedkidneyfunction	0.889934	0.872693	0.873173	0.890034	0.923835	0.889974	0.890321	0.773115	0.657798	0.740044
Noaccesstohandwashingfacility	0.877603	0.857781	0.862453	0.876519	0.909499	0.877112	0.877940	0.756686	0.667841	0.744063
Smoking	0.881579	0.866726	0.862831	0.882612	0.919020	0.881762	0.882099	0.760942	0.604036	0.705752
Log10Pop	0.728494	0.737902	0.714057	0.722320	0.747129	0.729527	0.727871	0.666572	0.421521	0.509917
DaysSinceInfection	0.422525	0.373010	0.419727	0.431233	0.443999	0.420775	0.424402	0.365316	0.263022	0.351935
Children0-18	0.167133	0.180823	0.181296	0.159580	0.119098	0.169559	0.164119	0.169612	0.049811	0.148864
Allriskfactors	0.882815	0.860944	0.865530	0.884217	0.920134	0.882487	0.883653	0.750594	0.660552	0.737247
State Area Ratio	-0.141342	-0.180449	-0.126323	-0.134563	-0.158168	-0.143787	-0.137979	-0.260308	-0.103905	-0.027420
Elevation Ratio	0.020332	0.007311	0.029598	0.023691	-0.000223	0.021715	0.018482	0.129890	0.052935	0.012281
Capital Area Ratio	-0.119284	-0.151665	-0.109968	-0.112407	-0.124967	-0.120614	-0.117747	-0.171038	-0.076449	-0.052022
Boundaries	0.499356	0.556393	0.479330	0.477960	0.517125	0.499487	0.497522	0.455272	0.125465	0.277110
Latitude Difference to State Capital	-0.268652	-0.211068	-0.252026	-0.293417	-0.327548	-0.268822	-0.271605	-0.115573	-0.406313	-0.233715
Longitude Difference to State Capital	-0.143646	-0.133106	-0.139285	-0.150250	-0.155036	-0.143567	-0.145263	-0.100819	-0.106975	-0.108075
Latitude Difference to DC	-0.400391	-0.397373	-0.403138	-0.407192	-0.381151	-0.404042	-0.398907	-0.449230	-0.184344	-0.284522
Longitude Difference to DC	-0.046601	-0.092974	-0.034115	-0.040031	-0.057825	-0.049875	-0.043076	-0.189118	0.272166	0.097547
Latitude Difference to Center	-0.400391	-0.397373	-0.403138	-0.407192	-0.381151	-0.404042	-0.398907	-0.449230	-0.184344	-0.284522
Longitude Difference to Center	-0.046601	-0.092974	-0.034115	-0.040031	-0.057825	-0.049875	-0.043076	-0.189118	0.272166	0.097547

```
#  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_Native_BEN	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	Sum of Average_Age_of_BEN	Sum of PCT_MEDICARE	% Urban Pop	Density (P/mi2)	Children 0-18	Latitude	Longitude	Land Area
0	1820384.0	62311.0	76773.0	46525.0	147917.0	23372.0	996.298679	10.069041	66.0	1.2863	181405.17	61.370716	-152.404419	570665.0
1	10804823.0	1549811.0	30624.0	65500.0	5556.0	58660.0	3967.220634	51.254704	59.0	96.9221	1105570.08	32.806671	-86.791130	50644.0
2	15892716.0	1334245.0	19642.0	108428.0	62782.0	61250.0	3928.834167	94.570949	56.2	58.4030	686482.50	34.969704	-92.373123	52030.0
4	10786064.0	221183.0	61840.0	689880.0	179818.0	114903.0	1009.367955	14.075942	89.8	64.9550	1744612.56	33.729759	-111.431221	113595.0
5	42579588.0	2072012.0	3276415.0	5674776.0	113871.0	562214.0	4001.853612	63.398334	95.0	256.3727	9481941.36	36.116203	-119.681564	155766.0

```
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	Sum of NUM_American_IndianAlaska_Native_BEN	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	Sum of Average_Age_of_BEN	Sum of PCT_MEDICARE	% Urban Pop	Density (P/mi2)	Children 0-18	Latitude	Longitu
count	5.100000e+01	5.100000e+01	5.100000e+01	5.100000e+01	51.000000	51.000000	51.000000	51.000000	51.000000	51.000000	5.100000e+01	51.000000	51.000
mean	1.038431e+07	9.464777e+05	1.411691e+05	5.310095e+05	38355.372549	86379.019608	3535.191553	52.223011	74.107843	431.560508	1.486858e+06	39.464823	-93.339
std	1.311026e+07	1.274593e+06	4.722330e+05	1.629961e+06	87337.002647	114765.665446	2518.178494	46.520870	14.885481	1647.225920	1.764935e+06	6.069546	19.288
min	1.655870e+05	2.960000e+02	1.660000e+02	4.130000e+02	0.000000	1693.000000	70.002893	0.972106	38.700000	1.286300	1.160123e+05	21.094318	-157.498
25%	2.252305e+06	5.366600e+04	6.445500e+03	3.101950e+04	2980.500000	17674.500000	1542.140834	13.385073	65.400000	50.604850	3.984744e+05	35.688955	-102.547
50%	6.272609e+06	3.156040e+05	2.579200e+04	1.042170e+05	7061.000000	58660.000000	3578.360041	51.254704	74.200000	108.049700	1.013513e+06	39.849426	-89.616
75%	1.471830e+07	1.547566e+06	7.063400e+04	2.005865e+05	28506.500000	100449.000000	5214.099778	78.017975	87.550000	223.983100	1.719581e+06	43.041292	-78.988
max	7.644909e+07	7.011107e+06	3.276415e+06	1.007620e+07	560433.000000	562214.000000	13644.965980	219.756971	100.000000	11814.541000	9.481941e+06	61.370716	-69.381

```
# 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	Sum of NUM_American_IndianAlaska_Native_BEN	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	Sum of Average_Age_of_BEN	Sum of PCT_MEDICARE	% Urban Pop	Density (P/mi2)	Children 0-18	Latitude	Longitu
count	3.800000e+01	3.800000e+01	3.800000e+01	3.800000e+01	38.000000	38.000000	38.000000	38.000000	38.000000	38.000000	3.800000e+01	38.000000	38.000
mean	1.014125e+07	9.685705e+05	1.623107e+05	3.942231e+05	37676.868421	93816.710526	3549.919212	51.835362	75.578947	509.044824	1.536694e+06	39.769502	-94.847
std	9.963253e+06	1.001560e+06	5.333709e+05	1.021129e+06	93507.556838	120712.415819	2056.853166	40.781616	14.338667	1899.378670	1.707713e+06	6.596259	20.534
min	3.472690e+05	2.689000e+03	4.580000e+02	2.622000e+03	45.000000	4201.000000	70.002893	0.972106	38.700000	1.286300	1.365427e+05	21.094318	-157.498
25%	2.518838e+06	4.934350e+04	1.427175e+04	3.676725e+04	4699.500000	19802.500000	1590.459797	18.987497	66.125000	43.927350	5.296779e+05	35.839934	-103.929
50%	7.473651e+06	5.120990e+05	3.068000e+04	1.071920e+05	9615.500000	62351.000000	3948.027401	53.017662	74.850000	112.688450	1.123149e+06	40.249745	-89.647
75%	1.563758e+07	1.560497e+06	9.455175e+04	1.983508e+05	27502.250000	100987.500000	5194.290279	79.721735	87.975000	249.285600	1.728603e+06	43.183955	-80.276
max	4.257959e+07	3.265865e+06	3.276415e+06	5.674776e+06	560433.000000	562214.000000	7981.626181	185.232318	100.000000	11814.541000	9.481941e+06	61.370716	-69.381

```
# Optimizing Hyperparameters
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

# Define classifier
forest = RandomForestRegressor(random_state = 1)

# Parameters to fit

max_depth = [0.95, 1.0, 1.05]
n_estimators = [16, 18, 20]
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, 0.05, 0.1]
min_weight_fraction_leaf = [0.0, 0.05, 0.1]

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:    1.5s
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed:    1.5s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1900s.) Setting batch_size=2.
[Parallel(n_jobs=-1)]: Batch computation too fast (0.0551s.) Setting batch_size=4.
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1051s.) Setting batch_size=8.
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 130 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done 202 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 290 tasks      | elapsed:    4.7s
[Parallel(n_jobs=-1)]: Done 378 tasks      | elapsed:    5.9s
[Parallel(n_jobs=-1)]: Done 482 tasks      | elapsed:    7.0s
[Parallel(n_jobs=-1)]: Done 586 tasks      | elapsed:    8.1s
[Parallel(n_jobs=-1)]: Done 706 tasks      | elapsed:    9.6s
[Parallel(n_jobs=-1)]: Done 826 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done 962 tasks      | elapsed:   12.5s
[Parallel(n_jobs=-1)]: Done 1098 tasks      | elapsed:   14.1s
[Parallel(n_jobs=-1)]: Done 1250 tasks      | elapsed:   15.6s
[Parallel(n_jobs=-1)]: Done 1402 tasks      | elapsed:   17.5s
[Parallel(n_jobs=-1)]: Done 1570 tasks      | elapsed:   19.3s
[Parallel(n_jobs=-1)]: Done 1738 tasks      | elapsed:   21.2s
[Parallel(n_jobs=-1)]: Done 1922 tasks      | elapsed:   23.4s
[Parallel(n_jobs=-1)]: Done 2106 tasks      | elapsed:   25.5s
The score achieved with the best parameters =  0.026269551100730 52

The parameters are: {'ccp_alpha': 0.0, 'max_depth': 1.0, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 18}
[Parallel(n_jobs=-1)]: Done 2187 out of 2187 | elapsed:   26.3s finished
```

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

```
Collecting category_encoders==2.0.0
  Downloading https://files.pythonhosted.org/packages/6e/a1/f7a22f144f33be78afeb06bfa78478e8284a64263a3c09b1ef54e673841e/category_encoders-2.0.0-py2.py3-none-any.whl (87kB)
    |████████████████████████████████| 92kB 5.3MB/s
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: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (1.4.1)
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.post1)
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: statsmodels>=0.6.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.10.2)
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: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.20.0->category_encoders==2.0.0) (0.14.1)
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: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21.1->category_encoders==2.0.0) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21.1->category_encoders==2.0.0) (2.8.1)
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=1, 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=18, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False))

pipeline1.fit(X_train, y_train)

# Get the model's training accuracy
print("Training Accuracy:  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))
```

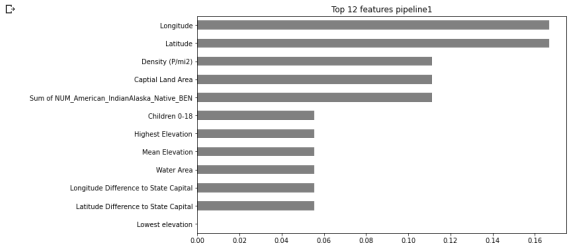
```
Training Accuracy:  R^2 =  0.38074038406249433
Validation Accuracy:  R^2 =  -0.18421766224718805
```

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

```
Feature Importances =
[0.         0.         0.         0.         0.11111111 0.
 0.         0.         0.         0.11111111 0.05555556 0.16666667
 0.16666667 0.         0.05555556 0.05555556 0.05555556 0.
 0.         0.         0.         0.11111111 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.05555556 0.05555556]
```

```
# Plot of feature importances from pure Random Forest Regressor
```

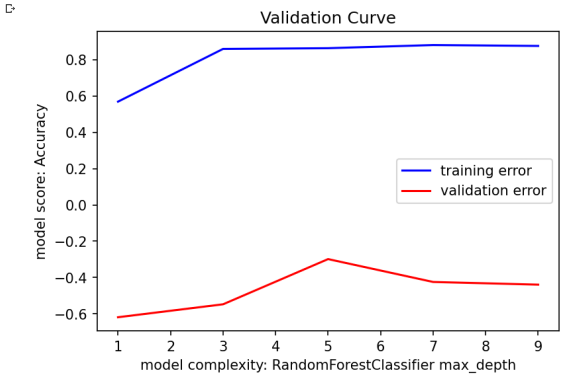
```
%matplotlib inline
import matplotlib.pyplot as plt
# Get feature importances
encoder = pipeline1.named_steps['onehotencoder']
encoded = encoder.transform(X_train)
rf = pipeline1.named_steps['randomforestregressor']
importances1 = pd.Series(rf.feature_importances_, encoded.columns)
# Plot feature 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 = 'Latitude'

pipeline3 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0, criterion='mse',
                      max_depth=1, 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=18, 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 Latitude: -0.3777862174242266
Validation Accuracy with Latitude: -0.18421766224718805
Drop-Column Importance for Latitude: 0.19356855517703853
```

```
# 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=1, 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=18, 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=1,
                      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=18, 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.py3-none-any.whl (105kB)
     |████████████████████████████████| 112kB 8.4MB/s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from eli5) (1.12.0)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (19.3.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (1.18.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from eli5) (2.11.2)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.22.2.post1)
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: graphviz in /usr/local/lib/python3.6/dist-packages (from eli5) (0.10.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2->eli5) (1.1.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.18->eli5) (0.14.1)
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 in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anythi
  warnings.warn(message, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.feature_selection.base module is  deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.feature
  warnings.warn(message, FutureWarning)
Using TensorFlow backend.
```

Weight	Feature
0.1227 ± 0.0096	Latitude
0.0115 ± 0.0067	Water Area
0 ± 0.0000	Number of bordering states
0 ± 0.0000	Capital Water Area
0 ± 0.0000	Sum of NUM_Black_or_African_American_BEN
0 ± 0.0000	Sum of NUM_Asian_Pacific_Islander_BEN
0 ± 0.0000	Sum of NUM_Hispanic_BEN
0 ± 0.0000	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified
0 ± 0.0000	Sum of Average_Age_of_BEN
0 ± 0.0000	Sum of PCT_MEDICARE
0 ± 0.0000	% Urban Pop
0 ± 0.0000	Land Area
0 ± 0.0000	Lowest elevation
0 ± 0.0000	On Coast
0 ± 0.0000	Borders Another Country
0 ± 0.0000	Sum of NUM_Medicare_BEN
0 ± 0.0000	Boundaries
0 ± 0.0000	Capital Area Ratio
0 ± 0.0000	State Area Ratio
0 ± 0.0000	Children0-18
0 ± 0.0000	DaysSinceInfection
0 ± 0.0000	Log10Pop
0 ± 0.0000	Capital Mean Elevation
0 ± 0.0000	DaysSinceTestStart
0 ± 0.0000	Elevation Ratio
0 ± 0.0000	DaysSinceFirstPositive
0 ± 0.0000	DaysSinceStayatHomeOrder
0 ± 0.0000	Became a State
0 ± 0.0000	Capital is the Largest City
-0.0131 ± 0.0342	Children 0-18
-0.0139 ± 0.0013	Highest Elevation
-0.0252 ± 0.2914	Sum of NUM_American_IndianAlaska_Native_BEN
-0.0280 ± 0.0435	Mean Elevation
-0.0301 ± 0.0306	Density (P/mi2)
-0.0301 ± 0.0403	Latitude Difference to State Capital
-0.0384 ± 0.0006	Longitude Difference to State Capital
-0.1409 ± 0.0783	Longitude
-0.1920 ± 0.0820	Captial Land Area

```
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.:  0.38074038406249433
Coefficient of determination r2 for the validation set.:  -0.18421766224718805
Mean squared error: 304.83
```

```
# Thus, Sum of NUM_American_IndianAlaska_Native_BEN is way more important according to feature permutation than acco
# Use importances for feature selection
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, 2)
```

```
# Random forest classifier with two 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=1, 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=18, 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))
```

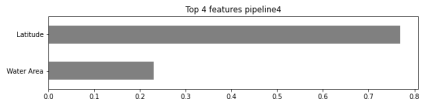
```
Coefficient of determination r2 for the training set.:  0.27110557327427665
Coefficient of determination r2 for the validation set.:  0.1883085673181527
Mean squared error: 208.94
```

```
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
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.gz (57.7MB)
     |████████████████████████████████| 57.7MB 63kB/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: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->pdpbox) (2018.9)
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: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.1.2->pdpbox) (0.10.0)
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->pdpbox) (2.4.7)
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=ac4abc84618b4c897cbef0d6aba3fdbe6824f4ccbf25b371e32bd60883f0a52e
  Stored in directory: /root/.cache/pip/wheels/7d/08/51/63fd122b04a2c87d780464eeffb94867c75bd96a64d500a3fe
Successfully built pdpbox
Installing collected packages: pdpbox
Successfully installed pdpbox-0.2.0
```

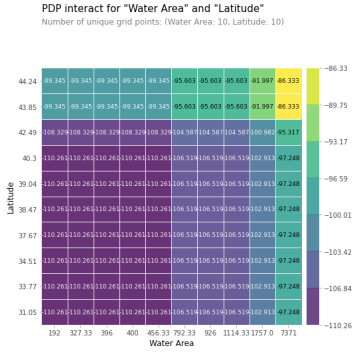
```
model2 =     RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                           max_depth=1, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=4,
                           min_samples_split=2, min_weight_fraction_leaf=0,
                           n_estimators=18, 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=1,
                      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=18, 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 = ['Water Area', 'Latitude']
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.
```



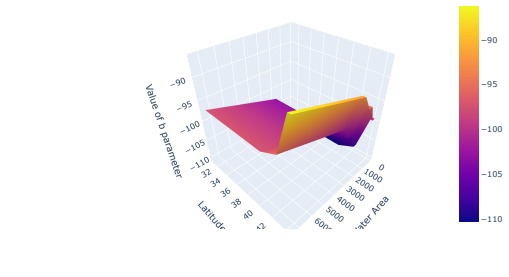

```
# A two feature partical dependence plot in 3D
pdp = interaction.pdp.pivot_table(
                                values='preds',
                                columns=features2[0],
                                index=features2[1]
                                )[::-1] # Slice notation to reverse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

target = 'Value of b 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()
```

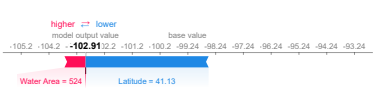


```
! pip install shap==0.23.0
! pip install -I shap
```

```
Collecting shap==0.23.0
  Downloading https://files.pythonhosted.org/packages/60/0d/8bd076821f7230edb2892ad982ea91ca25f2f925466563272e61eae891c6/shap-0.23.0.tar.gz (182kB)
     |████████████████████████████████| 184kB 8.7MB/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: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (2.4.7)
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: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (0.10.0)
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: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->shap==0.23.0) (2018.9)
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (0.7.5)
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: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (46.1.3)
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) (1.0.18)
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: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.8.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (2.1.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.1->matplotlib->shap==0.23.0) (1.12.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->shap==0.23.0) (0.1.9)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2->ipython->shap==0.23.0) (0.2.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from pexpect; sys_platform != "win32"->ipython->shap==0.23.0) (0.6.0)
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=235673 sha256=d7b19f033dda0f93a8e92001ec7f7969c37a80aa117f0a6c852589db2460d78d
  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.gz (273kB)
     |████████████████████████████████| 276kB 9.6MB/s
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/03/27/e35e7c6e6a52fab9fcc64fc2b20c6b516eba930bb02b10ace3b38200d3ab/numpy-1.18.4-cp36-cp36m-manylinux1_x86_64.whl (20.2MB)
     |████████████████████████████████| 20.2MB 67.9MB/s
Collecting scipy
  Downloading https://files.pythonhosted.org/packages/dc/29/162476fd44203116e7980cfbd9352eef9db37c49445d1fec35509022f6aa/scipy-1.4.1-cp36-cp36m-manylinux1_x86_64.whl (26.1MB)
     |████████████████████████████████| 26.1MB 1.5MB/s
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/5e/d8/312e03adf4c78663e17d802fe2440072376fee46cada1404f1727ed77a32/scikit_learn-0.22.2.post1-cp36-cp36m-manylinux1_x86_64.whl (7.1MB)
     |████████████████████████████████| 7.1MB 49.5MB/s
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/bb/71/8f53bdbcbc67c912b888b40def255767e475402e9df64050019149b1a943/pandas-1.0.3-cp36-cp36m-manylinux1_x86_64.whl (10.0MB)
     |████████████████████████████████| 10.0MB 47.6MB/s
Collecting tqdm>4.25.0
  Downloading https://files.pythonhosted.org/packages/c9/40/058b12e8ba10e35f89c9b1fdfc2d4c7f8c05947df2d5eb3c7b258019fda0/tqdm-4.46.0-py2.py3-none-any.whl (63kB)
     |████████████████████████████████| 71kB 9.3MB/s
Collecting joblib>=0.11
  Downloading https://files.pythonhosted.org/packages/28/5c/cf6a2b65a321c4a209efcdf64c2689efae2cb62661f8f6f4bb28547cf1bf/joblib-0.14.1-py2.py3-none-any.whl (294kB)
     |████████████████████████████████| 296kB 37.8MB/s
Collecting python-dateutil>=2.6.1
  Downloading https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b306af2bce5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any.whl (227kB)
     |████████████████████████████████| 235kB 52.0MB/s
Collecting pytz>=2017.2
  Downloading https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e59d7d4efda580b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.py3-none-any.whl (510kB)
     |████████████████████████████████| 512kB 25.6MB/s
Collecting six>=1.5
  Downloading https://files.pythonhosted.org/packages/65/eb/1f97cb97bfc2390a276969c6fae16075da282f5058082d4cb10c6c5c1dba/six-1.14.0-py2.py3-none-any.whl
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=394119 sha256=a99f99f9e861b9a391c9d681aee8f92cdfe6bbe4b5791f90711dc7f764ce57ac
  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 shap-0.35.0 six-1.14.0 tqdm-4.46.0
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)
```



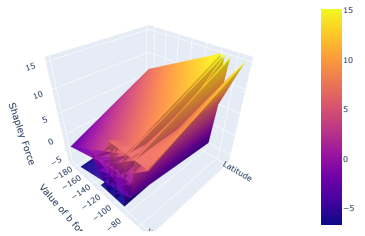

```
# Find Shapley Forces across the training sample i (i = 0 - 37)
processor = make_pipeline(
                        ce.OrdinalEncoder(),
                        SimpleImputer(strategy='median')
                        )

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 b 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=1, 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=18, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False)

#Selecting 2 features turns out to give maximum validation accuracy
number_selected_features = 2
rfe = RFE(rfr, n_features_to_select=number_selected_features, verbose =3)
rfe.fit(X_train,y_train)
```

```
RFE(estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                                    criterion='mse', max_depth=1,
                                    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=18,
                                    n_jobs=None, oob_score=False,
                                    random_state=0, verbose=0,
                                    warm_start=False),
        n_features_to_select=2, 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')
```

```
2 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.:  0.27110557327427665
Coefficient of determination r2 for the validation set.:  0.1883085673181527
Mean squared error: 208.94
```

```
# 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, 2) (13, 2)
```

```
# 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=1, 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=18, 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.:  0.27110557327427665
Coefficient of determination r2 for the validation set.:  0.1883085673181527
Mean squared error: 208.94
```

```
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');
```