```
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()
```

```
Choose Files | nst-est2019-01.csv
• nst-est2019-01.csv(application/vnd.ms-excel) - 676 bytes, last modified: 4/13/2020 - 100% done
Saving nst-est2019-01.csv to nst-est2019-01.csv
```

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

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

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

df_drop["population"] = ""

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

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

df_drop.head()
```

date	state	positive	negative	pending	hospitalizedCurrently	hospitalizedCumulative	recovered	death	totalTestResults	percent_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

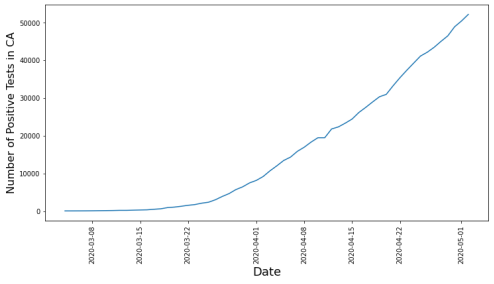
```
df_state_dict['CA'].head()
```

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
```

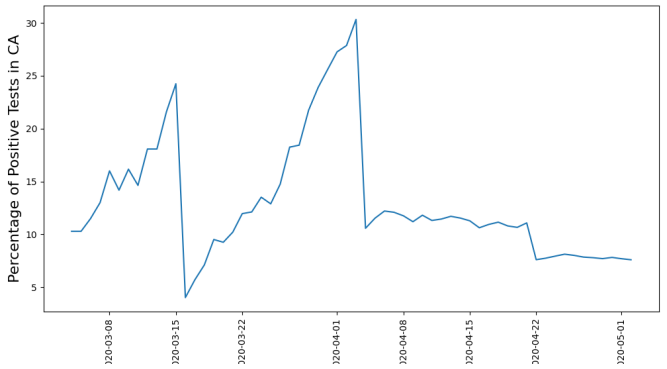
```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



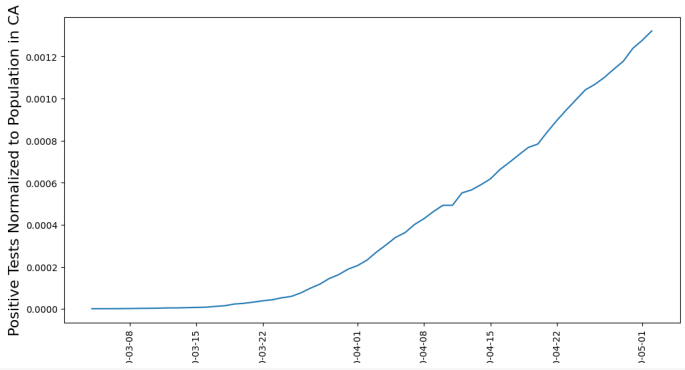

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



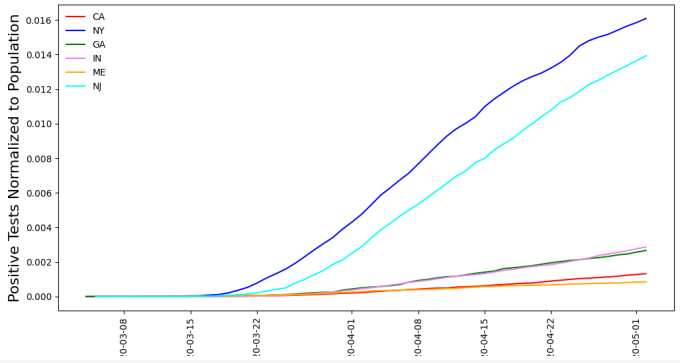

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



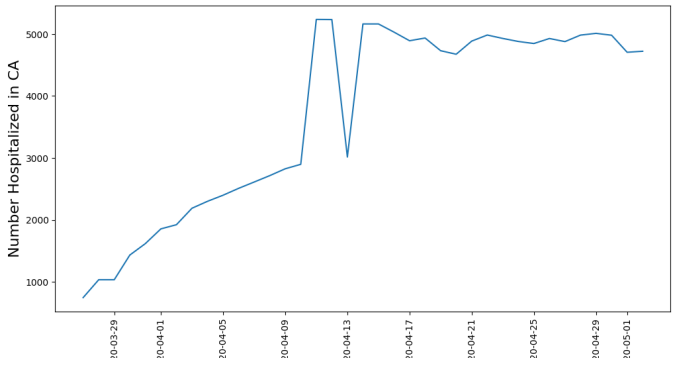

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



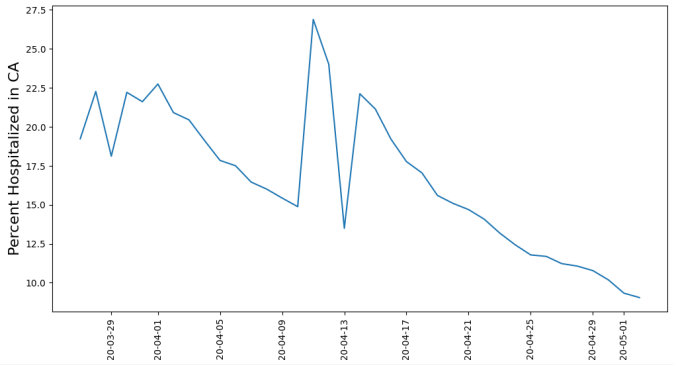

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



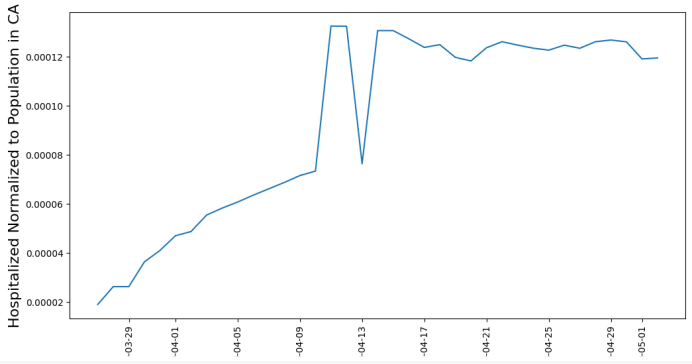

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



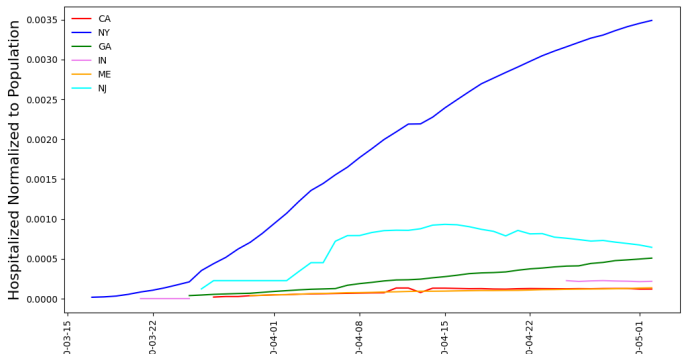

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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()
```

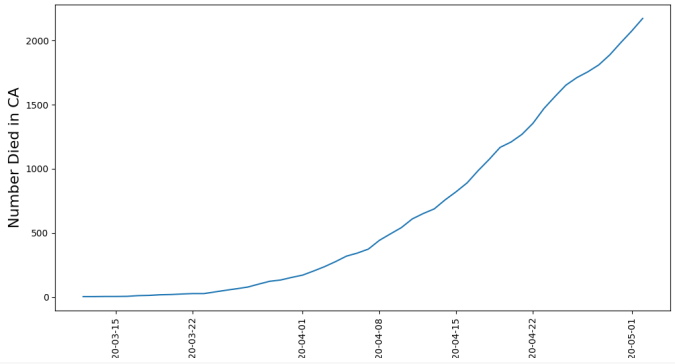
```
<Figure size 640x480 with 0 Axes>
```



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

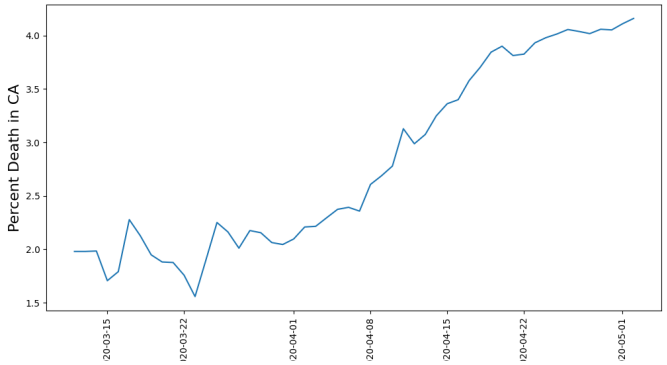
```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



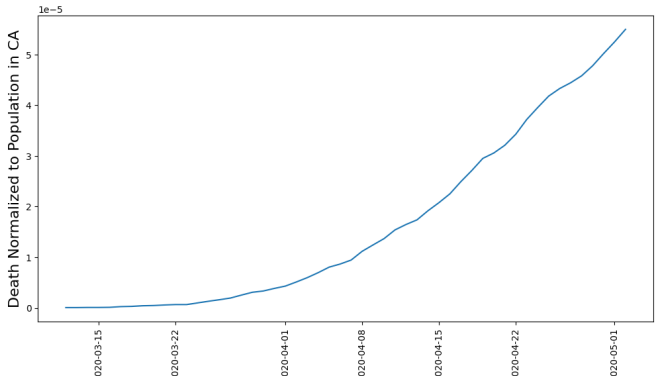

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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>
```



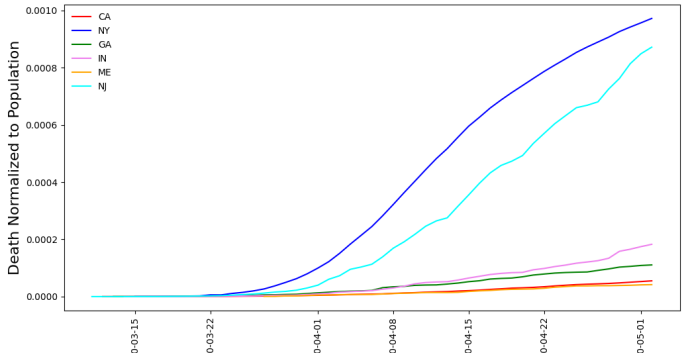

```
fig = plt.figure()
fig, ax = plt.subplots(figsize=(12, 6))
plt.rcParams.update(plt.rcParamsDefault)

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 (CexpDx^-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 | CexpDx^-1.csv
• CexpDx^-1.csv(application/vnd.ms-excel) - 1680 bytes, last modified: 4/16/2020 - 100% done
Saving CexpDx^-1.csv to CexpDx^-1.csv
```

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

	State	c (10^-4)	d	fit rank
0	AK	1.331139	-95.882596	2.0
1	AL	8.124937	-145.096536	1.0
2	AR	1.444874	-108.708991	3.0
3	AS	NaN	NaN	NaN
4	AZ	4.374538	-129.204382	1.0

```
df_state_params.describe()
```

	c (10^-4)	d	fit rank
count	51.000000	51.000000	51.000000
mean	28.922502	-142.879078	2.098039
std	53.235594	33.811201	2.156431
min	0.516899	-215.115296	1.000000
25%	3.745253	-165.040649	1.000000
50%	7.421743	-145.096536	1.000000
75%	20.958221	-123.240757	2.500000
max	231.216701	-47.945262	15.000000

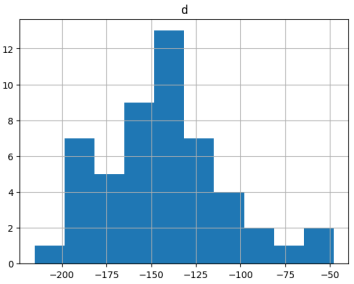
```
df_state_params.hist(column='c (10^-4)', bins=20)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f87007ee780>]],
```

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='d', bins=10)
```

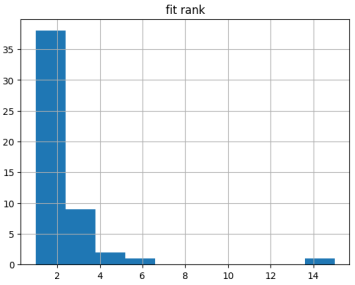
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f87007901d0>]],
       dtype=object)
```



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 0x7f8700762d68>]],
       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, 55]])
X = df_temp1.drop('State', axis = 1)
```

```
df_temp2 = df_state_params.drop(df_state_data.index[[3, 12, 27, 42, 50, 55]])
y = df_temp2['d']
```

```
# 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	0.981244	0.998612	0.998085	0.989852	0.999917	0.999896	0.895536	0.524429	0.894417	
Sum of NUM_BEN_Age_Less_65	0.981244	1.000000	0.977935	0.969186	0.960258	0.982419	0.979571	0.925224	0.473716	0.829126	
Sum of NUM_BEN_Age_65_to_74	0.998612	0.977935	1.000000	0.996336	0.982527	0.998360	0.998622	0.894585	0.516336	0.903356	
Sum of NUM_BEN_Age_75_to_84	0.998085	0.969186	0.996336	1.000000	0.992524	0.997902	0.998281	0.882970	0.528889	0.900554	
Sum of NUM_BEN_Age_Greater_84	0.989852	0.960258	0.982527	0.992524	1.000000	0.989495	0.990300	0.863288	0.560359	0.880694	
Sum of NUM_Female_BEN	0.999917	0.982419	0.998360	0.997902	0.989495	1.000000	0.999655	0.898089	0.522516	0.891099	
Sum of NUM_Male_BEN	0.999896	0.979571	0.998622	0.998281	0.990300	0.999655	1.000000	0.892308	0.525905	0.896619	
Sum of NUM_Black_or_African_American_BEN	0.895536	0.925224	0.894585	0.882970	0.863288	0.898089	0.892308	1.000000	0.300440	0.726598	
Sum of NUM_Asian_Pacific_Islander_BEN	0.524429	0.473716	0.516336	0.528889	0.560359	0.522516	0.525905	0.300440	1.000000	0.633176	
Sum of NUM_Hispanic_BEN	0.894417	0.829126	0.903356	0.900554	0.880694	0.891099	0.896619	0.726598	0.633176	1.000000	
Sum of NUM_American_IndianAlaska_Native_BEN	0.077349	0.053905	0.086472	0.081806	0.060473	0.076938	0.078057	-0.041752	0.116101	0.128504	
Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified	0.821569	0.771437	0.801707	0.830466	0.877807	0.819981	0.822968	0.634906	0.740095	0.734077	
Sum of NUM_Non-Hispanic_White_BEN	0.996809	0.978655	0.994347	0.996101	0.988772	0.997015	0.996718	0.887564	0.484280	0.866806	
Sum of NUM_Minorities	0.958404	0.925675	0.961032	0.957614	0.944932	0.957333	0.958539	0.867684	0.645007	0.959572	
Sum of Average_Age_of_BEN	0.678752	0.726826	0.682844	0.659778	0.633311	0.681583	0.674831	0.688783	0.128621	0.515698	
Sum of NUM_BEN_Atrial_Fibrillation	0.990319	0.969220	0.985453	0.991337	0.990274	0.990298	0.990716	0.888630	0.458859	0.852899	
Sum of NUM_BEN_Asthma	0.995489	0.979353	0.991510	0.992852	0.991685	0.995193	0.995563	0.892110	0.525949	0.882032	
Sum of NUM_BEN_Cancer	0.994721	0.971958	0.992833	0.994822	0.987275	0.994401	0.994876	0.899492	0.462638	0.884760	
Sum of NUM_BEN_Heart_Failure	0.997108	0.985088	0.995323	0.993852	0.984794	0.997149	0.996815	0.912205	0.483209	0.885584	
Sum of NUM_BEN_Chronic_Kidney_Disease	0.997480	0.980181	0.997065	0.995383	0.984109	0.997259	0.997594	0.906086	0.484030	0.894095	
Sum of NUM_BEN_Chronic_Obstructive_Pulmonary_Disease	0.986081	0.980417	0.981434	0.983841	0.977815	0.986841	0.985732	0.905511	0.428114	0.834249	
Sum of NUM_BEN_Hyperlipidemia	0.996199	0.974138	0.994686	0.996386	0.987456	0.996064	0.996454	0.902110	0.475920	0.885160	
Sum of NUM_BEN_Diabetes	0.997736	0.981117	0.996508	0.995642	0.985749	0.997730	0.997434	0.911839	0.493440	0.893757	
Sum of NUM_BEN_Hypertension	0.998843	0.982162	0.998059	0.996914	0.985866	0.998953	0.998618	0.907127	0.491385	0.887840	
Sum of NUM_BEN_Ischemic_Heart_Disease	0.993954	0.974989	0.991463	0.994045	0.985698	0.994069	0.993920	0.905308	0.456240	0.877711	
Sum of NUM_BEN_Stroke	0.990470	0.971925	0.988713	0.989929	0.980696	0.990390	0.990562	0.918281	0.446318	0.880341	
Sum of PCT_MEDICARE	0.710503	0.759188	0.713882	0.692945	0.667920	0.714560	0.706037	0.750005	0.138118	0.483164	
% Urban Pop	0.239324	0.172542	0.233998	0.252295	0.279217	0.235109	0.243800	0.173856	0.309518	0.280539	
Density (P/mi2)	-0.099963	-0.110703	-0.100658	-0.096325	-0.092020	-0.100642	-0.099698	-0.022034	-0.030915	-0.043443	
Children 0-18	0.884945	0.844648	0.874846	0.887257	0.911738	0.883447	0.886079	0.720117	0.776658	0.840977	
Adults 19-25	0.864191	0.823977	0.851022	0.867408	0.899146	0.862807	0.865269	0.694892	0.785158	0.809783	
Adults 26-34	0.846985	0.802138	0.833617	0.851409	0.884954	0.845326	0.848432	0.664003	0.812162	0.808332	
Adults 35-54	0.860076	0.817671	0.846322	0.864281	0.897686	0.859614	0.861368	0.692402	0.776687	0.803974	
Adults 55-64	0.838622	0.799478	0.819933	0.843902	0.887867	0.837386	0.840024	0.674409	0.735657	0.748042	
65+	0.840633	0.793344	0.820862	0.850354	0.895448	0.839427	0.842748	0.668530	0.692069	0.734700	
Latitude	-0.395637	-0.392189	-0.398492	-0.402613	-0.376290	-0.399287	-0.394167	-0.444864	-0.181758	-0.282176	
Longitude	0.036162	0.081918	0.023777	0.029848	0.047659	0.039383	0.032672	0.180157	-0.278308	-0.102843	
Land Area	0.235431	0.200886	0.248419	0.236252	0.212046	0.232714	0.237349	0.134233	0.203781	0.344879	
Water Area	0.038411	0.051521	0.032297	0.034407	0.046226	0.038427	0.038074	0.075830	0.047097	0.042598	
Mean Elevation	-0.133770	-0.196098	-0.117766	-0.126100	-0.141332	-0.139240	-0.128028	-0.298543	0.122569	0.060713	
Highest Elevation	-0.038246	-0.115800	-0.018904	-0.028611	-0.050574	-0.043899	-0.033001	-0.216534	0.306550	0.170534	
Lowest elevation	-0.344113	-0.337087	-0.333651	-0.346722	-0.365999	-0.345830	-0.342548	-0.297556	-0.559828	-0.292318	
Number of bordering states	0.092703	0.153356	0.090523	0.073651	0.071016	0.094695	0.089368	0.058746	-0.143034	-0.069825	
On Coast	0.464164	0.497887	0.435913	0.455132	0.512184	0.464668	0.463205	0.505677	0.168436	0.270946	
Borders Another Country	0.351913	0.303223	0.357825	0.350755	0.353612	0.345594	0.357028	0.180434	0.421510	0.499260	
Capital Latitude	-0.386561	-0.391908	-0.392011	-0.390199	-0.357046	-0.390466	-0.384523	-0.462045	-0.135382	-0.268392	
Capital Longitude	0.018177	0.067248	0.005968	0.010624	0.028374	0.021534	0.014318	0.173452	-0.302807	-0.121403	
Captial Land Area	0.003972	-0.007988	0.013931	0.004629	-0.017430	0.003967	0.003985	-0.025204	-0.015016	0.020859	
Capital Water Area	-0.091118	-0.100314	-0.086948	-0.090518	-0.095998	-0.091883	-0.090783	-0.100670	-0.021996	-0.041502	
Capital Mean Elevation	-0.166033	-0.186941	-0.154788	-0.163860	-0.181086	-0.169042	-0.162464	-0.226755	-0.114759	-0.033818	
Capital is the Largest City	-0.154074	-0.128106	-0.149158	-0.156946	-0.178305	-0.151938	-0.155487	-0.115770	-0.123610	-0.183826	
Largest City Latitude	-0.419120	-0.419459	-0.423088	-0.422919	-0.395974	-0.422660	-0.417371	-0.465860	-0.233447	-0.316075	
Largest City Longitude	0.048321	0.092830	0.035728	0.041774	0.061209	0.051430	0.044859	0.194353	-0.267233	-0.086233	
Number of Counties	0.659574	0.706073	0.666432	0.641478	0.607276	0.662444	0.655389	0.681011	0.096573	0.501717	
Became a State	-0.126570	-0.186422	-0.115157	-0.112935	-0.129547	-0.130157	-0.122258	-0.297191	0.083847	0.043321	
DaysSinceStayatHomeOrder	-0.021086	-0.020186	-0.030817	-0.027800	0.007419	-0.024088	-0.019335	-0.046409	0.222069	0.052387	
DaysSinceFirstPositive	0.357249	0.306142	0.355519	0.364255	0.380604	0.354390	0.360064	0.274180	0.255767	0.299965	
DaysSinceTestStart	0.273593	0.219953	0.272942	0.282120	0.296656	0.271182	0.275880	0.213147	0.187346	0.237754	
15-49yearsAllcauses	0.886884	0.854562	0.873498	0.888773	0.918524	0.885981	0.887769	0.736622	0.737674	0.796260	
15-49yearsAsthma	0.822646	0.785134	0.805485	0.825296	0.867899	0.821129	0.823386	0.663701	0.757688	0.750738	
15-49yearsChronickidneydisease	0.917925	0.892317	0.908566	0.917956	0.934714	0.917697	0.918009	0.803582	0.715805	0.829422	
15-49yearsChronicobstructivepulmonarydisease	0.895564	0.876357	0.879172	0.896199	0.927893	0.895413	0.895782	0.768727	0.635855	0.750723	
15-49yearsDiabetesmellitus	0.911319	0.879991	0.899800	0.913356	0.936198	0.910686	0.911822	0.779288	0.693258	0.813431	
15-49yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.879916	0.862208	0.865322	0.878905	0.908126	0.879919	0.879735	0.780069	0.644763	0.739273	
15-49yearsIschemicheartdisease	0.927678	0.926759	0.915842	0.922736	0.939065	0.928593	0.926497	0.847540	0.595987	0.766226	
15-49yearsNeoplasms	0.886136	0.858150	0.871628	0.887471	0.918495	0.885565	0.886670	0.745343	0.730821	0.786369	
15-49yearsOtherchronicrespiratorydiseases	0.905560	0.883613	0.891223	0.905653	0.934091	0.905184	0.905799	0.782455	0.653038	0.776105	
15-49yearsRheumaticheartdisease	0.902424	0.891711	0.892262	0.897798	0.916013	0.902292	0.902447	0.789490	0.691629	0.786644	
15-49yearsStroke	0.918867	0.897147	0.909310	0.918599	0.934170	0.918952	0.918838	0.805987	0.703053	0.816330	
50-69yearsAllcauses	0.878744	0.853509	0.861522	0.880659	0.917334	0.878249	0.879628	0.741745	0.678456	0.746293	
50-69yearsAsthma	0.799440	0.762340	0.778773	0.803715	0.854143	0.798097	0.800517	0.636677	0.742056	0.706594	
50-69yearsChronickidneydisease	0.916387	0.896945	0.904561	0.915572	0.937636	0.916311	0.916688	0.807095	0.676156	0.795122	
50-69yearsChronicobstructivepulmonarydisease	0.877906	0.870963	0.859255	0.877419	0.911277	0.878288	0.878455	0.762080	0.542320	0.678790	
50-69yearsDiabetesmellitus	0.881134	0.855438	0.863901	0.883450	0.919693	0.880643	0.882016	0.750770	0.653836	0.744109	
50-69yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.861583	0.838312	0.844421	0.862487	0.900025	0.861105	0.862351	0.735721	0.674419	0.726896	
50-69yearsIschemicheartdisease	0.904978	0.899635	0.888882	0.901757	0.930901	0.905480	0.904633	0.804866	0.618552	0.737135	
50-69yearsNeoplasms	0.871034	0.851227	0.852407	0.872097	0.911177	0.870768	0.871794	0.742697	0.651344	0.720355	
50-69yearsOtherchronicrespiratorydiseases	0.883753	0.873315	0.866185	0.882303	0.916676	0.883761	0.884109	0.777456	0.570326	0.702496	
50-69yearsRheumaticheartdisease	0.891423	0.888783	0.879360	0.885632	0.907455	0.891577	0.891644	0.791210	0.641520	0.739209	
50-69yearsStroke	0.906978	0.890724	0.893997	0.906473	0.929829	0.907205	0.907337	0.798197	0.657305	0.765372	
70+yearsAllcauses	0.847442	0.816751	0.826481	0.852488	0.898736	0.846808	0.848827	0.697071	0.654330	0.704089	
70+yearsAsthma	0.789028	0.744699	0.766961	0.797072	0.850124	0.787734	0.790683	0.607592	0.747748	0.697495	
70+yearsChronickidneydisease	0.875670	0.856224	0.857657	0.876360	0.914685	0.875402	0.876558	0.748543	0.643668	0.724464	
70+yearsChronicobstructivepulmonarydisease	0.865156	0.840259	0.845077	0.869812	0.910486	0.864851	0.866648	0.717738	0.599882	0.699384	
70+yearsDiabetesmellitus	0.843401	0.812744	0.821754	0.849108	0.896105	0.842866	0.844798	0.698870	0.638607	0.695980	
70+yearsInterstitiallungdiseaseandpulmonarysarcoidosis	0.831802	0.797053	0.811884	0.837251	0.883211	0.830937	0.833566	0.671418	0.689938	0.706600	
70+yearsIschemicheartdisease	0.839315	0.817188	0.816155	0.842376	0.892450	0.839017	0.840006	0.702589	0.634217	0.681467	
70+yearsNeoplasms	0.835509	0.805555	0.813851	0.840697	0.888696	0.834904	0.836996	0.685194	0.648382	0.686885	
70+yearsOtherchronicrespiratorydiseases	0.874566	0.857451	0.856689	0.874418	0.912408	0.874057	0.875635	0.748794	0.586877	0.705416	
70+yearsRheumaticheartdisease	0.842665	0.837198	0.824793	0.837776	0.877986	0.842106	0.843753	0.716967	0.626865	0.680450	
70+yearsStroke	0.870071	0.847350	0.852618	0.871917	0.909247	0.869793	0.871222	0.729313	0.660317	0.721808	

AllAgesAllcauses	0.878588	0.849145	0.861845	0.881318	0.917774	0.877905	0.879599	0.733191	0.695871	0.758571
AllAgesAsthma	0.831304	0.792231	0.813720	0.835086	0.877936	0.829823	0.832271	0.670182	0.749293	0.755115
AllAgesChronickidneydisease	0.904402	0.883840	0.890334	0.904351	0.932451	0.904221	0.904913	0.786269	0.672518	0.775388
AllAgesChronicobstructivepulmonarydisease	0.875803	0.858774	0.856544	0.878011	0.915264	0.875795	0.876810	0.742655	0.580530	0.697110
AllAgesDiabetesmellitus	0.878317	0.849967	0.860647	0.881652	0.919029	0.877785	0.879297	0.742684	0.658751	0.744450
AllAgesInterstitiallungdiseaseandpulmonarysarcoidosis	0.852165	0.823512	0.833849	0.855184	0.896166	0.851536	0.853348	0.710857	0.681959	0.721415
AllAgesIschemicheartdisease	0.882192	0.869062	0.862943	0.881839	0.920162	0.882348	0.882268	0.765407	0.628781	0.718536
AllAgesNeoplasms	0.863741	0.839097	0.844574	0.866307	0.907502	0.863308	0.864731	0.725375	0.663557	0.720556
AllAgesOtherchronicrespiratorydiseases	0.902524	0.884302	0.887253	0.902007	0.932378	0.902199	0.902973	0.782928	0.623094	0.754060
AllAgesRheumaticheartdisease	0.879079	0.873449	0.864765	0.873886	0.903225	0.878847	0.879647	0.764366	0.648372	0.728889
AllAgesStroke	0.894221	0.873914	0.879380	0.894925	0.924402	0.894172	0.894932	0.768243	0.667366	0.753694
AllAgesTotal	0.879105	0.851798	0.861916	0.881553	0.918458	0.878539	0.880046	0.737330	0.684312	0.751284
Airpollution	0.887961	0.886816	0.873716	0.881728	0.909727	0.888331	0.887745	0.777358	0.655261	0.730312
Highbody-massindex	0.892574	0.870891	0.875767	0.893521	0.928006	0.892269	0.893140	0.768343	0.661265	0.754688
Highfastingplasmaglucose	0.885519	0.868208	0.867475	0.886276	0.922271	0.885469	0.886040	0.770273	0.617289	0.725192
HighLDLcholesterol	0.892016	0.880761	0.874040	0.890927	0.925614	0.892300	0.891950	0.780580	0.627540	0.728204
Highsystolicbloodpressure	0.896298	0.880918	0.879042	0.896085	0.929809	0.896363	0.896543	0.785586	0.638860	0.742184
Impairedkidneyfunction	0.888684	0.870825	0.871779	0.888904	0.923102	0.888693	0.889102	0.770201	0.658349	0.741125
Noaccesstohandwashingfacility	0.876183	0.855685	0.860915	0.875209	0.908573	0.875653	0.876553	0.753540	0.668494	0.745193
Smoking	0.880256	0.864750	0.861340	0.881441	0.918294	0.880406	0.880814	0.757828	0.604253	0.706722
Log10Pop	0.730625	0.738162	0.716041	0.724834	0.750347	0.731527	0.730121	0.665876	0.425752	0.516893
DaysSinceInfection	0.412821	0.360632	0.410278	0.422147	0.434949	0.410914	0.414835	0.354265	0.259813	0.349814
Children0-18	0.170467	0.184747	0.184614	0.162743	0.122101	0.172944	0.167407	0.173000	0.050826	0.150062
Allriskfactors	0.881460	0.858902	0.864027	0.883001	0.919342	0.881097	0.882336	0.747372	0.661101	0.738270
State Area Ratio	-0.128550	-0.166800	-0.113602	-0.122087	-0.145833	-0.130919	-0.125230	-0.249270	-0.098980	-0.021322
Elevation Ratio	0.006435	-0.008386	0.016149	0.010278	-0.014214	0.007719	0.004657	0.117476	0.047721	0.006349
Capital Area Ratio	-0.107958	-0.139494	-0.098783	-0.101355	-0.113882	-0.109206	-0.106482	-0.160284	-0.071880	-0.046967
Boundaries	0.500872	0.558822	0.480645	0.479234	0.518677	0.501040	0.498998	0.456575	0.125069	0.276862
Latitude Difference to State Capital	-0.251296	-0.188306	-0.234552	-0.277897	-0.313035	-0.251254	-0.254554	-0.091516	-0.409883	-0.230231
Longitude Difference to State Capital	-0.132644	-0.120676	-0.128482	-0.139685	-0.144345	-0.132468	-0.134362	-0.089151	-0.102664	-0.103498
Latitude Difference to DC	-0.395637	-0.392189	-0.398492	-0.402613	-0.376290	-0.399287	-0.394167	-0.444864	-0.181758	-0.282176
Longitude Difference to DC	-0.036162	-0.081918	-0.023777	-0.029848	-0.047659	-0.039383	-0.032672	-0.180157	0.278308	0.102843
Latitude Difference to Center	-0.395637	-0.392189	-0.398492	-0.402613	-0.376290	-0.399287	-0.394167	-0.444864	-0.181758	-0.282176
Longitude Difference to Center	-0.036162	-0.081918	-0.023777	-0.029848	-0.047659	-0.039383	-0.032672	-0.180157	0.278308	0.102843

```
#  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: 50 entries, 0 to 54
Data columns (total 38 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Sum of NUM_Medicare_BEN                     50 non-null     float64
 1   Sum of NUM_Black_or_African_American_BEN    50 non-null     float64
 2   Sum of NUM_Asian_Pacific_Islander_BEN       50 non-null     float64
 3   Sum of NUM_Hispanic_BEN                     50 non-null     float64
 4   Sum of NUM_American_IndianAlaska_Native_BEN 50 non-null     float64
 5   Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified  50 non-null  float64
 6   Sum of Average_Age_of_BEN                   50 non-null     float64
 7   Sum of PCT_MEDICARE                         50 non-null     float64
 8   % Urban Pop                                 50 non-null     float64
 9   Density (P/mi2)                             50 non-null     float64
 10  Children 0-18                               50 non-null     float64
 11  Latitude                                    50 non-null     float64
 12  Longitude                                   50 non-null     float64
 13  Land Area                                   50 non-null     float64
 14  Water Area                                  50 non-null     float64
 15  Mean Elevation                              50 non-null     float64
 16  Highest Elevation                           50 non-null     float64
 17  Lowest elevation                            50 non-null     float64
 18  Number of bordering states                  50 non-null     float64
 19  On Coast                                    50 non-null     float64
 20  Borders Another Country                     50 non-null     float64
 21  Captial Land Area                           50 non-null     float64
 22  Capital Water Area                          50 non-null     float64
 23  Capital Mean Elevation                      50 non-null     float64
 24  Capital is the Largest City                 50 non-null     float64
 25  Became a State                              50 non-null     float64
 26  DaysSinceStayatHomeOrder                    50 non-null     float64
 27  DaysSinceFirstPositive                      50 non-null     float64
 28  DaysSinceTestStart                          50 non-null     float64
 29  Log10Pop                                    50 non-null     float64
 30  DaysSinceInfection                          50 non-null     float64
 31  Children0-18                                50 non-null     float64
 32  State Area Ratio                            50 non-null     float64
 33  Elevation Ratio                             50 non-null     float64
 34  Capital Area Ratio                          50 non-null     float64
 35  Boundaries                                  50 non-null     float64
 36  Latitude Difference to State Capital        50 non-null     float64
 37  Longitude Difference to State Capital       50 non-null     float64
dtypes: float64(38)
memory usage: 15.2 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	5.000000e+01	5.000000e+01	5.000000e+01	5.000000e+01	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	5.000000e+01	50.000000	50.000
mean	1.057661e+07	9.653450e+05	1.439833e+05	5.412557e+05	39027.080000	88021.940000	3574.438430	52.910339	74.294000	440.074918	1.513864e+06	39.399000	-93.060
std	1.317051e+07	1.280319e+06	4.765951e+05	1.644850e+06	88090.522177	115323.411288	2527.939135	46.730862	14.976518	1662.815407	1.772177e+06	6.112752	19.379
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%	2.518838e+06	6.328700e+04	6.770500e+03	3.269350e+04	2929.750000	19717.000000	1529.891773	13.039638	66.025000	56.532925	4.596994e+05	35.659511	-99.697
50%	6.848160e+06	3.978665e+05	2.777200e+04	1.050865e+05	7558.000000	58749.500000	3752.318846	52.406965	74.400000	110.435050	1.059542e+06	39.583974	-89.301
75%	1.479523e+07	1.548688e+06	7.370350e+04	2.012818e+05	28748.250000	100639.500000	5216.766839	79.721735	87.725000	226.003700	1.728603e+06	43.052506	-78.579
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
```

```
((37, 38), (37,), (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	3.700000e+01	3.700000e+01	37.000000	3.700000e+01	37.000000	37.000000	37.000000	37.000000	37.000000	37.000000	3.700000e+01	37.000000	37.000
mean	1.157925e+07	1.130874e+06	98436.675676	5.365955e+05	41090.810811	90806.621622	3662.637363	54.022267	74.981081	514.205595	1.534385e+06	38.957733	-92.705
std	1.384476e+07	1.398898e+06	171362.519286	1.696478e+06	97224.502922	103467.502844	2692.872668	45.920797	15.241297	1924.750858	1.503167e+06	6.614804	19.865
min	1.655870e+05	2.960000e+02	166.000000	4.130000e+02	0.000000	1693.000000	70.002893	1.702566	38.700000	1.286300	1.160123e+05	21.094318	-157.498
25%	3.242760e+06	1.057920e+05	12709.000000	4.230000e+04	3527.000000	20656.000000	1517.642713	12.694204	66.200000	57.654700	4.774097e+05	35.565342	-98.268
50%	8.517210e+06	5.217080e+05	30624.000000	1.112130e+05	7061.000000	63452.000000	3926.277651	54.780620	74.600000	112.820400	1.140729e+06	39.063946	-86.791
75%	1.629170e+07	1.693845e+06	76800.000000	2.027260e+05	28990.000000	101294.000000	5219.433901	81.425494	87.900000	228.024300	2.042904e+06	41.597782	-79.806
max	7.644909e+07	7.011107e+06	793067.000000	1.007620e+07	560433.000000	441947.000000	13644.965980	219.756971	100.000000	11814.541000	7.650584e+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 = [2, 3, 4]
n_estimators = [28, 29, 30]
min_samples_split = [1.5, 2, 2.5]
min_samples_leaf = [3.5, 4, 4.5]
max_leaf_nodes = [None]
max_features = ['auto']
ccp_alpha = [0.0, 0.00625, 0.0125]
min_weight_fraction_leaf = [0.0, 0.00625, 0.0125]

hyperF = dict(n_estimators = n_estimators, max_depth = max_depth,
              min_samples_split = min_samples_split,
              min_samples_leaf = min_samples_leaf,
              max_leaf_nodes = max_leaf_nodes,
              max_features = max_features,
              ccp_alpha=ccp_alpha,
              min_weight_fraction_leaf=min_weight_fraction_leaf)

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

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

```
Fitting 3 folds for each of 729 candidates, totalling 2187 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    1.2s
[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:    1.6s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1862s.) Setting batch_size=2.
[Parallel(n_jobs=-1)]: Batch computation too fast (0.0837s.) Setting batch_size=4.
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1513s.) Setting batch_size=8.
[Parallel(n_jobs=-1)]: Done  58 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 130 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 202 tasks      | elapsed:    4.7s
[Parallel(n_jobs=-1)]: Done 290 tasks      | elapsed:    6.0s
[Parallel(n_jobs=-1)]: Done 378 tasks      | elapsed:    7.7s
[Parallel(n_jobs=-1)]: Done 482 tasks      | elapsed:    9.1s
[Parallel(n_jobs=-1)]: Done 586 tasks      | elapsed:   10.5s
[Parallel(n_jobs=-1)]: Done 706 tasks      | elapsed:   12.8s
[Parallel(n_jobs=-1)]: Done 826 tasks      | elapsed:   14.4s
[Parallel(n_jobs=-1)]: Done 962 tasks      | elapsed:   16.8s
[Parallel(n_jobs=-1)]: Done 1098 tasks      | elapsed:   19.2s
[Parallel(n_jobs=-1)]: Done 1250 tasks      | elapsed:   21.4s
[Parallel(n_jobs=-1)]: Done 1402 tasks      | elapsed:   24.1s
[Parallel(n_jobs=-1)]: Done 1570 tasks      | elapsed:   26.5s
[Parallel(n_jobs=-1)]: Done 1738 tasks      | elapsed:   29.2s
[Parallel(n_jobs=-1)]: Done 1922 tasks      | elapsed:   32.4s
[Parallel(n_jobs=-1)]: Done 2106 tasks      | elapsed:   35.5s
The score achieved with the best parameters =  -0.1309525586842498

The parameters are: {'ccp_alpha': 0.0, 'max_depth': 3, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 29}
[Parallel(n_jobs=-1)]: Done 2187 out of 2187 | elapsed:   36.6s 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 2.9MB/s
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: patsy>=0.4.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.5.1)
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (1.0.3)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.6/dist-packages (from category_encoders==2.0.0) (0.22.2.post1)
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: 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)
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)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.0.0
```

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

pipeline1 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy='mean'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                          max_depth=3, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=4,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=29, 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))
```

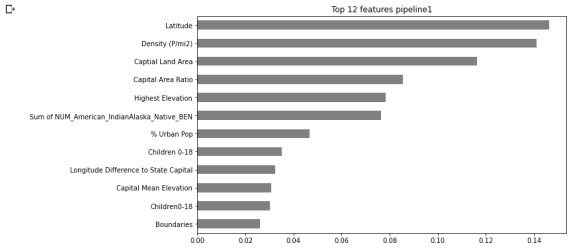
```
Training Accuracy:  R^2 =  0.6195488363328325
Validation Accuracy:  R^2 =  0.48283160859213214
```

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

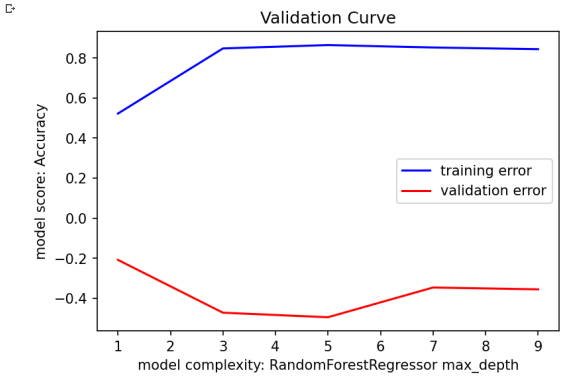
```
Feature Importances =
[0.         0.00426752 0.         0.         0.07631863 0.00543221
 0.00740174 0.         0.0467188  0.14111398 0.03505111 0.14622923
 0.00726956 0.00351066 0.         0.01670479 0.07845557 0.
 0.01871196 0.00280924 0.         0.11643129 0.00982198 0.03083072
 0.         0.01720428 0.0157221  0.0044032  0.01288937 0.
 0.         0.03006819 0.00778861 0.00929704 0.08558693 0.02593157
 0.01157612 0.03245364]
```

```
# 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: RandomForestRegressor max_depth')
plt.ylabel('model score: Accuracy')
plt.legend();
```




```
# Get drop-column importances
column = 'Density (P/mi2)'

pipeline3 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=29, 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 Density (P/mi2): 0.5009375902676718
Validation Accuracy with Density (P/mi2): 0.4828316085921321
Drop-Column Importance for Density (P/mi2): -0.018105981675396 73
```

```
# 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=0.15, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=29, 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=0.15, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=4,
                      min_samples_split=2, min_weight_fraction_leaf=0,
                      n_estimators=29, 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 2.8MB/s
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (1.18.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from eli5) (1.4.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.22.2.post1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (from eli5) (0.10.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from eli5) (1.12.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from eli5) (2.11.2)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.8.7)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (19.3.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.18->eli5) (0.14.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2->eli5) (1.1.1)
Installing collected packages: eli5
Successfully installed eli5-0.10.1
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  deprecated 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
0.1736 ± 0.1030	Capital Area Ratio
0.1545 ± 0.2241	Density (P/mi2)
0.0665 ± 0.0669	Captial Land Area
0.0546 ± 0.0414	Latitude
0.0284 ± 0.0096	% Urban Pop
0.0134 ± 0.0002	Longitude Difference to State Capital
0.0128 ± 0.0150	Became a State
0.0094 ± 0.0061	DaysSinceStayatHomeOrder
0.0072 ± 0.0004	Latitude Difference to State Capital
0.0068 ± 0.0193	Sum of NUM_American_IndianAlaska_Native_BEN
0.0060 ± 0.0121	Number of bordering states
0.0056 ± 0.0070	Boundaries
0.0046 ± 0.0216	Capital Mean Elevation
0.0034 ± 0.0172	Longitude
0.0027 ± 0.0078	Highest Elevation
0.0022 ± 0.0043	Land Area
0.0021 ± 0.0068	Sum of Average_Age_of_BEN
0.0020 ± 0.0000	Sum of NUM_Black_or_African_American_BEN
0.0013 ± 0.0088	DaysSinceTestStart
0.0013 ± 0.0031	State Area Ratio
0.0010 ± 0.0050	On Coast
0.0009 ± 0.0055	Capital Water Area
0 ± 0.0000	Sum of NUM_Asian_Pacific_Islander_BEN
0 ± 0.0000	Sum of NUM_Hispanic_BEN
0 ± 0.0000	Sum of PCT_MEDICARE
0 ± 0.0000	Sum of NUM_Medicare_BEN
0 ± 0.0000	Water Area
0 ± 0.0000	Lowest elevation
0 ± 0.0000	DaysSinceInfection
0 ± 0.0000	Borders Another Country
0 ± 0.0000	Log10Pop
0 ± 0.0000	Capital is the Largest City
-0.0004 ± 0.0009	DaysSinceFirstPositive
-0.0035 ± 0.0032	Elevation Ratio
-0.0060 ± 0.0175	Sum of NUM_BEN_With_Race_Not_Elsewhere_Classified
-0.0186 ± 0.0044	Children 0-18
-0.0193 ± 0.0076	Mean Elevation
-0.0236 ± 0.0123	Children0-18

```
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.:  0.6195488363328325
Coefficient of determination r2 for the validation set.:  0.4828316085921321
Mean squared error: 528.60
```

```
# Thus, Density remains important according to feature permutation than according to feature importance in the Rando
# Use importances for feature selection
print('Shape before removing features:', X_train.shape)
```

```
Shape before removing features: (37, 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)
```

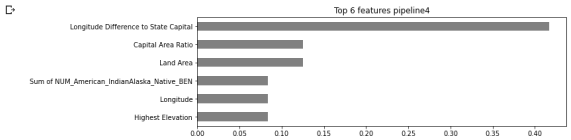
```
Shape after removing features: (37, 22)
```

```
# Random forest classifier with 22 features
X_val = X_val[features1]
pipeline4 = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'most_frequent'),
    RandomForestRegressor(bootstrap=True, ccp_alpha=0,
                          max_depth=3, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=4,
                          min_samples_split=2, min_weight_fraction_leaf=0,
                          n_estimators=29, 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);
```

```
# 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.:  0.626622750237471
Coefficient of determination r2 for the validation set.:  0.4897612195667084
Mean squared error: 521.52
```

```
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 = 6
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features pipeline4')
importances2.sort_values()[-n:].plot.barh(color='grey');
```




```
# Gradient boosting using XGboost with 15 estimators
from xgboost import XGBRegressor
pipeline5 = make_pipeline(
    ce.OrdinalEncoder(),
    XGBRegressor(n_estimators=15,
                 max_depth=1,
                 learning_rate=0.41, # try a higher learning rate
                 random_state=42,
                 n_jobs=-1)
)
pipeline5.fit(X_train, y_train);
```

```
[19:45:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
```

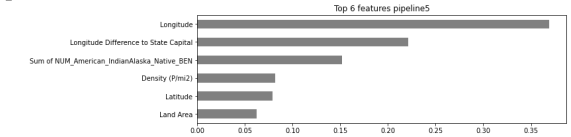
```
# Coefficient of determination r2 for the training set
pipeline_score = pipeline5.score(X_train,y_train)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)

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

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

```
Coefficient of determination r2 for the training set.:  0.7838756216629406
Coefficient of determination r2 for the validation set.:  0.605786244166149
Mean squared error: 402.93
```

The best validation score (0.605786) and lowest MSE (402.93) comes from using Gradient Boosting with 13 estimators.

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

# Get feature importances
encoder = pipeline5.named_steps['ordinalencoder']
encoded = encoder.transform(X_val)
rf = pipeline5.named_steps['xgbregressor']
importances3 = pd.Series(rf.feature_importances_, encoded.columns)

# Plot feature importances
n = 6
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features pipeline5')
importances3.sort_values()[-n:].plot.barh(color='grey');
```

```
[19:45:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
```



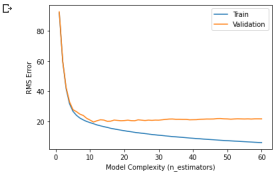

```
# Gradient boosting using XGboost with 1000 estimators
encoder = ce.OrdinalEncoder()
X_train_encoded = encoder.fit_transform(X_train)
X_val_encoded = encoder.transform(X_val)
X_train.shape, X_val.shape, X_train_encoded.shape, X_val_encoded.shape
```

```
((37, 22), (13, 22), (37, 22), (13, 22))
```

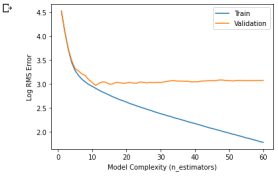
```
eval_set = [(X_train_encoded, y_train),
            (X_val_encoded, y_val)]

model2 = XGBRegressor(
          n_estimators=1000, # <= 1000 trees, depends on early stopping
          max_depth=1,
          learning_rate=0.41, # try higher learning rate
          n_jobs=-1)

model2.fit(X_train_encoded, y_train, eval_set=eval_set, eval_metric='rmse',
          early_stopping_rounds=50)
```

```
[19:45:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[0]    validation_0-rmse:91.9687    validation_1-rmse:92.6396
Multiple eval metrics have been passed: 'validation_1-rmse' will be used for early stopping.

Will train until validation_1-rmse hasn't improved in 50 rounds.
[1]    validation_0-rmse:59.8496    validation_1-rmse:59.9746
[2]    validation_0-rmse:41.644     validation_1-rmse:42.4277
[3]    validation_0-rmse:31.6015    validation_1-rmse:32.9433
[4]    validation_0-rmse:26.6825    validation_1-rmse:27.7771
[5]    validation_0-rmse:24.0516    validation_1-rmse:26.4663
[6]    validation_0-rmse:22.2411    validation_1-rmse:24.8219
[7]    validation_0-rmse:20.9869    validation_1-rmse:23.9266
[8]    validation_0-rmse:19.944     validation_1-rmse:22.0353
[9]    validation_0-rmse:19.1823    validation_1-rmse:20.8479
[10]   validation_0-rmse:18.435     validation_1-rmse:19.5501
[11]   validation_0-rmse:17.6743    validation_1-rmse:20.4497
[12]   validation_0-rmse:17.0584    validation_1-rmse:21.0673
[13]   validation_0-rmse:16.5325    validation_1-rmse:20.8832
[14]   validation_0-rmse:15.9991    validation_1-rmse:20.0731
[15]   validation_0-rmse:15.4335    validation_1-rmse:20.1635
[16]   validation_0-rmse:15.0028    validation_1-rmse:20.8857
[17]   validation_0-rmse:14.5576    validation_1-rmse:20.6639
[18]   validation_0-rmse:14.2005    validation_1-rmse:20.4452
[19]   validation_0-rmse:13.8148    validation_1-rmse:20.5736
[20]   validation_0-rmse:13.4191    validation_1-rmse:20.8303
[21]   validation_0-rmse:13.098     validation_1-rmse:20.5082
[22]   validation_0-rmse:12.765     validation_1-rmse:20.4495
[23]   validation_0-rmse:12.4537    validation_1-rmse:21.0325
[24]   validation_0-rmse:12.1802    validation_1-rmse:20.8216
[25]   validation_0-rmse:11.8973    validation_1-rmse:20.5977
[26]   validation_0-rmse:11.6352    validation_1-rmse:20.8435
[27]   validation_0-rmse:11.3495    validation_1-rmse:20.7061
[28]   validation_0-rmse:11.085     validation_1-rmse:20.8546
[29]   validation_0-rmse:10.8404    validation_1-rmse:20.8113
[30]   validation_0-rmse:10.6297    validation_1-rmse:21.0464
[31]   validation_0-rmse:10.3895    validation_1-rmse:21.3184
[32]   validation_0-rmse:10.1956    validation_1-rmse:21.5471
[33]   validation_0-rmse:9.95444    validation_1-rmse:21.5849
[34]   validation_0-rmse:9.76568    validation_1-rmse:21.4027
[35]   validation_0-rmse:9.58377    validation_1-rmse:21.4196
[36]   validation_0-rmse:9.37852    validation_1-rmse:21.3164
[37]   validation_0-rmse:9.17668    validation_1-rmse:21.3223
[38]   validation_0-rmse:8.99263    validation_1-rmse:21.0383
[39]   validation_0-rmse:8.84106    validation_1-rmse:21.0943
[40]   validation_0-rmse:8.64903    validation_1-rmse:21.1314
[41]   validation_0-rmse:8.47994    validation_1-rmse:21.4191
[42]   validation_0-rmse:8.29076    validation_1-rmse:21.471
[43]   validation_0-rmse:8.1178     validation_1-rmse:21.5766
[44]   validation_0-rmse:7.96713    validation_1-rmse:21.5596
[45]   validation_0-rmse:7.82337    validation_1-rmse:21.6025
[46]   validation_0-rmse:7.67152    validation_1-rmse:21.8777
[47]   validation_0-rmse:7.51868    validation_1-rmse:21.9137
[48]   validation_0-rmse:7.36816    validation_1-rmse:21.7134
[49]   validation_0-rmse:7.23374    validation_1-rmse:21.6397
[50]   validation_0-rmse:7.09412    validation_1-rmse:21.474
[51]   validation_0-rmse:6.95826    validation_1-rmse:21.6141
[52]   validation_0-rmse:6.83248    validation_1-rmse:21.7338
[53]   validation_0-rmse:6.68974    validation_1-rmse:21.6532
[54]   validation_0-rmse:6.55768    validation_1-rmse:21.6003
[55]   validation_0-rmse:6.44008    validation_1-rmse:21.682
[56]   validation_0-rmse:6.32668    validation_1-rmse:21.5601
[57]   validation_0-rmse:6.19023    validation_1-rmse:21.7015
[58]   validation_0-rmse:6.06951    validation_1-rmse:21.7155
[59]   validation_0-rmse:5.95723    validation_1-rmse:21.6837
[60]   validation_0-rmse:5.83285    validation_1-rmse:21.6631
Stopping. Best iteration:
[10]   validation_0-rmse:18.435     validation_1-rmse:19.5501

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.41, max_delta_step=0,
             max_depth=1, min_child_weight=1, missing=None, n_estimators=1000,
             n_jobs=-1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)
```

```
# Plot the results
results = model2.evals_result()
train_error = results['validation_0']['rmse']
val_error = results['validation_1']['rmse']
epoch = range(1, len(train_error)+1)
plt.plot(epoch, train_error, label='Train')
plt.plot(epoch, val_error, label='Validation')
plt.ylabel('RMS Error')
plt.xlabel('Model Complexity (n_estimators)')
# plt.ylim((0.18, 0.22)) # Zoom in
plt.legend();
```




```
# Plot log classification error versus model complexity
import numpy as np
results = model2.evals_result()
log_train_error = np.log(results['validation_0']['rmse'])
log_val_error = np.log(results['validation_1']['rmse'])
epoch = range(1, len(train_error)+1)
plt.plot(epoch, log_train_error, label='Train')
plt.plot(epoch, log_val_error, label='Validation')
plt.ylabel('Log RMS Error')
plt.xlabel('Model Complexity (n_estimators)')
# plt.ylim((0.18, 0.22)) # Zoom in
plt.legend();
```



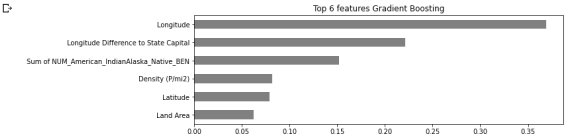

```
#Gradient Boosting R^2

gb = make_pipeline(
                ce.OrdinalEncoder(),
#                 XGBRegressor(n_estimators=45, objective='reg:squarederror', n_jobs=-1)
                XGBRegressor(n_estimators=15,
                             objective='reg:squarederror',
                             max_depth=1, # try deeper trees because of high cardinality categoricals
                             learning_rate=0.41, # try a higher learning rate
                             random_state=42,
                             n_jobs=-1)
                )
gb.fit(X_train, y_train)

#y_pred = gb.predict(X_val)
#print('Gradient Boosting R^2', r2_score(y_val, y_pred))
```

```
    Pipeline(memory=None,
             steps=[('ordinalencoder',
                     OrdinalEncoder(cols=[], drop_invariant=False,
                                    handle_missing='value', handle_unknown='value',
                                    mapping=[], return_df=True, verbose=0)),
                    ('xgbregressor',
                     XGBRegressor(base_score=0.5, booster='gbtree',
                                  colsample_bylevel=1, colsample_bynode=1,
                                  colsample_bytree=1, gamma=0,
                                  importance_type='gain', learning_rate=0.41,
                                  max_delta_step=0, max_depth=1, min_child_weight=1,
                                  missing=None, n_estimators=15, n_jobs=-1,
                                  nthread=None, objective='reg:squarederror',
                                  random_state=42, reg_alpha=0, reg_lambda=1,
                                  scale_pos_weight=1, seed=None, silent=None,
                                  subsample=1, verbosity=1))],
             verbose=False)
```

```
# Coefficient of determination r2 for the training set
#pipeline_score = gb.score(X_train,y_train)
y_train_pred = gb.predict(X_train)
pipeline_score = r2_score(y_train, y_train_pred)
print("Coefficient of determination r2 for the training set.: ", pipeline_score)

# Coefficient of determination r2 for the validation set
#pipeline_score = gb.score(X_val,y_val)
y_val_pred = gb.predict(X_val)
pipeline_score = r2_score(y_val, y_val_pred)
print("Coefficient of determination r2 for the validation set.: ", pipeline_score)

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

```
Coefficient of determination r2 for the training set.:  0.7838756216629406
Coefficient of determination r2 for the validation set.:  0.605786244166149
Mean squared error: 402.93
```

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

# Get feature importances
encoder = gb.named_steps['ordinalencoder']
encoded = encoder.transform(X_val)
rf = gb.named_steps['xgbregressor']
importances4 = pd.Series(rf.feature_importances_, encoded.columns)

# Plot feature importances
n = 6
plt.figure(figsize=(10,n/2))
#plt.title(f'Top {n} features pipeline5')
plt.title(f'Top {n} features Gradient Boosting')
importances4.sort_values()[-n:].plot.barh(color='grey');
```

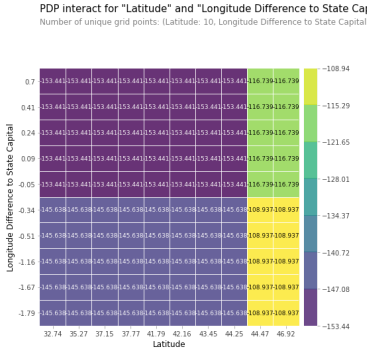



```
!pip install pdpbox
```

```
Collecting pdpbox
  Downloading https://files.pythonhosted.org/packages/87/23/ac7da5ba1c6c03a87c412e7e7b6e91a10d6ecf4474906c3e736f93940d49/PDPbox-0.2.0.tar.gz (57.7MB)
     |████████████████████████████████| 57.7MB 62kB/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: 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: 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: 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=9178ce1471a64bb075e7143d79a73f38354c202a78f9fb0fab52e284ac7ae097
  Stored in directory: /root/.cache/pip/wheels/7d/08/51/63fd122b04a2c87d780464eeffb94867c75bd96a64d500a3fe
Successfully built pdpbox
Installing collected packages: pdpbox
Successfully installed pdpbox-0.2.0
```

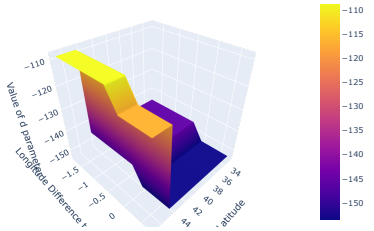
```
# Partial Dependence Plots with 2 features
from pdpbox.pdp import pdp_interact, pdp_interact_plot
features2 = ['Latitude', 'Longitude Difference to State Capital']
interaction = pdp_interact(
                           model=gb,
                           dataset=X_val,
                           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 d 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()
```



In order to establish feature importances, Shapley Force Plots are used. SHAP is both consistent and accurate as a way to allocate feature importances. The details are in a recent paper by Lundberg and Lee (papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf)

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

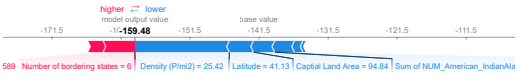
```
Collecting shap==0.23.0
  Downloading https://files.pythonhosted.org/packages/60/0d/8bd076821f7230edb2892ad982ea91ca25f2f925466563272e61eae891c6/shap-0.23.0.tar.gz (182kB)
     |████████████████████████████████| 184kB 2.6MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (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: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0.23.0) (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->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: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->shap==0.23.0) (2018.9)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (2.1.3)
Requirement already satisfied: 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: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.3.3)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (0.8.1)
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0) (4.8.0)
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: 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: 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)
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)
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=235681 sha256=5603ae310a497a67d8b08f126db57be6be8b61f724c4ac1e3868d43d51c97877
  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 2.8MB/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 63.7MB/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/d8/312e83adf4c78663e17d802fe2440072376fee46cada1404f1727ed77a32/scikit_learn-0.22.2.post1-cp36-cp36m-manylinux1_x86_64.whl (7.1MB)
     |████████████████████████████████| 7.1MB 41.5MB/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 175kB/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 10.3MB/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 50.3MB/s
Collecting pytz>=2017.2
  Downloading https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e59d7d4efda580b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.py3-none-any.whl (510kB)
     |████████████████████████████████| 512kB 48.3MB/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 49.7MB/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=394118 sha256=0a0aa13f591a373298b5792d516d10896c3c2c014b492f8df9867d36b4724b85
  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, pytz, six, python-dateutil, 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

model_shap =     XGBRegressor(n_estimators=15,
                              objective='reg:squarederror',
                              max_depth=1, # try deeper trees because of high cardinality categoricals
                              learning_rate=0.41, # try a higher learning rate
                              random_state=42,
                              n_jobs=-1)

#encoder = ce.OrdinalEncoder()
#X_train_shap_encoded = encoder.fit_transform(X_train)
#X_val_shap_encoded = encoder.transform(X_val)

#eval_set = [(X_train_shap_encoded, y_train),
#            (X_val_shap_encoded, y_val)]
eval_set = [(X_train, y_train),
            (X_val, y_val)]

#model_shap.fit(X_train_shap_encoded,
model_shap.fit(X_train,
               y_train,
               eval_set=eval_set,
               eval_metric='rmse',
               early_stopping_rounds=50)

shap.initjs()
#explainer = shap.TreeExplainer(model2)
explainer = shap.TreeExplainer(model_shap)
#shap_values = explainer.shap_values(X_train_shap_encoded)
shap_values = explainer.shap_values(X_train)
```

```
i = 32
shap.force_plot(explainer.expected_value,
                shap_values[i],
#                X_train_shap_encoded.loc[i],
                features=X_train.loc[i],
#                X_train_shap_encoded.columns)
                feature_names=X_train.columns)
```

```
[0]	validation_0-rmse:91.9687	validation_1-rmse:92.6396
Multiple eval metrics have been passed: 'validation_1-rmse' will be used for early stopping.

Will train until validation_1-rmse hasn't improved in 50 rounds.
[1]	validation_0-rmse:59.8496	validation_1-rmse:59.9746
[2]	validation_0-rmse:41.644	validation_1-rmse:42.4277
[3]	validation_0-rmse:31.6015	validation_1-rmse:32.9433
[4]	validation_0-rmse:26.6825	validation_1-rmse:27.7771
[5]	validation_0-rmse:24.0516	validation_1-rmse:26.4663
[6]	validation_0-rmse:22.2411	validation_1-rmse:24.8219
[7]	validation_0-rmse:20.9869	validation_1-rmse:23.9266
[8]	validation_0-rmse:19.944	validation_1-rmse:22.0353
[9]	validation_0-rmse:19.1823	validation_1-rmse:20.8479
[10]	validation_0-rmse:18.435	validation_1-rmse:19.5501
[11]	validation_0-rmse:17.6743	validation_1-rmse:20.4497
[12]	validation_0-rmse:17.0584	validation_1-rmse:21.0673
[13]	validation_0-rmse:16.5325	validation_1-rmse:20.8832
[14]	validation_0-rmse:15.9991	validation_1-rmse:20.0731
```




```
# 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(model_shap)
        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 d for state'),
                zaxis=dict(title= 'Shapley Force')
)
)
fig = go.Figure(surface, layout)
fig.show()
```