Project 6 - US Elections 2020 Analysis

January 27, 2021

1 US Elections 2020 Analysis, Visualization, Comparison and Machine-Learning

1.0.1 This project was made as a part of the Data Insight Program of 2020

Author : Omar Ossama Mahmoud Ahmed

ID #: 87

2 Introduction to US Elections and Electoral Vote

United States of America holds Presidential elections every four years on the 1st of November, this event is regarded as one of the most important political events all over the globe, due to the gravity of the US Political influence.

The United States of America holds Presidential elections in a unique system called "The United States Electoral College".

2.1 Introduction to The United States Electoral College.

The United States Electoral College is the group of presidential electors required by the Constitution to form every four years for the sole purpose of electing the president and vice president. Each state appoints electors according to its legislature, equal in number to its congressional delegation (senators and representatives). Federal office holders cannot be electors. Of the current 538 electors, an absolute majority of 270 or more electoral votes is required to elect the president and vice president. If no candidate achieves an absolute majority there, a contingent election is held by the United States House of Representatives to elect the president, and by the United States Senate to elect the vice president.

Currently, the states and the District of Columbia hold a statewide or districtwide popular vote on Election Day in November to choose electors based upon how they have pledged to vote for president and vice president, with some state laws against faithless electors. All jurisdictions use a winner-take-all method to choose their electors, except for Maine and Nebraska, which choose one elector per congressional district and two electors for the ticket with the highest statewide vote. The electors meet and vote in December and the inauguration of the president and vice president takes place in January.

The appropriateness of the Electoral College system is a matter of ongoing debate. Supporters argue that it is a fundamental component of American federalism by preserving the constitutional role of the states in presidential elections. Its implementation by the states may leave it open to

criticism; winner-take-all systems, especially in populous states, may not align with the principle of "one person, one vote". Almost 10% of presidential elections under the system have not elected the winners of the nationwide popular vote.

2.1.1 Sources

- Wikipedia
- Census
- CNN
- Kaggle

2.1.2 Datasets

- Election, COVID, and Demographic Data by County Kaggle by Ethan Schacht
- US Census Data US Census by US Census Data
- This dataset is updated on annual basis by US Census Data
- US Census Demographic Data Kaggle by MuonNeutrino

3 Table of Content :

- Statistical Analysis of US Elections
  - Correlation analysis of cleaned data
    - Elimnating Location data columns
    - Setting Graph Style
    - Calculating Correlation
    - Formating Graphical Output
  - Ploting CDF for confirmed cases counts
    - Income
    - Poverty
    - Unemployment
    - Employment
    - Men
    - Women
- General Data Analysis
  - Exploratory Data Analysis
  - Total Votes Per State
  - Total Electoral Votes
  - Election total votes & Total Electoral Votes Per US State
  - Election total votes & Total COVID-19 Cases Per US State
  - Election total votes & Total Unemployment Per US State
  - Election total votes & Poverty Per US State
  - Election total votes & Income Per US State
- 3D Geospacial Maps
  - Parsing data to JSON
  - Executing Maps from json file
  - US Elections 2020 VS Income and Total population
  - US Elections 2020 VS Unemployment and Poverty
- Machine Learning Process
  - Subsetting ML Dataset
  - Redefining Categorical Data
  - Cleaning Data and setting Target column
  - Feature selection
  - Splitting Data into Training and Testing Data
  - Defining Pipeline used
    - Average CV score on the training set was: 0.975
    - Using GradientBoostClassifier
  - Fitting Data to the pipeline
  - Appending Results to variable
- Testing ML Model
  - Defining list of results
  - Testing Model nth times
  - Predicting Data
  - Grouping Results
  - Formating Data Output
  - Printing Percentage Result
- Conclusion
- Final Thoughts

3.0.1 Enabling Widget Extensions in NB

[1]:
```
!jupyter nbextension enable --py widgetsnbextension
```

```
Enabling notebook extension jupyter-js-widgets/extension…
      - Validating: ok
```

3.1 Importing Libraries

- Pandas : for dataset handeling
- Numpy : Support for Pandas and calculations
- GradientBoostingClassifier: for Machine Learning
- train_test_split: for Machine Learning
- make_pipeline: for Machine Learning
- Normalizer: for Machine Learning
- Math : for mathimatical operations
- Matplotlib : for visualization (basic)
- json : for JSON Manipulation
- csv : for CSV Manipulation & import
- pydeck : for 3D Map visualization
- ast : for JSON parsing
- jinja2 : templating syntax library
- HTML : for HTML Parsing
- Seaborn : for visualization and plotting (Presentable)
- pycountry : Library for getting continent (name) to from their country names
- plotly : for interative plots

[2]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import seaborn as sns
import matplotlib.pyplot as plt
import json
import csv
import math
import pydeck as pdk
import ast
import jinja2
from ipywidgets import HTML
```

3.2 Defining Functions

- ***create_legend() :*** for HTML Legend Creation
- ***ecdf() :*** for CDF calculation

[3]:
```
def create_legend(labels: list) -> HTML:
    """Creates an HTML legend from a list dictionary of the format {'text':␣
↪str, 'color': [r, g, b]}"""
    labels = list(labels)
    for label in labels:
        assert label['color'] and label['text']
        assert len(label['color']) in (3, 4)
        label['color'] = ', '.join([str(c) for c in label['color']])
    legend_template = jinja2.Template('''
    <style>
      .legend {
        width: 300px;
      }
      .square {
        height: 10px;
        width: 10px;
        border: 1px solid grey;
      }
      .left {
        float: left;
      }
      .right {
        float: right;
      }
    </style>
    {% for label in labels %}
    <div class='legend'>
      <div class="square left" style="background:rgba({{ label['color'] }})"></
↪div>
      <span class="right">{{label['text']}}</span>
      <br />
    </div>
    {% endfor %}
    ''')
    html_str = legend_template.render(labels=labels)
    return HTML(html_str)


def ecdf(data):
    #credits DataCamp Justin Bois
    """Compute ECDF for a one-dimensional array of measurements."""
    # Number of data points: n
    n = len(data)
```

```
    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n

    return x, y
```

3.3 Importing Datasets from Local files

US Elections Datasets

- ***actual votes.csv :*** Actual US Election results
- ***trump_biden_polls.csv :*** US Polling Results

Supplementary Datasets

- ***country_statistics.csv :*** US Demographic Statistics
- ***electoral_votes.csv :*** US Electoral Collage Counts per state
- ***locations.csv :*** US States Geographical centers
- ***states_names.csv :*** US States ANSI Codes

[4]:
```
# Importing US Elections Datasets
elections = pd.read_csv(r'Final Data\\actual votes.csv')
polls = pd.read_csv(r'Final Data\\trump_biden_polls.csv')

# Importing Supplementary Datasets
electoral_votes = pd.read_excel(r'Final Data\\electoral_votes.xlsx')
country_statistics = pd.read_csv(r'Final Data\\county_statistics.csv')
locations = pd.read_excel(r'Final Data\\location.xlsx')
states_names = pd.read_excel(r'Final Data\\States.xlsx')
```

3.4 Exploring Imported Datasets

Using .head(),.describe() and .info() methods of pandas

[5]:
```
display(elections.head())
display(elections.describe())
display(elections.info())

display(polls.head())
display(polls.describe())
display(polls.info())

display(country_statistics.head())
display(country_statistics.describe())
display(country_statistics.info())
```

```
  state  percentage_Donald_Trump  percentage_Joe_Biden  total_votes  \
0    AK                    53.100                43.000     356845.0
1    AL                     0.715                 0.270      27639.0
2    AL                     0.762                 0.223     108945.0
3    AL                     0.536                 0.456      10457.0
4    AL                     0.784                 0.207       9573.0

   votes_Donald_Trump  votes_Joe_Biden
0            189543.0         153502.0
1             19764.0           7450.0
2             83055.0          24344.0
3              5605.0           4772.0
4              7508.0           1982.0
```

```
       percentage_Donald_Trump  percentage_Joe_Biden   total_votes  \
count              4451.000000           4451.000000  4.594000e+03
mean                  0.597729              0.404068  3.347056e+04
std                   0.808374              0.663833  1.194844e+05
min                   0.000000              0.031000  0.000000e+00
25%                   0.454000              0.245000  2.292250e+03
50%                   0.604000              0.376000  7.514000e+03
75%                   0.737000              0.524000  2.016500e+04
max                  53.100000             43.000000  4.139895e+06

       votes_Donald_Trump  votes_Joe_Biden
count        4.594000e+03     4.594000e+03
mean         1.583196e+04     1.703780e+04
std          4.427545e+04     7.654748e+04
min          0.000000e+00     0.000000e+00
25%          1.215750e+03     7.302500e+02
50%          4.403000e+03     2.417500e+03
75%          1.239350e+04     7.452250e+03
max          1.107090e+06     2.947568e+06
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4768 entries, 0 to 4767
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   4768 non-null   object
 1   percentage_Donald_Trump  4451 non-null   float64
 2   percentage_Joe_Biden    4451 non-null   float64
 3   total_votes             4594 non-null   float64
 4   votes_Donald_Trump      4594 non-null   float64
 5   votes_Joe_Biden         4594 non-null   float64
dtypes: float64(5), object(1)
memory usage: 223.6+ KB

None
```

```
   question_id  poll_id  cycle         state  pollster_id  \
0       136283    72621   2020          Iowa          383
1       136283    72621   2020          Iowa          383
2       136322    72647   2020  Pennsylvania          461
3       136322    72647   2020  Pennsylvania          461
4       136322    72647   2020  Pennsylvania          461

                            pollster sponsor_ids sponsors  \
0               Public Policy Polling         NaN      NaN
1               Public Policy Polling         NaN      NaN
2  Susquehanna Polling & Research Inc.         NaN      NaN
3  Susquehanna Polling & Research Inc.         NaN      NaN
4  Susquehanna Polling & Research Inc.         NaN      NaN

                         display_name  pollster_rating_id  …  \
0               Public Policy Polling               263.0  …
1               Public Policy Polling               263.0  …
2  Susquehanna Polling & Research Inc.               326.0  …
3  Susquehanna Polling & Research Inc.               326.0  …
4  Susquehanna Polling & Research Inc.               326.0  …

      created_at notes                                                url  \
0  11/2/20 09:02   NaN  https://www.publicpolicypolling.com/wp-content…
1  11/2/20 09:02   NaN  https://www.publicpolicypolling.com/wp-content…
2  11/2/20 12:49   NaN  https://www.realclearpolitics.com/docs/2020/Su…
3  11/2/20 12:49   NaN  https://www.realclearpolitics.com/docs/2020/Su…
4  11/2/20 12:49   NaN  https://www.realclearpolitics.com/docs/2020/Su…

     stage race_id     answer candidate_id       candidate_name  \
0  general    6223      Biden        13256  Joseph R. Biden Jr.
1  general    6223      Trump        13254         Donald Trump
2  general    6249      Biden        13256  Joseph R. Biden Jr.
3  general    6249      Trump        13254         Donald Trump
4  general    6249  Jorgensen        14611         Jo Jorgensen

   candidate_party   pct
0              DEM  49.0
1              REP  48.0
2              DEM  48.4
3              REP  49.2
4              LIB   1.4

[5 rows x 38 columns]
```

```
         question_id       poll_id    cycle   pollster_id  pollster_rating_id  \
count   16438.000000  16438.000000  16438.0  16438.000000        16425.000000
mean   125958.788235  68190.269315   2020.0   1072.308249          299.262222
std     11813.231604   4481.450463      0.0    403.882143          132.974792
```

```
min     92078.000000  57025.000000   2020.0     11.000000            3.000000
25%    121449.000000  65041.000000   2020.0    788.000000          218.000000
50%    131186.500000  70388.000000   2020.0   1193.000000          324.000000
75%    134101.000000  71582.750000   2020.0   1323.000000          326.000000
max    136606.000000  72803.000000   2020.0   1642.000000          636.000000

        sample_size  seat_number  seat_name       race_id  candidate_id  \
count  16436.000000      16438.0        0.0  16438.000000  16438.000000
mean    2990.533950          0.0        NaN   6247.780204  13356.531573
std     4772.879308          0.0        NaN    207.763162    360.873271
min       88.000000          0.0        NaN   6210.000000  13253.000000
25%      797.000000          0.0        NaN   6210.000000  13254.000000
50%     1072.000000          0.0        NaN   6224.000000  13256.000000
75%     2922.000000          0.0        NaN   6245.000000  13256.000000
max    71789.000000          0.0        NaN   8718.000000  16083.000000

                pct
count  16438.000000
mean      43.403030
std       14.008338
min        0.000000
25%       41.000000
50%       46.000000
75%       51.000000
max       94.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16438 entries, 0 to 16437
Data columns (total 38 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   question_id                16438 non-null  int64
 1   poll_id                    16438 non-null  int64
 2   cycle                      16438 non-null  int64
 3   state                      11385 non-null  object
 4   pollster_id                16438 non-null  int64
 5   pollster                   16438 non-null  object
 6   sponsor_ids                9968 non-null   object
 7   sponsors                   9968 non-null   object
 8   display_name               16438 non-null  object
 9   pollster_rating_id         16425 non-null  float64
 10  pollster_rating_name       16425 non-null  object
 11  fte_grade                  15253 non-null  object
 12  sample_size                16436 non-null  float64
 13  population                 16438 non-null  object
 14  population_full            16438 non-null  object
 15  methodology                16138 non-null  object
 16  office_type                16438 non-null  object
```

```
 17  seat_number                16438 non-null  int64
 18  seat_name                  0 non-null      float64
 19  start_date                 16438 non-null  object
 20  end_date                   16438 non-null  object
 21  election_date              16438 non-null  object
 22  sponsor_candidate          54 non-null     object
 23  internal                   16438 non-null  bool
 24  partisan                   743 non-null    object
 25  tracking                   6384 non-null   object
 26  nationwide_batch           16438 non-null  bool
 27  ranked_choice_reallocated  16438 non-null  bool
 28  created_at                 16438 non-null  object
 29  notes                      623 non-null    object
 30  url                        16230 non-null  object
 31  stage                      16438 non-null  object
 32  race_id                    16438 non-null  int64
 33  answer                     16438 non-null  object
 34  candidate_id               16438 non-null  int64
 35  candidate_name             16438 non-null  object
 36  candidate_party            16438 non-null  object
 37  pct                        16438 non-null  float64
dtypes: bool(3), float64(4), int64(7), object(24)
memory usage: 4.4+ MB

None
```

	i	county	state	percentage16_Donald_Trump	\
0	4955	all	AK	0.000	
1	107	Autauga	AL	0.734	
2	116	Baldwin	AL	0.774	
3	128	Barbour	AL	0.523	
4	197	Bibb	AL	0.770	

	percentage16_Hillary_Clinton	total_votes16	votes16_Donald_Trump	\
0	0.000	0.0	0.0	
1	0.240	24661.0	18110.0	
2	0.196	94090.0	72780.0	
3	0.467	10390.0	5431.0	
4	0.214	8748.0	6733.0	

	votes16_Hillary_Clinton	percentage20_Donald_Trump	percentage20_Joe_Biden	\
0	0.0	0.531	0.430	
1	5908.0	0.715	0.270	
2	18409.0	0.762	0.223	
3	4848.0	0.536	0.456	
4	1874.0	0.784	0.207	

	…	Walk	OtherTransp	WorkAtHome	MeanCommute	Employed	PrivateWork	\
0	…	3.925117	2.573868	2.166647	18.8	354045.0	33.1	

```
1  …  0.600000     1.300000    2.500000         25.8   24112.0         74.1
2  …  0.800000     1.100000    5.600000         27.0   89527.0         80.7
3  …  2.200000     1.700000    1.300000         23.4    8878.0         74.1
4  …  0.300000     1.700000    1.500000         30.0    8171.0         76.0

   PublicWork  SelfEmployed  FamilyWork  Unemployment
0        0.33           3.0    0.001077           4.0
1       20.20           5.6    0.100000           5.2
2       12.90           6.3    0.100000           5.5
3       19.10           6.5    0.300000          12.4
4       17.40           6.3    0.300000           8.2

[5 rows x 51 columns]
```

	i	percentage16_Donald_Trump	percentage16_Hillary_Clinton	\
count	4768.000000	3112.000000	3112.000000	
mean	2396.661074	0.635991	0.316711	
std	1394.381641	0.156489	0.153345	
min	0.000000	0.000000	0.000000	
25%	1191.750000	0.550000	0.204000	
50%	2383.500000	0.667000	0.284500	
75%	3615.250000	0.750250	0.399000	
max	4955.000000	0.953000	0.928000	

	total_votes16	votes16_Donald_Trump	votes16_Hillary_Clinton	\
count	3.112000e+03	3112.000000	3.112000e+03	
mean	4.090270e+04	19343.684769	1.956023e+04	
std	1.082553e+05	39125.636705	6.847920e+04	
min	0.000000e+00	0.000000	0.000000e+00	
25%	4.821000e+03	3206.000000	1.163250e+03	
50%	1.092950e+04	7113.000000	3.140000e+03	
75%	2.866450e+04	17391.750000	9.535250e+03	
max	2.314275e+06	590465.000000	1.654626e+06	

	percentage20_Donald_Trump	percentage20_Joe_Biden	total_votes20	\
count	4451.000000	4451.000000	4.594000e+03	
mean	0.585918	0.394504	3.347056e+04	
std	0.184103	0.181246	1.194844e+05	
min	0.000000	0.031000	0.000000e+00	
25%	0.454000	0.245000	2.292250e+03	
50%	0.604000	0.376000	7.514000e+03	
75%	0.737000	0.524000	2.016500e+04	
max	0.962000	1.000000	4.139895e+06	

	votes20_Donald_Trump	…	Walk	OtherTransp	WorkAtHome	\
count	4.594000e+03	…	3114.000000	3114.000000	3114.000000	
mean	1.583196e+04	…	3.035814	1.527834	4.796039	
std	4.427545e+04	…	2.953239	1.153157	3.078256	

```
min            0.000000e+00  …     0.000000     0.000000     0.000000
25%            1.215750e+03  …     1.400000     0.800000     2.900000
50%            4.403000e+03  …     2.300000     1.300000     4.100000
75%            1.239350e+04  …     3.800000     1.900000     5.800000
max            1.107090e+06  …    42.400000    13.800000    33.000000

       MeanCommute      Employed  PrivateWork   PublicWork  SelfEmployed  \
count  3114.000000  3.114000e+03  3114.000000  3114.000000   3114.000000
mean     23.466602  4.836197e+04    75.199775    16.737229      7.761593
std       5.508682  1.583616e+05     7.403020     6.013065      3.870077
min       6.600000  3.900000e+01    31.100000     0.330000      0.000000
25%      19.600000  4.605500e+03    71.800000    12.600000      5.200000
50%      23.200000  1.088350e+04    76.400000    15.700000      6.800000
75%      26.900000  2.977525e+04    80.300000    19.300000      9.100000
max      45.100000  4.805817e+06    88.800000    64.800000     38.000000

        FamilyWork  Unemployment
count  3114.000000   3114.000000
mean      0.282145      6.332595
std       0.451099      3.000809
min       0.000000      0.000000
25%       0.100000      4.400000
50%       0.200000      6.000000
75%       0.300000      7.800000
max       8.000000     28.700000

[8 rows x 49 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4768 entries, 0 to 4767
Data columns (total 51 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   i                               4768 non-null   int64
 1   county                          4768 non-null   object
 2   state                           4768 non-null   object
 3   percentage16_Donald_Trump       3112 non-null   float64
 4   percentage16_Hillary_Clinton    3112 non-null   float64
 5   total_votes16                   3112 non-null   float64
 6   votes16_Donald_Trump            3112 non-null   float64
 7   votes16_Hillary_Clinton         3112 non-null   float64
 8   percentage20_Donald_Trump       4451 non-null   float64
 9   percentage20_Joe_Biden          4451 non-null   float64
 10  total_votes20                   4594 non-null   float64
 11  votes20_Donald_Trump            4594 non-null   float64
 12  votes20_Joe_Biden               4594 non-null   float64
 13  lat                             3222 non-null   float64
 14  long                            3222 non-null   float64
```

```
 15  cases                       3222 non-null   float64
 16  deaths                      3222 non-null   float64
 17  TotalPop                    3114 non-null   float64
 18  Men                         3114 non-null   float64
 19  Women                       3114 non-null   float64
 20  Hispanic                    3114 non-null   float64
 21  White                       3114 non-null   float64
 22  Black                       3114 non-null   float64
 23  Native                      3114 non-null   float64
 24  Asian                       3114 non-null   float64
 25  Pacific                     3114 non-null   float64
 26  VotingAgeCitizen            3114 non-null   float64
 27  Income                      3114 non-null   float64
 28  IncomeErr                   3114 non-null   float64
 29  IncomePerCap                3114 non-null   float64
 30  IncomePerCapErr             3114 non-null   float64
 31  Poverty                     3114 non-null   float64
 32  ChildPoverty                3113 non-null   float64
 33  Professional                3114 non-null   float64
 34  Service                     3114 non-null   float64
 35  Office                      3114 non-null   float64
 36  Construction                3114 non-null   float64
 37  Production                  3114 non-null   float64
 38  Drive                       3114 non-null   float64
 39  Carpool                     3114 non-null   float64
 40  Transit                     3114 non-null   float64
 41  Walk                        3114 non-null   float64
 42  OtherTransp                 3114 non-null   float64
 43  WorkAtHome                  3114 non-null   float64
 44  MeanCommute                 3114 non-null   float64
 45  Employed                    3114 non-null   float64
 46  PrivateWork                 3114 non-null   float64
 47  PublicWork                  3114 non-null   float64
 48  SelfEmployed                3114 non-null   float64
 49  FamilyWork                  3114 non-null   float64
 50  Unemployment                3114 non-null   float64
dtypes: float64(48), int64(1), object(2)
memory usage: 1.9+ MB

None
```

3.5 Preprocessing of Datasets

3.5.1 Cleaning, Feature Engineering and Merging Datasets

[6]:
```
# Filtering and simplifing Datasets
country_statistics =␣
 ↪country_statistics[['state','cases','deaths','TotalPop','Men','Women','VotingAgeCitizen','I
```

```
elections =␣
 ↪elections[['state','total_votes','votes_Donald_Trump','votes_Joe_Biden']]

# Defining columns aggregates
percentage_of_total=␣
 ↪['Hispanic','White','Black','Asian','Pacific','Native','Poverty','Unemployment']
percentage_of_employment =␣
 ↪['Professional','Service','Office','Construction','Production','FamilyWork','SelfEmployed',



# Fixing aggregating columns formating
for i in percentage_of_total:
    for j in range(len(country_statistics)):
        country_statistics[i][j] = (country_statistics[i][j] / 100) *␣
 ↪country_statistics['TotalPop'][j]

for i in percentage_of_employment:
    for j in range(len(country_statistics)):
        country_statistics[i][j] = (country_statistics[i][j] / 100) *␣
 ↪country_statistics['Employed'][j]



# Pivoting and grouping statistics by state for further analysis


# Grouping by summing

# Defining Mean and Median columns
mean_columns = country_statistics[['state','IncomePerCap','Income']]
sum_columns = country_statistics.drop(['IncomePerCap','Income'],axis=1)

temp1 = mean_columns.groupby('state').min()
temp2 = sum_columns.groupby('state').sum()


# Joining temp dataframes
states_df = temp1.join(temp2).reset_index()

elections = elections.groupby('state').sum()
states_df = states_df.merge(elections,how='left',on='state').reset_index()


# Merging states with their corresponding electoral collage votes
states_df = states_df.merge(electoral_votes,how='right',on='state')
```

```
# Feature engineering election results column (answer)
states_df['answer'] = 'Tie'

for i in range(len(states_df)):
    if ((states_df.votes_Joe_Biden[i]) > (states_df.votes_Donald_Trump[i])):
        states_df.answer[i] = 'Biden'
    if ((states_df.votes_Joe_Biden[i]) < (states_df.votes_Donald_Trump[i])):
        states_df.answer[i] = 'Trump'



# Merging Polling Data with main dataframe
polls = polls.merge(states_names,on = 'state',how = 'left')
polls = polls[['state2','sample_size','pct','answer']]


# Filtering polls for most important candidates
polls = polls[(polls.answer == 'Biden') | (polls.answer == 'Trump')]
polls.reset_index(inplace=True,drop=True)


# Feature engineering vote counts from pct
polls['votes'] = 0
for i in range(len(polls)):
    polls.votes[i] = (polls.pct[i] / 100) * (polls.sample_size[i])
polls = polls[['state2','sample_size','votes','answer']]


# Pivoting polling data to states
polls = polls.
 ↪pivot_table(values=['sample_size','votes'],index='state2',columns='answer',aggfunc=np.
 ↪sum).reset_index()
polls.columns = polls.columns.map('_'.join)


# Feature engineering calculating sample size
polls['sample_size'] = 1
for i in range(len(polls)):
    polls.sample_size[i] = (polls.sample_size_Biden[i] + polls.
 ↪sample_size_Trump[i])/2
polls = polls[['state2_','sample_size','votes_Biden','votes_Trump']]


# Fixing column names
```

```
polls.rename(columns={'state2_':'state','sample_size':
 ↪'polls_sample','votes_Biden':'polls_biden','votes_Trump':
 ↪'polls_trump'},inplace=True)


# Merging both main dataframes and geo locations
merged_df = states_df.merge(polls,on='state',how='right')
merged_df = merged_df.merge(locations,on='state',how='left')


# Creating color column to visualize candidates parties
color = {'color':['[255, 20, 20]','[20, 138, 255]']}
color = pd.DataFrame(color,index=['Trump','Biden']).reset_index()
color = color.rename(columns={'index':'answer'})
df = merged_df.merge(pd.DataFrame(color),on='answer',how='left')


# Saving Dataframe for easier recall
df.to_csv(r'df.csv')
```

3.5.2 Exploring Cleaned Datasets

Using .head(),.describe() and .info() methods of pandas

[7]:
```
display(df.head())
display(df.describe())
display(df.info())
```

	index	state	IncomePerCap	Income	cases	deaths	TotalPop	\
0	0	AK	36978.0	60147.0	28892.0	118.0	731545.0	
1	1	AL	13449.0	20954.0	193985.0	2973.0	4850771.0	
2	2	AR	13142.0	26652.0	113057.0	1958.0	2977944.0	
3	3	AZ	13865.0	32360.0	247473.0	5979.0	6809946.0	
4	4	CA	17303.0	36563.0	935878.0	17671.0	38982847.0	

	Men	Women	VotingAgeCitizen	…	votes_Donald_Trump	\
0	380433.0	351112.0	533151.0	…	189543.0	
1	2350806.0	2499965.0	3651914.0	…	1434159.0	
2	1461651.0	1516293.0	2183895.0	…	757052.0	
3	3385055.0	3424891.0	4690177.0	…	1651812.0	
4	19366579.0	19616268.0	24970109.0	…	5416035.0	

	votes_Joe_Biden	electoral vote	answer	polls_sample	polls_biden	\
0	153502.0	3	Trump	46559	20347.7190	
1	843473.0	9	Trump	121507	47264.7602	
2	418051.0	6	Trump	73117	28913.7878	
3	1663447.0	11	Biden	371192	182676.5003	
4	10339137.0	55	Biden	1052037	621968.7055	

```
   polls_trump      lat      long           color
0   24669.5808  64.0685 -152.2782   [255, 20, 20]
1   71625.3169  32.7794  -86.8287   [255, 20, 20]
2   42581.6581  34.8938  -92.4426   [255, 20, 20]
3  175638.3002  34.2744 -111.6602  [20, 138, 255]
4  386238.6593  37.1841 -119.4696  [20, 138, 255]

[5 rows x 39 columns]
```

	index	IncomePerCap	Income	cases	deaths	\
count	50.000000	50.000000	50.00000	50.000000	50.000000	
mean	25.360000	18494.640000	35771.34000	182578.060000	4589.020000	
std	14.790731	5556.191607	10534.00139	211231.077834	6333.067975	
min	0.000000	9334.000000	19264.00000	2196.000000	58.000000	
25%	13.250000	14115.750000	29050.25000	47255.000000	663.250000	
50%	25.500000	17476.000000	36245.00000	122305.500000	2327.000000	
75%	37.750000	20588.000000	40024.00000	210163.500000	5131.250000	
max	50.000000	36978.000000	62553.00000	936816.000000	33535.000000	

	TotalPop	Men	Women	VotingAgeCitizen	\
count	5.000000e+01	5.000000e+01	5.000000e+01	5.000000e+01	
mean	6.406500e+06	3.153876e+06	3.252624e+06	4.528514e+06	
std	7.211922e+06	3.560995e+06	3.651469e+06	4.765890e+06	
min	5.832000e+05	2.983010e+05	2.848990e+05	4.328140e+05	
25%	1.851112e+06	9.166025e+05	9.345100e+05	1.374197e+06	
50%	4.543918e+06	2.230132e+06	2.313786e+06	3.386388e+06	
75%	7.079962e+06	3.531930e+06	3.565782e+06	5.043953e+06	
max	3.898285e+07	1.936658e+07	1.961627e+07	2.497011e+07	

	Employed	…	PrivateWork	total_votes	votes_Donald_Trump	\
count	5.000000e+01	…	5.000000e+01	5.000000e+01	5.000000e+01	
mean	3.004829e+06	…	2.402793e+06	3.069091e+06	1.454315e+06	
std	3.337960e+06	…	2.664015e+06	3.144985e+06	1.367378e+06	
min	2.936330e+05	…	1.171889e+05	2.767650e+05	1.124880e+05	
25%	7.812938e+05	…	5.905542e+05	8.813052e+05	5.448772e+05	
50%	1.984694e+06	…	1.582150e+06	2.228981e+06	1.075661e+06	
75%	3.498785e+06	…	2.832144e+06	3.874003e+06	1.723388e+06	
max	1.799392e+07	…	1.406529e+07	1.609273e+07	5.866019e+06	

	votes_Joe_Biden	electoral vote	polls_sample	polls_biden	\
count	5.000000e+01	50.00000	5.000000e+01	50.000000	
mean	1.559719e+06	10.70000	2.203278e+05	112072.712142	
std	1.805060e+06	9.72321	2.158373e+05	116150.225453	
min	7.349100e+04	3.00000	2.353400e+04	7505.091400	
25%	4.186275e+05	5.00000	6.996175e+04	29462.607300	
50%	9.676010e+05	8.00000	1.490950e+05	71056.309400	
75%	2.304360e+06	11.75000	3.133268e+05	154748.976150	

```
max       1.033914e+07        55.00000  1.052037e+06  621968.705500

         polls_trump        lat        long
count      50.000000  50.000000   50.000000
mean   100918.732494  39.745206  -93.677304
std     95448.005389   6.316332   19.240868
min      9304.352000  20.292700 -156.373700
25%     38360.280125  36.189525 -104.277325
50%     68481.239350  39.611550  -89.830950
75%    125070.414325  43.509300  -78.987200
max    408622.750800  64.068500  -69.242800

[8 rows x 36 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50 entries, 0 to 49
Data columns (total 39 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   index               50 non-null     int64
 1   state               50 non-null     object
 2   IncomePerCap        50 non-null     float64
 3   Income              50 non-null     float64
 4   cases               50 non-null     float64
 5   deaths              50 non-null     float64
 6   TotalPop            50 non-null     float64
 7   Men                 50 non-null     float64
 8   Women               50 non-null     float64
 9   VotingAgeCitizen    50 non-null     float64
 10  Employed            50 non-null     float64
 11  Hispanic            50 non-null     float64
 12  White               50 non-null     float64
 13  Black               50 non-null     float64
 14  Asian               50 non-null     float64
 15  Pacific             50 non-null     float64
 16  Native              50 non-null     float64
 17  Poverty             50 non-null     float64
 18  Unemployment        50 non-null     float64
 19  Professional        50 non-null     float64
 20  Service             50 non-null     float64
 21  Office              50 non-null     float64
 22  Construction        50 non-null     float64
 23  Production          50 non-null     float64
 24  FamilyWork          50 non-null     float64
 25  SelfEmployed        50 non-null     float64
 26  PublicWork          50 non-null     float64
 27  PrivateWork         50 non-null     float64
 28  total_votes         50 non-null     float64
```

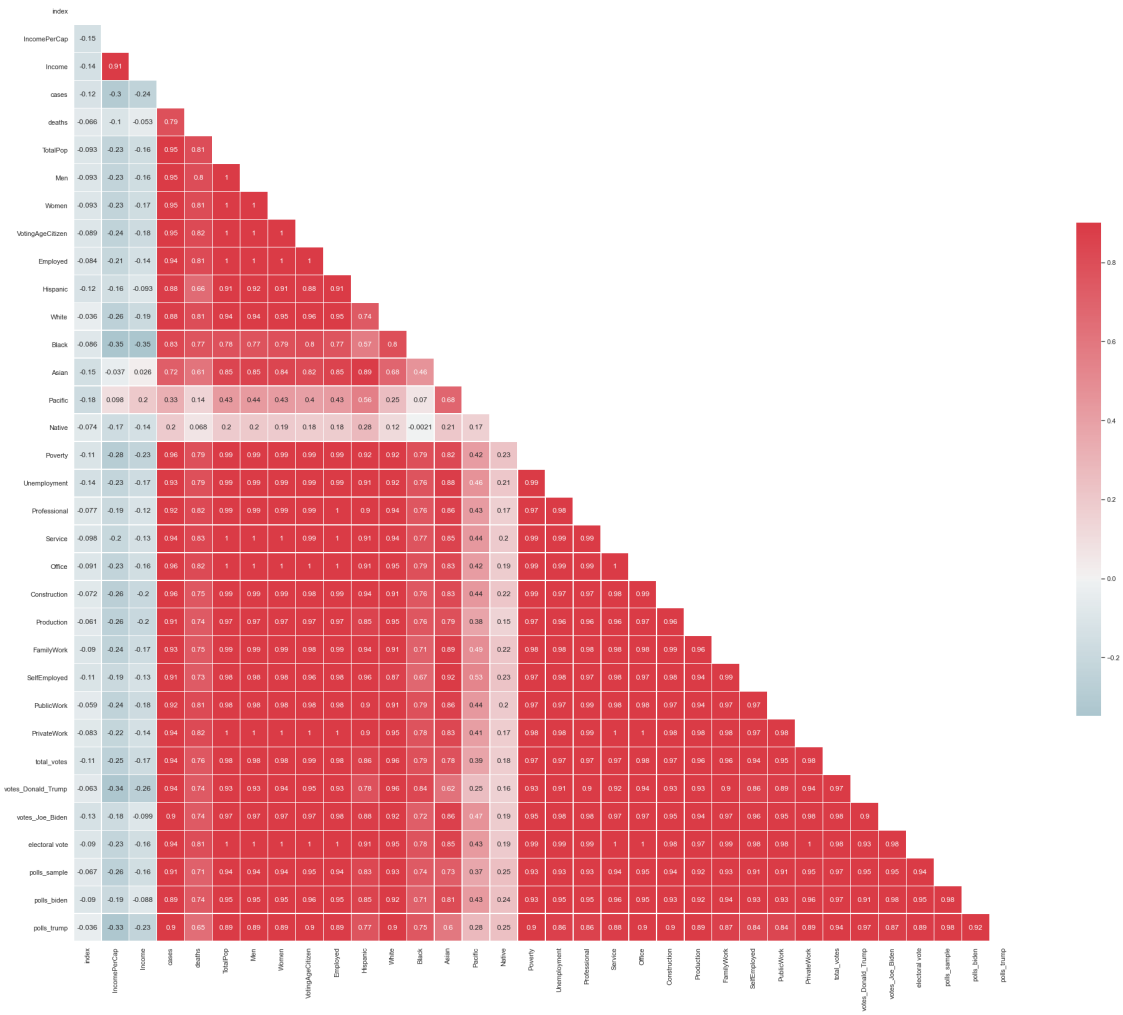
```
 29  votes_Donald_Trump  50 non-null     float64
 30  votes_Joe_Biden     50 non-null     float64
 31  electoral vote      50 non-null     int64
 32  answer              50 non-null     object
 33  polls_sample        50 non-null     int64
 34  polls_biden         50 non-null     float64
 35  polls_trump         50 non-null     float64
 36  lat                 50 non-null     float64
 37  long                50 non-null     float64
 38  color               50 non-null     object
dtypes: float64(33), int64(3), object(3)
memory usage: 15.6+ KB

None
```

3.6 Statistical Analysis of US Elections Dataset

3.6.1 Correlation analysis of cleaned data

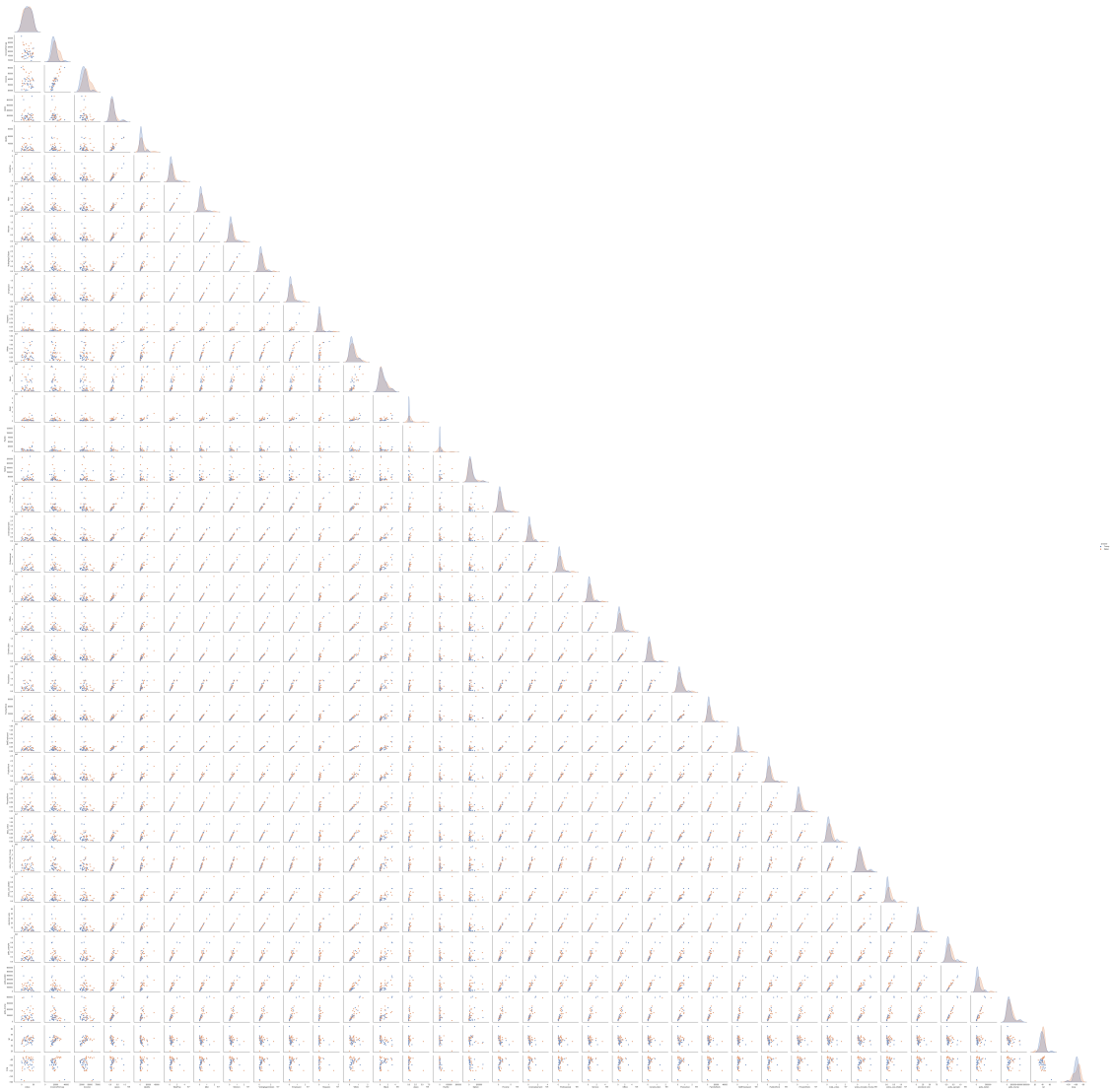
Using sns.corr() method to find correlations between data knowing that correlation does not necessarily mean causation

[8]:
```
# Elimnating Location data columns
df2 = df.loc[:, ((df.columns != 'lat')&(df.columns != 'long'))]

# Setting Graph Style
sns.set(style='white')

# Calculating Correlation
corr = df2.corr()

# Formating Graphical Output
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(35, 30))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.9, center=0, square=True,␣
 ↪linewidths=.5, annot=True,cbar_kws={'shrink': .5});
```



[9]:
```
sns.pairplot(df, hue="answer", corner=True)
```

[9]: <seaborn.axisgrid.PairGrid at 0x14894d18700>



Correlation shows:

- A strong positive correlation between almost every element which is to be expected.
- A strong positive correlation between covid-19 cases and votes for trump columns.
- A weak positive correlation between income and general votes.

3.6.2 Ploting CDF for most correlated features

Using ecdf() function and plotly library to plot CDF for statistical distribution

[10]:
```
x,y = list(ecdf(df.Income))
x1,y1 = list(ecdf(df.Poverty))
x2,y2 = list(ecdf(df.Unemployment))
x3,y3 = list(ecdf(df.Employed))
```

```
x4,y4 = list(ecdf(df.Men))
x5,y5 = list(ecdf(df.Women))

fig = make_subplots(rows=2, cols=3)
fig.add_trace(go.Scatter(x= x,y = y,name = 'Income'),row=1, col=1)
fig.add_trace(go.Scatter(x= x1,y = y1,name = 'Poverty'),row=1, col=2)
fig.add_trace(go.Scatter(x= x2,y = y2,name = 'Unemployment'),row=1, col=3)
fig.add_trace(go.Scatter(x= x3,y = y3,name = 'Employment'),row=2, col=1)
fig.add_trace(go.Scatter(x= x4,y = y4,name = 'Men'),row=2, col=2)
fig.add_trace(go.Scatter(x= x5,y = y5,name = 'Women'),row=2, col=3)

fig.update_layout(height=500, width=1000, title_text="Cumulative distribution␣
↪functions")
fig.show()
```

CDF shows:

- 80% of the States has avarage Income Below 41K with maximum of 65K, Which shows almost even destribution of income.
- 90% of the States has Poverty Counts Below 1.74M with maximum of 5.9M, Which shows that 10% of the states has 5 times more poverty than the rest of the states, Yet this could be an effect of total population.
- 90% of the States has Unemployment Counts Below 833K with maximum of 3.02M, Which shows that 10% of the states has 4 times more Unemployment than the rest of the states, Yet this could be an effect of total population.
- 90% of the States has Employment Counts Below 6.09M with maximum of 18M, Which shows that 10% of the states has 2 times more Employment than the rest of the states, Yet this could be an effect of total population.
- Gender distribution in most states are almost equal in count.

3.7 General Data Analysis

3.7.1 Exploratory Data Analysis

[11]:
```
# Creating bar plot of the total votes for both candidates
fig = go.Figure(data=[
    go.Bar(name='Biden', x=df.state, y=df.votes_Joe_Biden),
    go.Bar(name='Trump', x=df.state, y=df.votes_Donald_Trump)
])

# Adding average indicator line for Trumps total votes
fig.add_shape(
        go.layout.Shape(
            type="line",
            x0=0,
            y0=df.votes_Donald_Trump.mean(),
            x1=len(y),
            y1=df.votes_Donald_Trump.mean(),
```

```
            line=dict(
                color="red",
                width=1,
                dash="dash"
            ),
    ))

# Adding average indicator line for Bidens total votes
fig.add_shape(
        go.layout.Shape(
            type="line",
            x0=0,
            y0=df.votes_Joe_Biden.mean(),
            x1=len(y),
            y1=df.votes_Joe_Biden.mean(),
            line=dict(
                color="blue",
                width=1,
                dash="dash"
            ),
    ))

# Change the bar mode
fig.update_layout(barmode='group',height=600,width=950,title='Total Votes Per␣
 ↪State',xaxis_title="States",
                  yaxis_title="Total Votes",
                  legend_title="Candidates")
```

Figure shows:

- Bidens Average total votes is slightly higher than Trumps with less than 100K in difference.
- Biden excels over Trump with huge difference in California.

[12]:
```
# Creating Bar plot of total electoral votes for both candidates
fig = px.bar(df, x="state", y="electoral vote",
             color='answer',color_discrete_sequence=["red", "blue"])

# Adding average indicator line for Trumps total electoral votes
fig.add_shape(
        go.layout.Shape(
            type="line",
            x0=0,
            y0=df[df.answer == 'Trump']['electoral vote'].mean(),
            x1=len(y),
            y1=df[df.answer == 'Trump']['electoral vote'].mean(),
            line=dict(
                color="red",
```

```
                width=1,
                dash="dash",
            ),
    ))

# Adding average indicator line for Bidens total electoral votes
fig.add_shape(
        go.layout.Shape(
            type="line",
            x0=0,
            y0=df[df.answer == 'Biden']['electoral vote'].mean(),
            x1=len(y),
            y1=df[df.answer == 'Biden']['electoral vote'].mean(),
            line=dict(
                color="blue",
                width=1,
                dash="dash",
            ),
    ))

# Change the bar mode
fig.update_layout(barmode='group',height=600,width=950,title='Total Electoral␣
 ↪Votes Per State',xaxis_title="States",
                  yaxis_title="Total Electoral Votes",
                  legend_title="Candidates")
fig.show()
```

Figure shows:

- Bidens Average total Electoral votes is slightly higher than Trumps with less than 5 electoral votes in difference.
- Biden excels over Trump with California 55 Electoral votes.

[13]:
```
# Creating Scatter plot illustrating
fig = go.Figure()

# Defining Sets of Data for each candidate
trump = df[df.answer == 'Trump']
biden = df[df.answer == 'Biden']

# Adding Trumps Total Votes vs Electoral vote (size) per State
fig.add_trace(go.Scatter(
    x=trump.state,
    y=trump['votes_Donald_Trump'],
    text=trump['electoral vote'],
    marker=dict(color="red",size=trump['electoral vote']),
    showlegend=True,
```

```
    mode='markers',
    name='Trump',
    opacity=0.7
))

# Adding Bidens Total Votes vs Electoral vote (size) per State
fig.add_trace(go.Scatter(
    x=biden.state,
    y=biden['votes_Joe_Biden'],
    text=biden['electoral vote'],
    marker=dict(color="blue",size=biden['electoral vote']),
    showlegend=True,
    mode='markers',
    name = 'Biden',
    opacity=0.7
))

# Updating Title and axis names
fig.update_layout(title='Election total votes & Total Electoral Votes Per US␣
 ↪State',
                  xaxis_title="States",
                  yaxis_title="Total Votes",
                  legend_title="Candidates")

# Showing Final Figure
fig.show()
```

Figure shows:

- Again Bidens CA winning is the main feature of this graph.
- Yet Trumps manages to win TX and FL which are the second most awarding states with electoral votes.
- Figure shows a general advantage to Bidens Point sizes indicating more electoral votes on average.

[14]:
```
# Creating Scatter plot illustrating
fig = go.Figure()

# Defining Sets of Data for each candidate
trump = df[df.answer == 'Trump']
biden = df[df.answer == 'Biden']


# Adding Trumps Total Votes Percentage of Population vs Total COVID-19 Cases␣
 ↪Percentage of Population (size) per State
fig.add_trace(go.Scatter(
    x=trump.state,
```

```
    y=trump['votes_Donald_Trump']/trump['TotalPop']*100,
    text=trump['cases']/trump['TotalPop']*100,
    marker=dict(color = 'red',size=trump['cases']/trump['TotalPop']*100 *5),
    showlegend=True,
    mode='markers',
    name = 'Trump',
    opacity=0.7
))

# Adding Bidens Total Votes Percentage of Population vs Total COVID-19 Cases␣
 ↪Percentage of Population (size) per State
fig.add_trace(go.Scatter(
    x=biden.state,
    y=biden['votes_Joe_Biden']/biden['TotalPop']*100,
    text=biden['cases']/biden['TotalPop']*100,
    marker=dict(color="blue",size=biden['cases']/biden['TotalPop']*100 *5),
    showlegend=True,
    mode='markers',
    name = 'Biden',
    opacity=0.7
))

# Updating Title and axis names
fig.update_layout(title='Election total votes & Total COVID-19 Cases Per US␣
 ↪State',
                  xaxis_title="States",
                  yaxis_title="Total Votes % of Total Population",
                  legend_title="Candidates")

# Showing Final Figure
fig.show()
```

Figure shows:

- States who voted for Biden seem to have less COVID-19 Cases than those who voted for Trump.
- Does this indicate more educated states voted for Biden? further analysis is required in this area.

[15]:
```
# Creating Scatter plot illustrating
fig = go.Figure()

# Defining Sets of Data for each candidate
trump = df[df.answer == 'Trump']
biden = df[df.answer == 'Biden']
```

```
# Adding Trumps Total Votes Percentage of Population vs Unemployment Percentage␣
 ↪of Population (size) per State
fig.add_trace(go.Scatter(
    x=trump.state,
    y=trump['votes_Donald_Trump'],
    text=trump['Unemployment'],
    marker=dict(color = 'red',size=trump['Unemployment']/trump['TotalPop']*300),
    showlegend=True,
    mode='markers',
    name = 'Trump',
    opacity=0.7
))


# Adding Bidens Total Votes vs Total Unemployment Percentage of Population␣
 ↪(size) per State
fig.add_trace(go.Scatter(
    x=biden.state,
    y=biden['votes_Joe_Biden'],
    text=biden['Unemployment'],
    marker=dict(color="blue",size=biden['Unemployment']/biden['TotalPop']*300),
    showlegend=True,
    mode='markers',
    name = 'Biden',
    opacity=0.7
))

# Updating Title and axis names
fig.update_layout(title='Election total votes & Total Unemployment Per US␣
 ↪State',
                  xaxis_title="States",
                  yaxis_title="Total Votes",
                  legend_title="Candidates")

# Showing Final Figure
fig.show()
```

Figure shows:

- States Unemployment Rates seems to have almost no impact on total votes for each candidate.

[16]:
```
# Creating Scatter plot illustrating
fig = go.Figure()

# Defining Sets of Data for each candidate
trump = df[df.answer == 'Trump']
biden = df[df.answer == 'Biden']
```

```
# Adding Trumps Total Votes vs Poverty Percentage of Population (size) per State
fig.add_trace(go.Scatter(
    x=trump.state,
    y=trump['votes_Donald_Trump'],
    text=trump['Poverty'],
    marker=dict(color = 'red',size=trump['Poverty']/trump['TotalPop']*100),
    showlegend=True,
    mode='markers',
    name = 'Trump',
    opacity=0.7
))

# Adding Bidens Total Votes vs Poverty Percentage of Population (size) per State
fig.add_trace(go.Scatter(
    x=biden.state,
    y=biden['votes_Joe_Biden'],
    text=biden['Poverty'],
    marker=dict(color="blue",size=biden['Poverty']/biden['TotalPop']*100),
    showlegend=True,
    mode='markers',
    name = 'Biden',
    opacity=0.7
))

# Updating Title and axis names
fig.update_layout(title='Election total votes & Poverty Per US State',
                  xaxis_title="States",
                  yaxis_title="Total Votes",
                  legend_title="Candidates")

# Showing Final Figure
fig.show()
```

Figure shows:

- On average poverty rate in states voting for Biden are less than that of Trumps voting States.

[17]:
```
# Creating Scatter plot illustrating
fig = go.Figure()

# Defining Sets of Data for each candidate
trump = df[df.answer == 'Trump']
biden = df[df.answer == 'Biden']

# Adding Trumps Total Votes vs Income Percentage of Population (size) per State
fig.add_trace(go.Scatter(
```

```
    x=trump.state,
    y=trump['votes_Donald_Trump'],
    text=trump['Poverty'],
    marker=dict(color = 'red',size=trump['Income']/trump['TotalPop']*1000),
    showlegend=True,
    mode='markers',
    name = 'Trump',
    opacity=0.7
))

# Adding Bidens Total Votes vs Income Percentage of Population (size) per State
fig.add_trace(go.Scatter(
    x=biden.state,
    y=biden['votes_Joe_Biden'],
    text=biden['Poverty'],
    marker=dict(color="blue",size=biden['Income']/biden['TotalPop']*1000),
    showlegend=True,
    mode='markers',
    name = 'Biden',
    opacity=0.7
))

# Updating Title and axis names
fig.update_layout(title='Election total votes & Income Per US State',
                  xaxis_title="States",
                  yaxis_title="Total Votes",
                  legend_title="Candidates")

# Showing Final Figure
fig.show()
```

Figure shows:

- States Income Rates seems to have almost no impact on total votes for each candidate.

3.8 3D Geospacial Maps

Exploring relationships between most correlated data through 3 dimentional Maps, using json and PyDeck.

[18]:
```
# Defining Dictionaries
data = {}
geojson = {}

# Parsing DF to a dictionary
with open('df.csv', 'r') as f:
  reader = csv.DictReader(f);
  for row in reader:
```

```
    STUSPS = str(row['state'])
    IncomePerCap = float(row['IncomePerCap'])
    Income = float(row['Income'])
    cases = float(row['cases'])
    deaths = float(row['deaths'])
    TotalPop = float(row['TotalPop'])
    Men = float(row['Men'])
    Women = float(row['Women'])
    VotingAgeCitizen = float(row['VotingAgeCitizen'])
    Employed = float(row['Employed'])
    Hispanic = float(row['Hispanic'])
    White = float(row['White'])
    Black = float(row['Black'])
    Asian = float(row['Asian'])
    Pacific = float(row['Pacific'])
    Native = float(row['Native'])
    Poverty = float(row['Poverty'])
    Unemployment = float(row['Unemployment'])
    Professional = float(row['Professional'])
    Service = float(row['Service'])
    Office = float(row['Office'])
    Construction = float(row['Construction'])
    Production = float(row['Production'])
    FamilyWork = float(row['FamilyWork'])
    SelfEmployed = float(row['SelfEmployed'])
    PublicWork = float(row['PublicWork'])
    PrivateWork = float(row['PrivateWork'])
    total_votes = float(row['total_votes'])
    votes_Donald_Trump = float(row['votes_Donald_Trump'])
    votes_Joe_Biden = float(row['votes_Joe_Biden'])
    electoral_vote = float(row['electoral vote'])
    answer = str(row['answer'])
    polls_sample = float(row['polls_sample'])
    polls_biden = float(row['polls_biden'])
    polls_trump = float(row['polls_trump'])
    lat = float(row['lat'])
    long = float(row['long'])
    color = ast.literal_eval(row['color'])

# Parsing missing data to json
    if STUSPS not in data:
        data[STUSPS] = {}
    data[STUSPS] = {
      'state': STUSPS,
      'IncomePerCap': IncomePerCap,
      'Income': Income,
      'cases': cases,
```

```
      'deaths': deaths,
      'TotalPop': TotalPop,
      'Men': Men,
      'Women': Women,
      'VotingAgeCitizen': VotingAgeCitizen,
      'Employed': Employed,
      'Hispanic': Hispanic,
      'White': White,
      'Black': Black,
      'Asian': Asian,
      'Pacific': Pacific,
      'Native': Native,
      'Poverty': Poverty,
      'Professional': Professional,
      'Service': Service,
      'Office': Office,
      'Construction': Construction,
      'Production': Production,
      'FamilyWork': FamilyWork,
      'SelfEmployed': SelfEmployed,
      'PublicWork': PublicWork,
      'PrivateWork': PrivateWork,
      'total_votes': total_votes,
      'votes_Donald_Trump': votes_Donald_Trump,
      'votes_Joe_Biden': votes_Joe_Biden,
      'electoral vote': electoral_vote,
      'answer': answer,
      'polls_sample': polls_sample,
      'polls_biden': polls_biden,
      'polls_trump': polls_trump,
      'lat':lat,
      'long':long,
      'color':color
    }

# Adding Data to json Shape file
with open(r'Final Data\\shapes.json', 'r') as f:
    geojson = json.load(f)
    missing = []
    for feature in geojson['features']:
        featureProperties = feature['properties']
        if featureProperties['STUSPS'] in data:
            STUSPS = str(featureProperties['STUSPS'])
            featureData = data.get(STUSPS)
            for key in featureData.keys():
                featureProperties[key] = featureData[key]
        else:
```

```
            missing.append(featureProperties['STUSPS'])

#Save the augmented shapefile
with open('1.geojson', 'w') as f:
  json.dump(geojson, f)
```

Executing Maps from json file

[19]:
```
# HTML Legend Creation
legend_l = [{'text': 'Trump', 'color': [255, 20, 20]},{'text': 'Biden', 'color':
 ↪ [20, 138, 255]},{'text': 'Income', 'color': [230, 230, 230]}]
legend = create_legend(legend_l)


# Load in the JSON data
DATA_URL = r'Final Data\\1.geojson'
json = geojson

# Defining View State for PDK
view_state = pdk.ViewState(
    longitude=df.long[5],
    latitude=df.lat[5],
    zoom=3,
    min_zoom=3,
    max_zoom=4,
    pitch=45,
    bearing=0)

# Defining First Layer of PDK Map
Totalpop = pdk.Layer(
    'ColumnLayer',
    df,
    get_position=['long', 'lat'],
    get_elevation='TotalPop',
    auto_highlight=True,
    elevation_scale=0.02,
    pickable=True,
    elevation_range=[0, 10],
    extruded=True,
    coverage=5,
    get_fill_color=[216, 243, 212],
    radius=5000)


# Defining Second Layer of PDK Map
states = pdk.Layer(
    "GeoJsonLayer",
    json,
```

```
    opacity=0.5,
    stroked=False,
    filled=True,
    extruded=True,
    wireframe=True,
    get_elevation=0,
    get_fill_color="properties.color",
    get_line_color=[255, 255, 255],
)


# Defining Third Layer of PDK Map
Income = pdk.Layer(
    "ScatterplotLayer",
    df,
    opacity=0.4,
    stroked=True,
    filled=True,
    radius_scale=800,
    radius_min_pixels=1,
    radius_max_pixels=100,
    line_width_min_pixels=1,
    get_position=['long','lat'],
    get_radius="Income/80000",
    get_fill_color=[230, 230, 230],
    get_line_color=[0, 0, 0],
)

# Defining Tooltip Layer of PDK Map
tooltip = {"html": "<b>N Cases:</b> {cases} K <br /><b>N Deaths:</b> {deaths}␣
 ↪K"}



# Initializing Map PyDeck
r = pdk.Deck(
    [Totalpop,states,Income],
    initial_view_state=view_state,
    map_style=pdk.map_styles.LIGHT,
    tooltip=tooltip,
    mapbox_key='pk.
 ↪eyJ1Ijoib3Nvczk2IiwiYSI6ImNraXB4eWh4dTA4ZTgydG55d2UzOWE1MHgifQ.
 ↪_3Ib-ZEWbqLdmSQ6rR8K6Q'
)


# Displaying Title
```

```
display(HTML("""
   <strong>US Elections 2020 VS Income and Total population</strong>
   (Data from <a href="https://www.kaggle.com/etsc9287/
 ↪2020-general-election-polls">Kaggle</a>)
"""))


# Displaying Legend
display(legend)

# Saving HTML file
r.to_html("polygon_layer.html")
```

```
HTML(value='\n   <strong>US Elections 2020 VS Income and Total population</
 ↪strong>\n   (Data from <a href="htt…
```

```
HTML(value='\n    <style>\n      .legend {\n        width: 300px;\n      }\n    ␣
 ↪ .square {\n        height: 1…
```

[19]:
```
<IPython.core.display.HTML object>
```

Figure shows:

- Figure Shows states where Biden claims tend to have more income on average.

[20]:
```
# HTML Legend Creation
legend_l = [{'text': 'Trump', 'color': [255, 20, 20]},{'text': 'Biden', 'color':
 ↪ [20, 138, 255]},{'text': 'Poverty', 'color': [230, 230, 230]}]
legend = create_legend(legend_l)


# Load in the JSON data
DATA_URL = r'D:\DATASCIENCE\Project 6\Final Data\\1.geojson'
json = geojson


# Defining View State for PDK
view_state = pdk.ViewState(
    longitude=df.long[5],
    latitude=df.lat[5],
    zoom=3,
    min_zoom=3,
    max_zoom=4,
    pitch=45,
    bearing=0)


# Defining First Layer of PDK Map
```

```
Totalpop = pdk.Layer(
    'ColumnLayer',
    df,
    get_position=['long', 'lat'],
    get_elevation='Poverty',
    auto_highlight=True,
    elevation_scale=0.02,
    pickable=True,
    elevation_range=[0, 10],
    extruded=True,
    coverage=5,
    get_fill_color=[216, 243, 212],
    radius=5000)


# Defining Second Layer of PDK Map
states = pdk.Layer(
    "GeoJsonLayer",
    json,
    opacity=0.5,
    stroked=False,
    filled=True,
    extruded=True,
    wireframe=True,
    get_elevation=0,
    get_fill_color="properties.color",
    get_line_color=[255, 255, 255],
)


# Defining Third Layer of PDK Map
unemployment = pdk.Layer(
    "ScatterplotLayer",
    df,
    opacity=0.4,
    stroked=True,
    filled=True,
    radius_scale=800,
    radius_min_pixels=1,
    radius_max_pixels=100,
    line_width_min_pixels=1,
    get_position=['long','lat'],
    get_radius="Unemployment/10000",
    get_fill_color=[230, 230, 230],
    get_line_color=[0, 0, 0],
)
```

```
# Defining Tooltip Layer of PDK Map
tooltip = {"html": "<b>N Cases:</b> {cases} K <br /><b>N Deaths:</b> {deaths}␣
 ↪K"}



# Initializing Map PyDeck
r = pdk.Deck(
    [Totalpop,states,unemployment],
    initial_view_state=view_state,
    map_style=pdk.map_styles.LIGHT,
    tooltip=tooltip,
    mapbox_key='pk.
 ↪eyJ1Ijoib3Nvczk2IiwiYSI6ImNraXB4eWh4dTA4ZTgydG55d2UzOWE1MHgifQ.
 ↪_3Ib-ZEWbqLdmSQ6rR8K6Q'
)


# Displaying Title
display(HTML("""
   <strong>US Elections 2020 VS Unemployment and Poverty</strong>
   (Data from <a href="https://www.kaggle.com/etsc9287/
 ↪2020-general-election-polls">Kaggle</a>)
"""))

# Displaying Legend
display(legend)

# Saving HTML file
r.to_html("polygon_layer.html")
```

```
HTML(value='\n   <strong>US Elections 2020 VS Unemployment and Poverty</
 ↪strong>\n   (Data from <a href="https:…
```

```
HTML(value='\n    <style>\n      .legend {\n        width: 300px;\n      }\n    ␣
 ↪ .square {\n        height: 1…
```

[20]: <IPython.core.display.HTML object>

Figure shows:

- Figure Shows states with high unemployment rate seem to have more poverty.
- States with highest unemployment rate and poverty seem to vote for Biden.

3.9 Machine Learning Process

Exploring Machine Learning Pipeline to predict election winner based on current countries demographical statistics such as race, population, unemployment, poverty and sickness.

Yet these features are not inclusive of everything that factor into the selection process. Thus further analysis of historic data is required in a later stage due to unaccessable data.

[21]:
```
# Subsetting ML Dataset
machine_learning_df =␣
 ↪df[['state','total_votes','polls_sample','polls_biden','polls_trump','cases','deaths','Tota
 ↪vote']]

# Redefining Categorical Data
machine_learning_df = pd.get_dummies(machine_learning_df)

# Cleaning Data and setting Target column
data = machine_learning_df.drop(['answer_Biden'],axis=1)
data.rename(columns={'answer_Trump':'target'},inplace=True)

# Feature selection
features = data.drop('target', axis=1)

# Splitting Data into Training and Testing Data
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, data['target'], random_state=4)


# Defining Pipeline used

# Average CV score on the training set was: 0.975
pipe = make_pipeline(
    Normalizer(norm="max"),
    GradientBoostingClassifier(learning_rate=0.1, max_depth=7, max_features=0.
 ↪2, min_samples_leaf=8, min_samples_split=5, n_estimators=185, subsample=0.65)
)

# Fitting Data to the pipeline
pipe.fit(training_features, training_target)

# Appending Results to variable
results = pipe.predict(testing_features)
```

3.9.1 Testing ML Model

[ ]:
```
# Defining list of results
Percent = []

# Testing Model nth times
for i in range(10000):
    # Subsetting ML Dataset
```

```
    machine_learning_df =␣
↪df[['state','total_votes','polls_sample','polls_biden','polls_trump','cases','deaths','Tota
↪vote']]

    # Redefining Categorical Data
    machine_learning_df = pd.get_dummies(machine_learning_df)

    # Cleaning Data and setting Target column
    data = machine_learning_df.drop(['answer_Biden'],axis=1)
    data.rename(columns={'answer_Trump':'target'},inplace=True)

    # Feature selection
    features = data.drop('target', axis=1)

    # Splitting Data into Training and Testing Data
    training_features, testing_features, training_target, testing_target = \
                train_test_split(features, data['target'], random_state=4)


    # Defining Pipeline used

    # Average CV score on the training set was: 0.975
    pipe = make_pipeline(
        Normalizer(norm="max"),
        GradientBoostingClassifier(learning_rate=0.1, max_depth=7,␣
↪max_features=0.2, min_samples_leaf=8, min_samples_split=5, n_estimators=185,␣
↪subsample=0.65)
    )

    # Fitting Data to the pipeline
    pipe.fit(training_features, training_target)

    # Defining Test Data
    data = machine_learning_df.drop(['answer_Biden'],axis=1)
    data.rename(columns={'answer_Trump':'target'},inplace=True)

    datatest = data.drop('target',axis=1)

    # Predicting Data
    trump = pipe.predict(datatest)
    biden = 1-trump
    datatest['trump'] = trump
    datatest['biden'] = biden

    # Grouping Results
    answer = datatest[['trump','biden','electoral vote']].
↪groupby(['trump','biden']).sum()
```

```
    # Formating Data Output
    if answer.iloc[0]['electoral vote'] > answer.iloc[1]['electoral vote']:
        Percent.append(1)
    else:
        Percent.append(0)
# Printing Percentage Result
print(f"{sum(Percent)/100}%")
```

3.10 Conclusion

EDA Shows:-

- A strong positive correlation between almost every element which is to be expected.
- A strong positive correlation between covid-19 cases and votes for trump columns.
- A weak positive correlation between income and general votes.
- 80% of the States has avarage Income Below 41K with maximum of 65K, Which shows almost even destribution of income.
- 90% of the States has Poverty Counts Below 1.74M with maximum of 5.9M, Which shows that 10% of the states has 5 times more poverty than the rest of the states, Yet this could be an effect of total population.
- 90% of the States has Unemployment Counts Below 833K with maximum of 3.02M, Which shows that 10% of the states has 4 times more Unemployment than the rest of the states, Yet this could be an effect of total population.
- 90% of the States has Employment Counts Below 6.09M with maximum of 18M, Which shows that 10% of the states has 2 times more Employment than the rest of the states, Yet this could be an effect of total population.
- Gender distribution in most states are almost equal in count.
- Bidens Average total votes is slightly higher than Trumps with less than 100K in difference.
- Biden excels over Trump with huge difference in California.
- Bidens Average total Electoral votes is slightly higher than Trumps with less than 5 electoral votes in difference.
- Biden excels over Trump with California 55 Electoral votes.
- Again Bidens CA winning is the main feature of this graph.
- Yet Trumps manages to win TX and FL which are the second most awarding states with electoral votes.
- Figure shows a general advantage to Bidens Point sizes indicating more electoral votes on average.
- States who voted for Biden seem to have less COVID-19 Cases than those who voted for Trump.

- Does this indicate more educated states voted for Biden? further analysis is required in this area.
- States Unemployment Rates seems to have almost no impact on total votes for each candidate.
- On average poverty rate in states voting for Biden are less than that of Trumps voting States.
- States Income Rates seems to have almost no impact on total votes for each candidate.
- Figure Shows states where Biden claims tend to have more income on average.
- Figure Shows states with high unemployment rate seem to have more poverty.
- States with highest unemployment rate and poverty seem to vote for Biden.

Final Thoughts:- Further Data gathering is required to reach a solid conclusion. Historical data of past elections is needed yet unaccessable due to insufficient demographic data of this time. Behavioral science and input is also required to further understand the inclination of the demographic public.

Simple analysis done seems to suggest that Biden won as a result of bad management as a result of low income and other factors such as poverty and unemplyment.