Group: Tres Analytics

US 2020'S ELECTION ANALYSIS

Investigating Significant Factors Influencing Voter Preferences in the 2020 Election

TABLE OF CONTENTS

01 *2020 Election Background*

02 *The US Statistics*

03 *Predicting Voters' Preferences*

04 *Conclusions*

01. 2020 ELECTION BACKGROUND

The US presidential election is a pivotal event held every four years where citizens vote to elect the President and Vice President of the United States.
This election shapes the nation's leadership and policy direction for the next term.
Candidates typically represent major political parties, such as the Democratic and Republican parties, though independent and third-party candidates also participate.
Campaigns focus on key issues like the economy, healthcare, foreign policy, and social justice, with debates and rallies helping voters make informed decisions.
The election's outcome is determined through the Electoral College system, which assigns votes to states based on their population.

FACTORS WHICH COULD INFLUENCE VOTER PREFERENCES

01 Economy

Voters may prioritize economic stability, job creation, and policies that support financial growth, such as tax cuts or government spending.

02 Healthcare

Access to affordable healthcare and policies on insurance, public health, and medical costs often shape voting decisions, particularly in times of crisis or reform.

03 Foreign Policy

Voters might be influenced by a candidate's stance on international relations, trade agreements, military interventions, and global cooperation.

04 Social Justice

Issues like racial equality, gender rights, and social equity can significantly impact voter preferences, especially for those advocating for systemic change.

THE OBJECTIVE

Our objective is to investigate the *significant factors* influencing Voter Preferences in the 2020 Election

1. *Did the US economic condition affect Voter Preferences?*
2. *Did the US Healthcare Policy affect the Voter Preferences?*
3. *Did the US Foreign Policy affect Voter Preferences?*
4. *Did the US Social Policy affect the Voters Preferences?*
5. *What are the other factors that significantly influence Voters Preference?*

02. THE US STATISTICS

THE VOTERS - DEMOGRAPHIC DATASET

This is the dataset sourced from ANNES and cleaning process we conducted to describe voters' demographic in 2020 US-election. This cleaning process including: re-indexing, transposing the data, and cleaning unnecessary characters

```
import pandas as pd
import numpy as np

voters=pd.read_csv("Voters_demographics_2019_USCB.csv")
voters.columns = voters.columns.str.strip()
voters.columns = voters.columns.str.encode('ascii', 'ignore').str.decode('ascii')
```

```
voters.rename(columns={'Label (Grouping)': 'Region'}, inplace=True)
voters = voters.T
voters.head()
```

	0	1	2	3	4	5
Region	Citizens 18 years and over	AGE	18 to 29 years	30 to 44 years	45 to 64 years	65 years and over
United States!!Total!!Estimate	235,418,734	NaN	49,554,247	56,305,720	77,246,978	52,311,789
United States!!Percent!!Estimate	(X)	NaN	21.0%	23.9%	32.8%	22.2%
Alabama!!Total!!Estimate	3,731,336	NaN	763,649	875,096	1,243,536	849,055
Alabama!!Percent!!Estimate	(X)	NaN	20.5%	23.5%	33.3%	22.8%

5 rows × 35 columns

```
[194] voters.index = voters.index.str.replace(r'!!Total!!Estimate', '', regex=True)
```

```
[195] voters= voters[~voters.index.str.contains('!!Percent!!Estimate')]
```

```
voters.reset_index(drop=True, inplace=True)
voters.columns = ['Region'] + list(voters.iloc[0, 1:]
voters = voters.drop(index=0)
```

```
voters.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 1 to 104
Data columns (total 35 columns):
 #   Column                                                                                  Non-Null Count  Dtype
---  ------                                                                                  --------------  -----
 0   Region                                                                                  104 non-null    object
 1       AGE                                                                                 0 non-null      object
 2           18 to 29 years                                                                  104 non-null    object
 3           30 to 44 years                                                                  104 non-null    object
 4           45 to 64 years                                                                  104 non-null    object
 5           65 years and over                                                               104 non-null    object
 6       SEX                                                                                 0 non-null      object
 7           Male                                                                            104 non-null    object
 8           Female                                                                          104 non-null    object
 9       RACE AND HISPANIC ORIGIN                                                            0 non-null      object
 10          White alone                                                                     104 non-null    object
 11          Black or African American alone                                                 104 non-null    object
 12          Asian alone                                                                     104 non-null    object
 13          American Indian and Alaska Native alone                                         104 non-null    object
 14          Native Hawaiian and Other Pacific Islander alone                                104 non-null    object
 15          Some Other Race alone                                                           104 non-null    object
 16          Two or More Races                                                               104 non-null    object
 17          Hispanic or Latino                                                              104 non-null    object
 18          White alone, Not Hispanic or Latino                                             104 non-null    object
 19      EDUCATIONAL ATTAINMENT                                                              0 non-null      object
 20          Less than 9th grade                                                             104 non-null    object
 21          9th to 12th grade, no diploma                                                   104 non-null    object
 22          High school graduate (includes equivalency)                                     104 non-null    object
 23          Some college, no degree                                                         104 non-null    object
 24          Associate's degree                                                              104 non-null    object
 25          Bachelor's degree                                                               104 non-null    object
 26          Graduate or professional degree                                                 104 non-null    object
 27          High school degree or higher                                                    104 non-null    object
 28          Bachelor's degree or higher                                                     104 non-null    object
 29      POVERTY STATUS                                                                      0 non-null      object
 30          Total Citizens, 18 years and older, for whom poverty status is determined       104 non-null    object
 31          Income in the past 12 months below poverty level                                104 non-null    object
 32          Income in the past 12 months at or above the poverty level                      104 non-null    object
 33      HOUSEHOLD INCOME (IN 2019 INFLATION-ADJUSTED DOLLARS)                               0 non-null      object
 34          Median Household Income for Households with a Citizen, Voting-Age Householder  104 non-null    object
dtypes: object(35)
memory usage: 28.6+ KB
```

THE VOTERS - DEMOGRAPHIC DATASET

In this part, we set the data with multi-index level to give a better perspective.

```
outer_index = [
    ("Citizens 18 years and over", ""),
    ("AGE", ""),("AGE", "18 to 29 years"),
    ("AGE", "30 to 44 years"),("AGE", "45 to 64 years"),
    ("AGE", "65 years and over"),
    ("SEX", "SEX"),("SEX", "Male"), ("SEX", "Female"),
    ("RACE AND HISPANIC ORIGIN", ""),
    ("RACE AND HISPANIC ORIGIN", "White alone"),
    ("RACE AND HISPANIC ORIGIN", "Black or African American alone"),
    ("RACE AND HISPANIC ORIGIN", "Asian alone"),
    ("RACE AND HISPANIC ORIGIN", "American Indian and Alaska Native alone"),
    ("RACE AND HISPANIC ORIGIN", "Native Hawaiian and Other Pacific Islander alone"),
    ("RACE AND HISPANIC ORIGIN", "Some Other Race alone"),
    ("RACE AND HISPANIC ORIGIN", "Two or More Races"),
    ("RACE AND HISPANIC ORIGIN", "Hispanic or Latino"),
    ("RACE AND HISPANIC ORIGIN", "White alone, Not Hispanic or Latino"),
    ("EDUCATIONAL ATTAINMENT", ""),
    ("EDUCATIONAL ATTAINMENT", "Less than 9th grade"),
    ("EDUCATIONAL ATTAINMENT", "9th to 12th grade, no diploma"),
    ("EDUCATIONAL ATTAINMENT", "High school graduate (includes equivalency)"),
    ("EDUCATIONAL ATTAINMENT", "Some college, no degree"),
    ("EDUCATIONAL ATTAINMENT", "Associate's degree"),
    ("EDUCATIONAL ATTAINMENT", "Bachelor's degree"),
    ("EDUCATIONAL ATTAINMENT", "Graduate or professional degree"),
    ("EDUCATIONAL ATTAINMENT", "High school degree or higher"),
    ("EDUCATIONAL ATTAINMENT", "Bachelor's degree or higher"),
    ("POVERTY STATUS", ""),
    ("POVERTY STATUS", "Total Citizens, 18 years and older, for whom poverty status is determined"),
    ("POVERTY STATUS", "Income in the past 12 months below poverty level"),
    ("POVERTY STATUS", "Income in the past 12 months at or above the poverty level"),
    ("HOUSEHOLD INCOME (IN 2019 INFLATION-ADJUSTED DOLLARS)", ""),
    ("HOUSEHOLD INCOME (IN 2019 INFLATION-ADJUSTED DOLLARS)",
     "Median Household Income for Households with a Citizen, Voting-Age Householder")
]

multi_index = pd.MultiIndex.from_tuples(outer_index, names=["Category", "Subcategory"])

voters.columns = multi_index
```

In this part, we created a new dataset (df) for our study and drop some unwanted columns.

```
df = voters.copy()
df = df.drop(df.columns[[1, 6, 9, 19, 29]], axis=1)
```

```
df = df.drop(df.columns[28], axis=1)
```

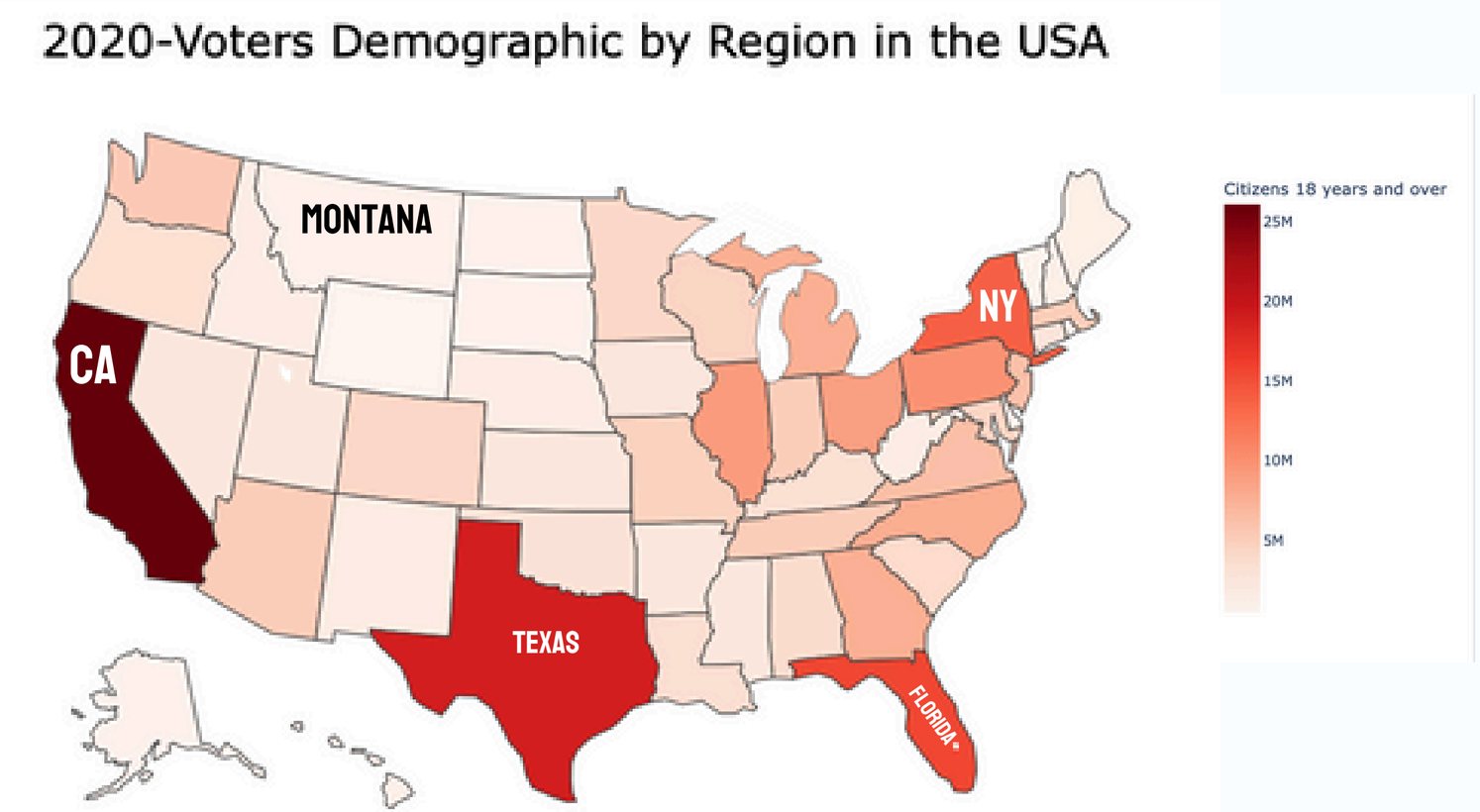
```
# Clean column names by stripping leading/trailing whitespaces
df.columns = pd.MultiIndex.from_tuples([(col[0].strip(), col[1].strip()) for col in df.columns])

# Remove commas or other non-numeric characters and convert to numeric
df = df.applymap(lambda x: pd.to_numeric(x.replace(',', ''), errors='coerce') if isinstance(x, str) else x)
# Check the result
df.dtypes
```

The info of the data after cleaning process:

		0
Citizens 18 years and over		int64
AGE	18 to 29 years	int64
	30 to 44 years	int64
	45 to 64 years	int64
	65 years and over	int64
SEX	Male	int64
	Female	int64
RACE AND HISPANIC ORIGIN	White alone	int64
	Black or African American alone	float64
	Asian alone	float64
	American Indian and Alaska Native alone	float64
	Native Hawaiian and Other Pacific Islander alone	float64
	Some Other Race alone	float64
	Two or More Races	float64
	Hispanic or Latino	int64
	White alone, Not Hispanic or Latino	int64
EDUCATIONAL ATTAINMENT	Less than 9th grade	int64
	9th to 12th grade, no diploma	int64
	High school graduate (includes equivalency)	int64
	Some college, no degree	int64
	Associate's degree	int64
	Bachelor's degree	int64
POVERTY STATUS	Total Citizens, 18 years and older, for whom poverty status is determined	int64
	Income in the past 12 months below poverty level	int64
	Income in the past 12 months at or above the poverty level	int64
HOUSEHOLD INCOME (IN 2019 INFLATION-ADJUSTED DOLLARS)	Median Household Income for Households with a Citizen, Voting-Age Householder	int64

dtype: object

THE VOTERS - DEMOGRAPHIC




```python
import plotly.express as px

fig = px.choropleth(
    df_filtered,
    locations='code_state',  # Use the state abbreviations
    locationmode='USA-states',  # Match US states
    color='Citizens 18 years and over',  # Color by population
    scope='usa',  # Focus on the USA
    color_continuous_scale='reds',  # Use a red-to-orange color map
)

fig.update_layout(
    title={
        'text': "2020-Voters Demographic by Region in the USA",
        'font': {
            'size': 24,  # Adjust the font size as needed
            'color': 'black'  # Set the title color to black
        },
        'x': 0.45,  # Center the title
        'xanchor': 'center'  # Anchor the title at the center
    }
)

fig.show()
```

```python
state_mapping = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA',
    'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'District of Columbia': 'DC',
    'Florida': 'FL', 'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL',
    'Indiana': 'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA',
    'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI',
    'Minnesota': 'MN', 'Mississippi': 'MS', 'Missouri': 'MO', 'Montana': 'MT',
    'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH', 'New Jersey': 'NJ',
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND',
    'Ohio': 'OH', 'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI',
    'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX',
    'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA',
    'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'
}

df['code_state'] = df.index.map(state_mapping)
```

- **We assigned a Region Index to the dataset by mapping the code_state column to illustrate voter distribution.**
- **The data visualization highlights population density in each US region, using darker map colors to represent a higher number of citizens aged 18 and over (eligible voters).**
- **The visualization reveals that California had the highest number of eligible voters, while Montana had the lowest.**

THE VOTERS - DEMOGRAPHIC DATASET

This is the process how we plot the data into the graphs

```python
import matplotlib.pyplot as plt

# Extract SEX data for United States
sex_data = df.loc["United States", [("SEX", "Male"), ("SEX", "Female")]]

# Rename columns
sex_data.index = ["Male", "Female"]

# Convert population to millions for y-axis
sex_data_millions = sex_data / 1_000_000

# Plot the data
plt.figure(figsize=(4, 4))
sex_data_millions.plot(kind="bar", color=["skyblue", "pink"], edgecolor="black")

# Add labels and title
plt.ylabel("Population (in millions)")
plt.title("Population by Sex in the United States")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="-", alpha=0.7)

# Format y-axis with no decimals
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{int(x)}"))

# Show the plot
plt.tight_layout()
plt.show()
```

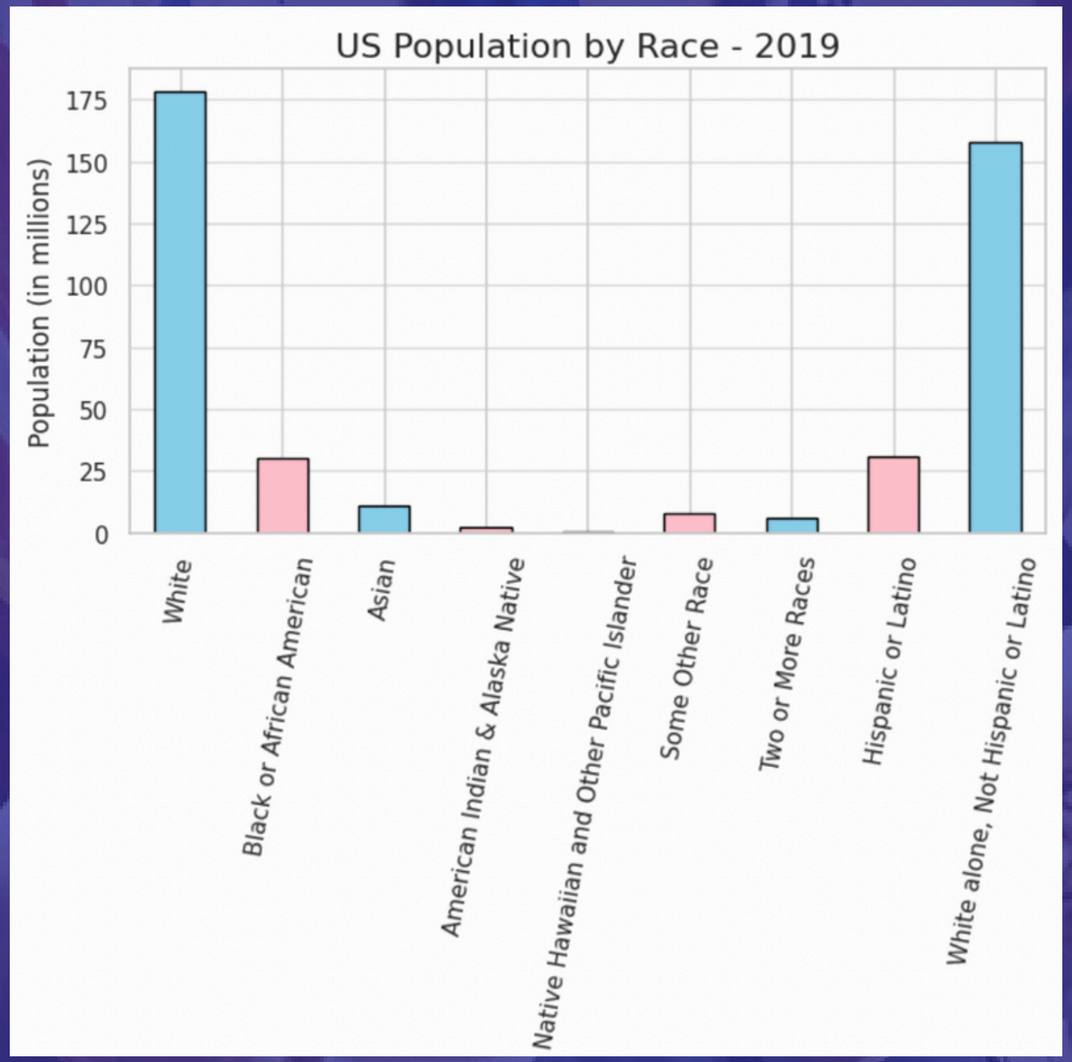
```python
import matplotlib.pyplot as plt

age_data = df.loc["United States", [("AGE", "18 to 29 years"),
 ("AGE", "30 to 44 years"),
  ("AGE", "45 to 64 years"),
  ("AGE", "65 years and over")
]]

age_data.index = ["18 to 29 years",
                  "30 to 44 years",
                 "45 to 64 years", "65 years and over"]
age_data_millions = age_data / 1_000_000

plt.figure(figsize=(5, 5))
age_data_millions.plot(kind="bar", color=["skyblue", "pink"], edgecolor="black")

plt.ylabel("Population (in millions)", fontsize=12)
plt.title("US Population by Age - 2019", fontsize=14)
plt.xticks(rotation=-75)
plt.grid(axis="y", linestyle="--", alpha=0.9)

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{int(x)}"))

plt.tight_layout()
plt.show()
```

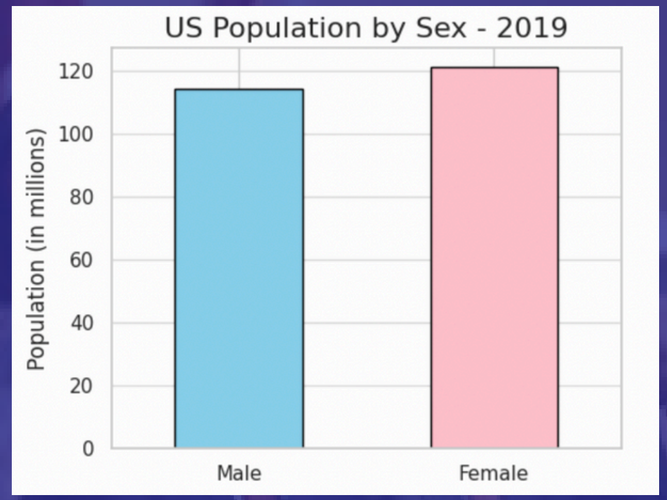
```python
import matplotlib.pyplot as plt

race_data = df.loc["United States", [("RACE AND HISPANIC ORIGIN", "White alone"),
 ("RACE AND HISPANIC ORIGIN", "Black or African American alone"),
  ("RACE AND HISPANIC ORIGIN", "Asian alone"),
   ("RACE AND HISPANIC ORIGIN", "American Indian and Alaska Native alone"),
    ("RACE AND HISPANIC ORIGIN", "Native Hawaiian and Other Pacific Islander alone"),
     ("RACE AND HISPANIC ORIGIN", "Some Other Race alone"),
      ("RACE AND HISPANIC ORIGIN", "Two or More Races"),
       ("RACE AND HISPANIC ORIGIN", "Hispanic or Latino"),
        ("RACE AND HISPANIC ORIGIN", "White alone, Not Hispanic or Latino")
]]

race_data.index = ["White",
                   "Black or African American",
                   "Asian",
                   "American Indian & Alaska Native",
                   "Native Hawaiian and Other Pacific Islander",
                   "Some Other Race",
                   "Two or More Races",
                   "Hispanic or Latino",
                   "White alone, Not Hispanic or Latino"]

race_data_millions = race_data / 1_000_000

plt.figure(figsize=(5, 7))
race_data_millions.plot(kind="bar", color=["skyblue", "pink"], edgecolor="black")

plt.ylabel("Population (in millions)")
plt.title("Population by Race in the United States")
plt.xticks(rotation=80)
plt.grid(axis="y", linestyle="-", alpha=0.7)

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{int(x)}"))

plt.tight_layout()
plt.show()
```

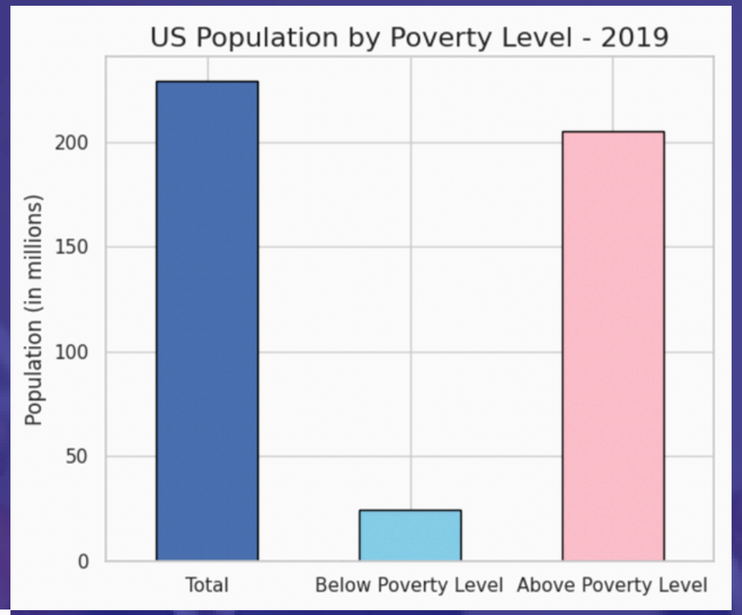
```python
import matplotlib.pyplot as plt

edu_data = df.loc["United States", [("EDUCATIONAL ATTAINMENT", "Less than 9th grade"),
 ("EDUCATIONAL ATTAINMENT", "9th to 12th grade, no diploma"),
  ("EDUCATIONAL ATTAINMENT", "High school graduate (includes equivalency)"),
  ("EDUCATIONAL ATTAINMENT", "Some college, no degree"),
  ("EDUCATIONAL ATTAINMENT", "Associate's degree"),
  ("EDUCATIONAL ATTAINMENT", "Bachelor's degree"),
  ("EDUCATIONAL ATTAINMENT", "Graduate or professional degree"),
  ("EDUCATIONAL ATTAINMENT", "High school degree or higher"),
  ("EDUCATIONAL ATTAINMENT", "Bachelor's degree or higher")
]]

edu_data.index = ["Less than 9th grade",
                  "9th to 12th grade, no diploma",
                  "High school graduate (includes equivalency)",
                 "Some college, no degree",
                  "Associate's degree",
                  "Bachelor's degree",
                  "Graduate or professional degree",
                  "High school degree or higher",
                  "Bachelor's degree or higher"]

edu_data_millions = edu_data / 1_000_000

plt.figure(figsize=(6, 4))
edu_data_millions.plot(kind="bar", color=["skyblue", "pink"], edgecolor="black")

plt.ylabel("Population (in millions)")
plt.title("Population by Education in the United States")
plt.xticks(rotation=85)
plt.grid(axis="y", linestyle="-", alpha=0.9)

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{int(x)}"))

plt.tight_layout()
plt.show()
```

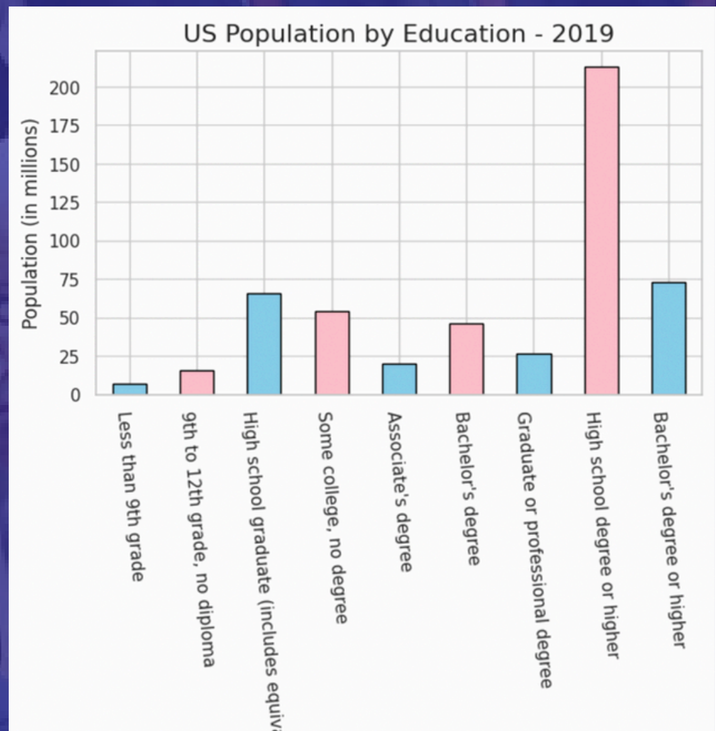
```python
pov_data = df.loc["United States", [("POVERTY STATUS", "Total Citizens, 18 years and older, for whom poverty status is determined"),
 ("POVERTY STATUS", "Income in the past 12 months below poverty level"),
  ("POVERTY STATUS", "Income in the past 12 months at or above the poverty level")
]]

pov_data.index = ["Total", "Below Poverty Level", "Above Poverty Level"]
pov_data_millions = pov_data / 1_000_000

plt.figure(figsize=(5, 5))
pov_data_millions.plot(kind="bar", color=["skyblue"], edgecolor="black")

plt.ylabel("Population (in millions)")
plt.title("Population by Poverty Level in the United States")
plt.xticks(rotation=-75)
plt.grid(axis="y", linestyle="-", alpha=0.9)

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f"{int(x)}"))

plt.tight_layout()
plt.show()
```

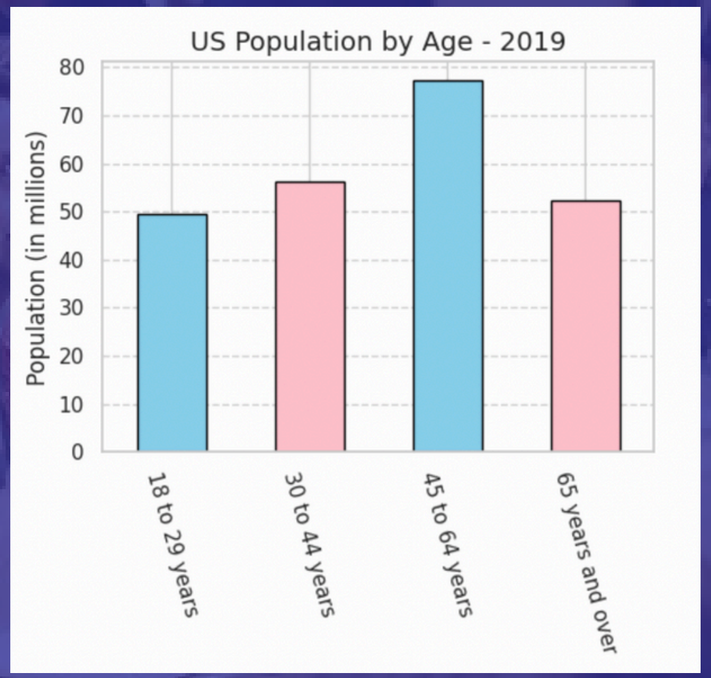
THE VOTERS - DEMOGRAPHIC











THE US ECONOMY DATASET - GDP

```
gdp = pd.read_excel('/content/gdp_per_state.xlsx',
                    sheet_name='gdp_dollars', skiprows=4)
new_column_names = {
    gdp.columns[0]: "Region",
    gdp.columns[1]: "Q1_2019",
    gdp.columns[2]: "Q2_2019",
    gdp.columns[3]: "Q3_2019",
    gdp.columns[4]: "Q4_2019",
    gdp.columns[5]: "Q1_2020",
    gdp.columns[6]: "Q2_2020",
    gdp.columns[7]: "Q3_2020",
    gdp.columns[8]: "Q4_2020",
}

gdp.rename(columns=new_column_names, inplace=True)
gdp.head()
```

	Region	Q1_2019	Q2_2019	Q3_2019	Q4_2019	Q1_2020	Q2_2020	Q3_2020	Q4_2020
0	United States	21115309.0	21329877.0	21540325.0	21747394.0	21561139.0	19520114.0	21170252.0	21494731.0
1	New England	1123191.0	1130963.0	1140932.0	1147568.0	1141524.0	1035902.0	1121716.0	1137921.0
2	Connecticut	285446.0	286646.0	288493.0	290703.0	288368.0	262708.0	283601.0	288923.0
3	Maine	66662.0	67053.0	68168.0	68985.0	68319.0	61496.0	67129.0	67839.0
4	Massachusetts	588308.0	594310.0	600545.0	603210.0	600740.0	546546.0	590307.0	598562.0

```
selected_gdp = gdp[["Region", "Q4_2019", "Q2_2020"]]
#Drop Unnecessary Rows
selected_gdp = selected_gdp.drop(selected_gdp.index[60:])
selected_gdp.tail()
```

	Region	Q4_2019	Q2_2020
55	California	3205000.0	2893054.0
56	Hawaii	97001.0	83429.0
57	Nevada	181752.0	156421.0
58	Oregon	258693.0	233799.0
59	Washington	624861.0	579695.0

```
state_mapping = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA',
    'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'District of Columbia': 'DC',
    'Florida': 'FL', 'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL',
    'Indiana': 'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA',
    'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI',
    'Minnesota': 'MN', 'Mississippi': 'MS', 'Missouri': 'MO', 'Montana': 'MT',
    'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH', 'New Jersey': 'NJ',
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND',
    'Ohio': 'OH', 'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI',
    'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX',
    'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA',
    'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'
}
selected_gdp['Region'] = selected_gdp['Region'].str.title()
selected_gdp['Region'] = selected_gdp['Region'].str.strip()
selected_gdp['code_state'] = selected_gdp['Region'].map(state_mapping)

missing_regions = selected_gdp[selected_gdp['code_state'].isna()]
selected_gdp = selected_gdp[selected_gdp['Region'].isin(state_mapping.keys())]
selected_gdp = selected_gdp.sort_values(by='Q2_2020', ascending=False)

selected_gdp
```

	Region	Q4_2019	Q2_2020	code_state
55	California	3205000.0	2893054.0	CA
46	Texas	1861582.0	1628185.0	TX
13	New York	1791567.0	1587879.0	NY
32	Florida	1126510.0	1026676.0	FL

This is a GDP dataset sourced from the U.S. Bureau of Economic Analysis.
We performed data cleaning by renaming the index and removing unnecessary rows.
Additionally, we added the state_code column to facilitate visualization on map graphs.

THE US GDP BY REGION IN Q2-2020

	Region	Q4_2019	Q2_2020	code_state
1	California	3205000.0	2893054.0	CA
2	Texas	1861582.0	1628185.0	TX
3	New York	1791567.0	1587879.0	NY
4	Florida	1126510.0	1026676.0	FL
5	Illinois	893356.0	807383.0	IL
6	Pennsylvania	818449.0	723830.0	PA
7	Ohio	703369.0	626275.0	OH
8	Georgia	634138.0	580732.0	GA
9	Washington	624861.0	579695.0	WA
10	New Jersey	642968.0	574018.0	NJ
11	North Carolina	601788.0	546776.0	NC
12	Massachusetts	603210.0	546546.0	MA
13	Virginia	566529.0	519912.0	VA
14	Michigan	543489.0	475494.0	MI
15	Maryland	432998.0	398609.0	MD
16	Colorado	400863.0	365009.0	CO
17	Arizona	379019.0	350141.0	AZ
18	Minnesota	389504.0	348034.0	MN
19	Indiana	384872.0	344033.0	IN
20	Tennessee	380823.0	333194.0	TN

	Region	Q4_2019	Q2_2020	code_state
21	Wisconsin	353935.0	314027.0	WI
22	Missouri	332660.0	299131.0	MO
23	Connecticut	290703.0	262708.0	CT
24	Oregon	258693.0	233799.0	OR
25	South Carolina	251665.0	224689.0	SC
26	Louisiana	259079.0	224129.0	LA
27	Alabama	230750.0	209852.0	AL
28	Kentucky	218426.0	193878.0	KY
29	Utah	196639.0	182895.0	UT
30	Iowa	196247.0	179037.0	IA
31	Oklahoma	201604.0	173061.0	OK
32	Kansas	178605.0	161464.0	KS
33	Nevada	181752.0	156421.0	NV
34	Arkansas	132596.0	120812.0	AR
35	Nebraska	133201.0	119692.0	NE
36	Mississippi	117642.0	105681.0	MS
37	New Mexico	106914.0	93367.0	NM
38	Hawaii	97001.0	83429.0	HI
39	New Hampshire	88015.0	78691.0	NH
40	Idaho	85791.0	77384.0	ID

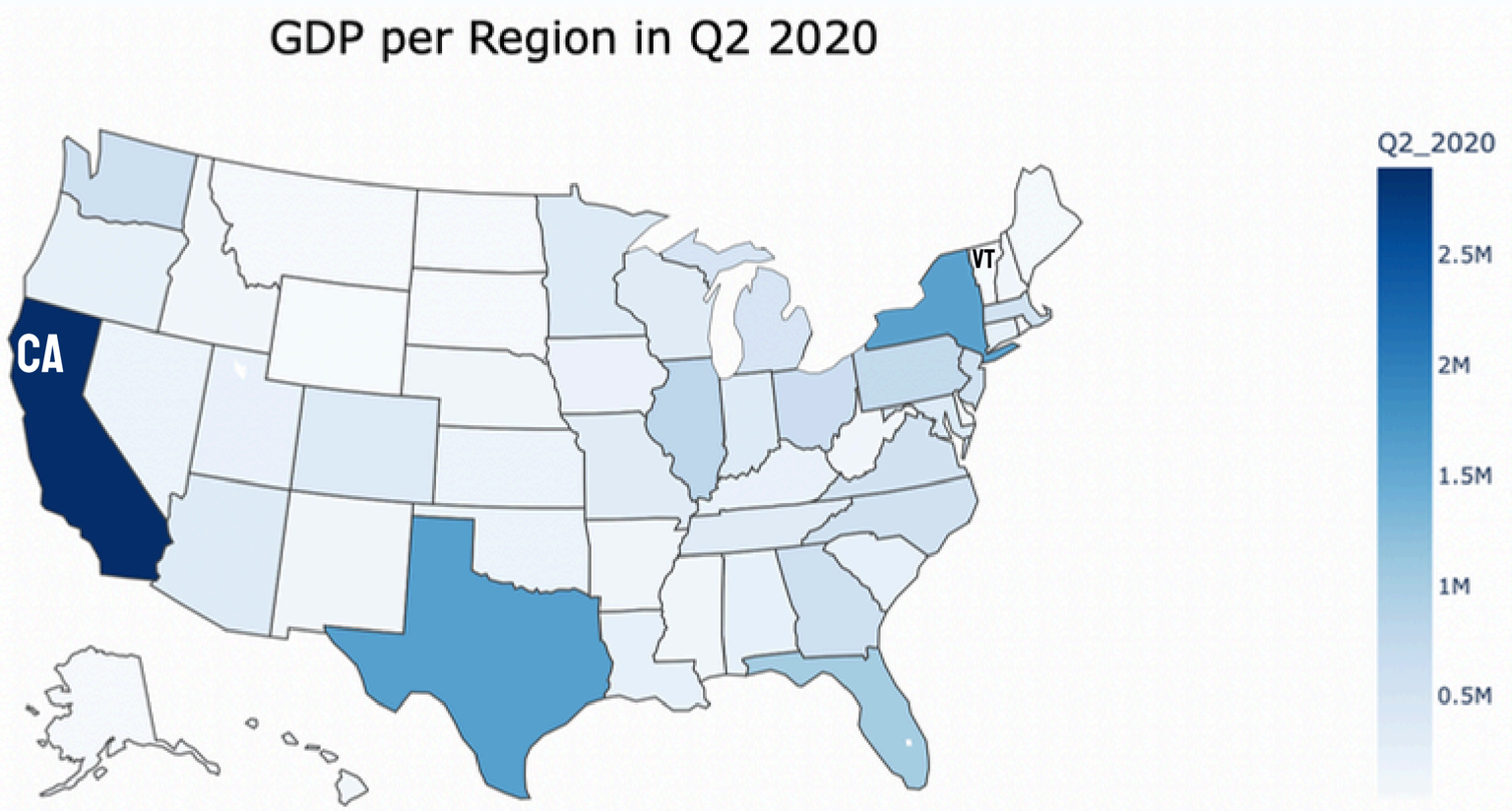
	Region	Q4_2019	Q2_2020	code_state
41	Delaware	77879.0	71611.0	DE
42	West Virginia	78480.0	68001.0	WV
43	Maine	68985.0	61496.0	ME
44	Rhode Island	62335.0	56285.0	RI
45	South Dakota	56052.0	50951.0	SD
46	North Dakota	57472.0	49881.0	ND
47	Montana	54035.0	47721.0	MT
48	Alaska	54675.0	45644.0	AK
49	Wyoming	40764.0	33233.0	WY
50	Vermont	34320.0	30175.0	VT

From the Data above, California had the biggest GDP among the US region and Vermont had the lowest.

THE US ECONOMY OVERVIEW IN 2020

In this step, we process the data and transform it for visualization on map distributions.

```
import plotly.express as px

fig = px.choropleth(
    selected_gdp,
    locations='code_state',  # Use the state abbreviations
    locationmode='USA-states',  # Match US states
    color='Q2_2020',
    scope='usa',
    color_continuous_scale= 'blues',
    )

fig.update_layout(
    title={
        'text': "GDP per Region in Q2 2020",
        'font': {
            'size': 24,
            'color': 'black'
        },
        'x': 0.5,  # Center the title
        'xanchor': 'center'
    }
)
fig.show()
```



- **We illustrate the distribution of GDP in the U.S. Region.**
- **California dominates the GDP amount in the U.S. followed by few countries.**
- **Vermont has the lowest GDP among other regions.**

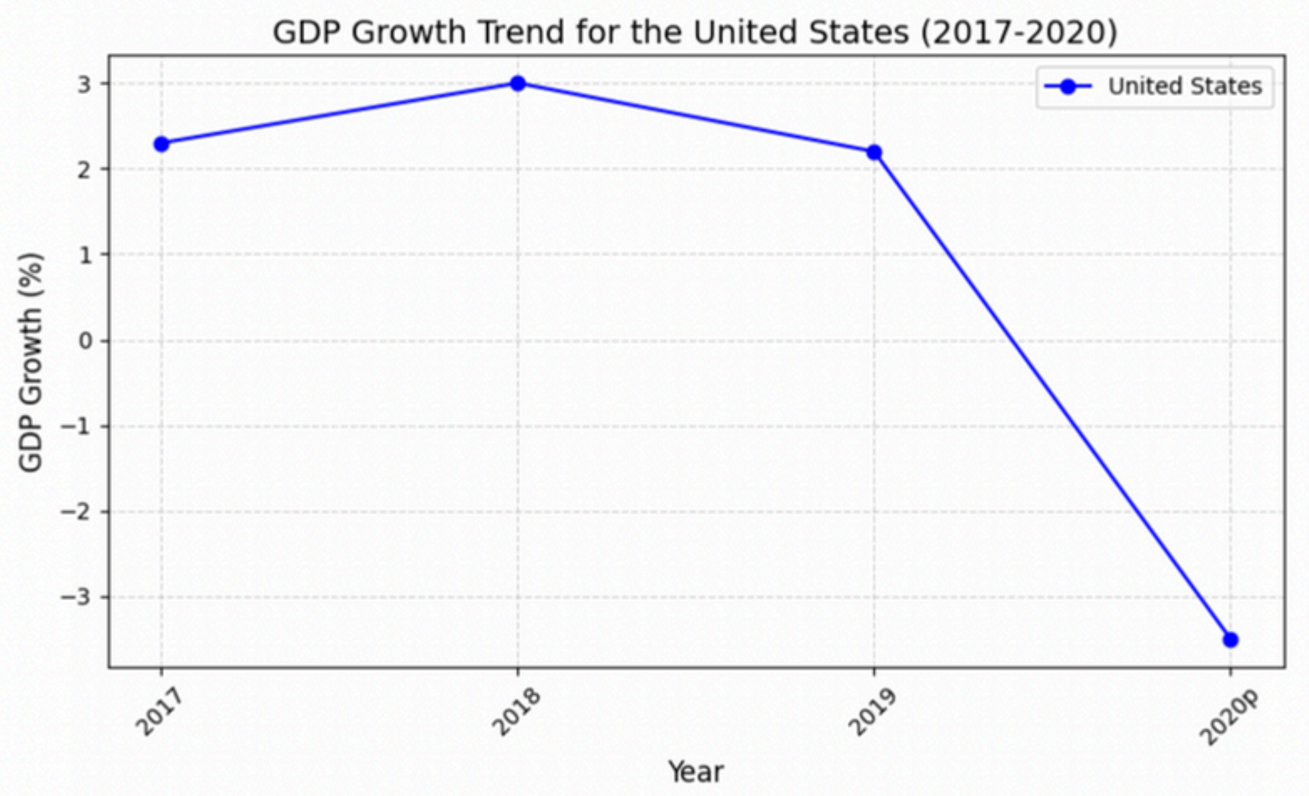
US GDP IN Q2- 2020

In this step, we use another worksheet (Table 4) to analyze the trend of U.S. GDP

```
gdp_grwth = pd.read_excel('/content/gdp_per_state.xlsx',
                          sheet_name='Table 4', skiprows=1)
gdp_grwth = gdp_grwth.drop(gdp_grwth.index[0])
gdp_grwth.rename(columns={'Unnamed: 0': 'Region'}, inplace=True)


gdp_grwth.head()
```

	Region	2017	2018	2019	2020p	Rank 2020
1	United States	2.3	3.0	2.2	-3.5	.....
2	New England	1.7	2.3	1.8	-4.0	.....
3	Connecticut	0.9	0.4	0.9	-4.1	34
4	Maine	1.6	2.4	2.6	-4.1	35
5	Massachusetts	2.4	3.4	2.4	-3.8	28

```
import matplotlib.pyplot as plt

# Extract United States data
us_data = gdp_grwth.iloc[0, 1:5]

# Plot the data
plt.figure(figsize=(8, 5))
us_data.plot(kind='line', marker='o', color='blue', label='United States')

# Add labels, title, and grid
plt.title('GDP Growth Trend for the United States (2017-2020)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('GDP Growth (%)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.xticks(range(len(us_data.index)), us_data.index, rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```

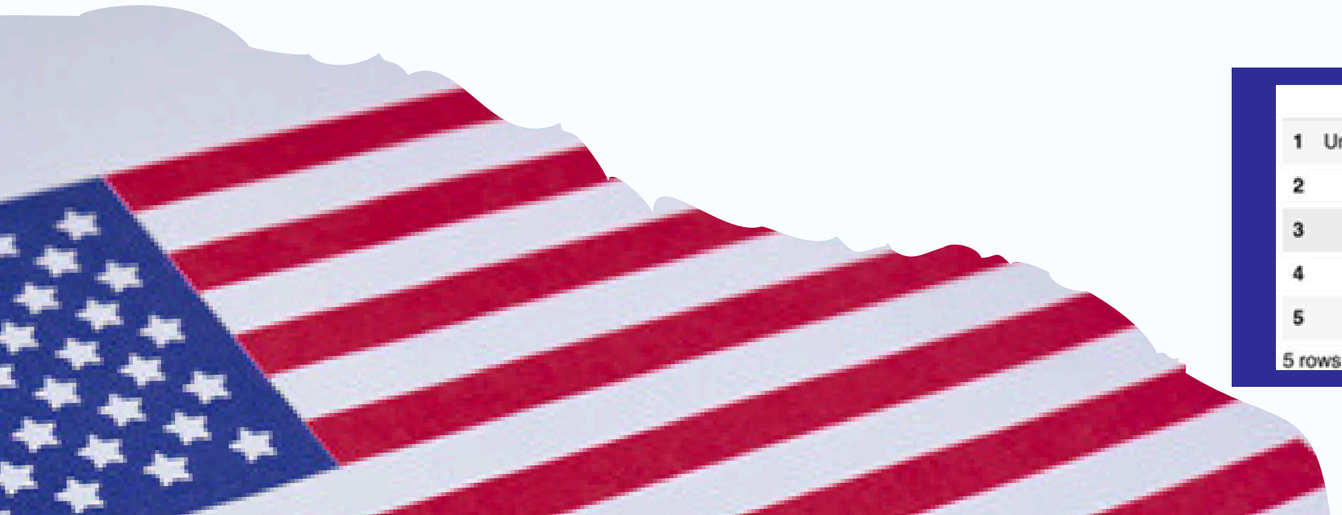


The graph shows that after considerable growth from 2017 to 2018, the growth of U.S. GDP began to decline in 2019, followed by a sharp drop in 2020.

THE US ECONOMY DATASET
UNEMPLOYMENT RATE

- **This is a Unemployment Rate dataset sourced from the U.S. Bureau of Labor Statistics.**

- **We performed data cleaning by assigning suitable index, removing unnecessary rows, and modifying the data types.**



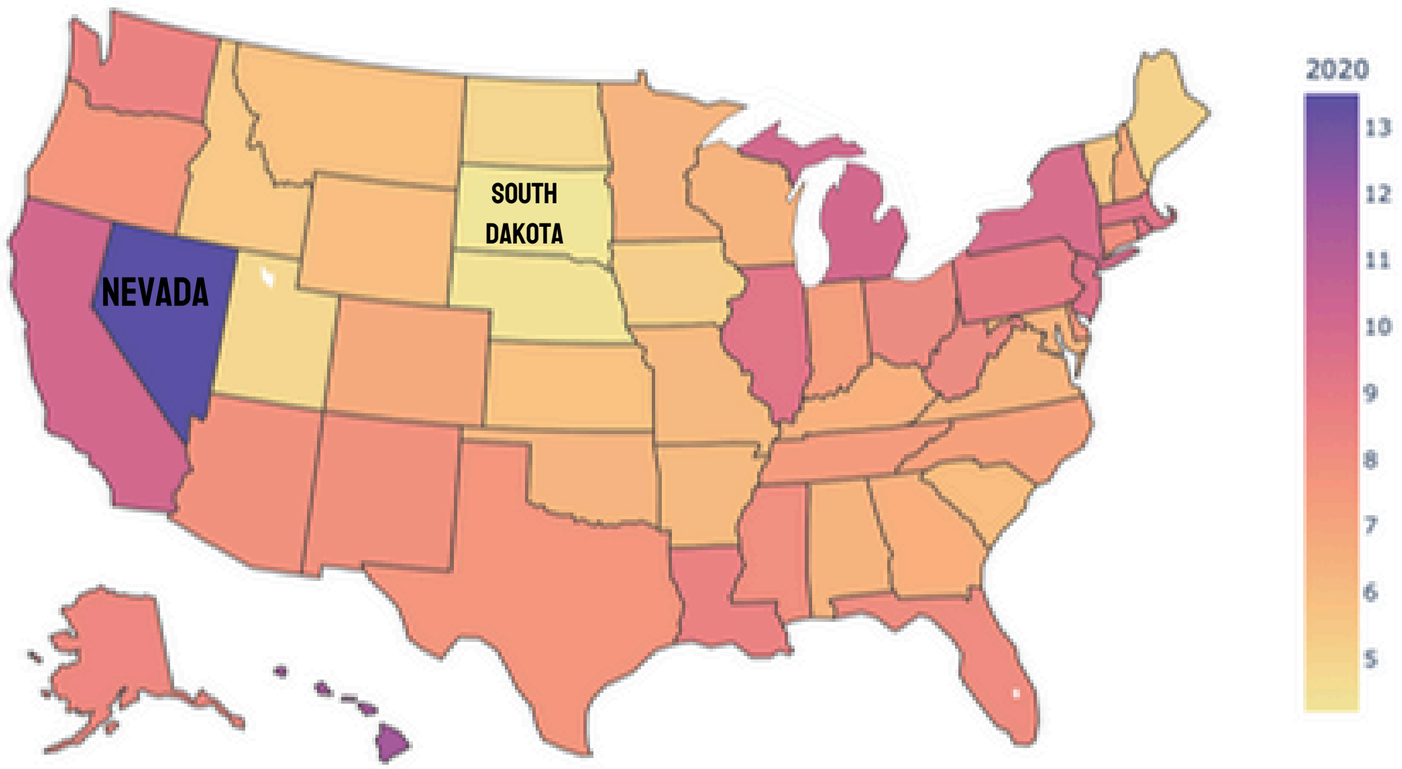
```
unemploy = pd.read_excel('/content/unemployement_per_state.xlsx',
                         skiprows=5)

unemploy = unemploy.drop(index=0)
unemploy.columns.values[0] = 'Region'
unemploy.head()
```

```
unemploy = unemploy.loc[:, ~unemploy.columns.astype(str).str.contains('Unnamed')]
unemploy = unemploy.drop(unemploy.index[52:])
unemploy = unemploy.applymap(lambda x: x.strip() if isinstance(x, str) else x)

unemploy.head()
```

	Region	2023	2022	2021	2020	2019	2018	2017	2016	2015	...	2009	2008	2007	2006
1	United States	3.6	3.6	5.3	8.1	3.7	3.9	4.4	4.9	5.3	...	9.3	5.8	4.6	4.6
2	Alabama	2.5	2.5	3.4	6.4	3.2	3.9	4.5	5.9	6.1	...	10.1	5.8	4.1	4.0
3	Alaska	4.2	4.2	6.4	8.3	5.6	6.0	6.5	6.6	6.3	...	8.0	6.5	6.3	6.6
4	Arizona	3.9	3.8	5.1	7.8	4.8	4.8	5.0	5.5	6.1	...	9.7	5.8	3.8	4.3
5	Arkansas	3.3	3.2	4.0	6.2	3.5	3.7	3.7	4.0	5.0	...	7.7	5.4	5.4	5.2

5 rows × 25 columns

Unemployment Rate in 2020 per Region in the USA




```
state_mapping = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA',
    'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'District of Columbia': 'DC',
    'Florida': 'FL', 'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL',
    'Indiana': 'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA',
    'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI',
    'Minnesota': 'MN', 'Mississippi': 'MS', 'Missouri': 'MO', 'Montana': 'MT',
    'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH', 'New Jersey': 'NJ',
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND',
    'Ohio': 'OH', 'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI',
    'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX',
    'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA',
    'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'
}
unemploy['Region'] = unemploy['Region'].str.title()
unemploy['Region'] = unemploy['Region'].str.strip()
unemploy['code_state'] = unemploy['Region'].map(state_mapping)
```

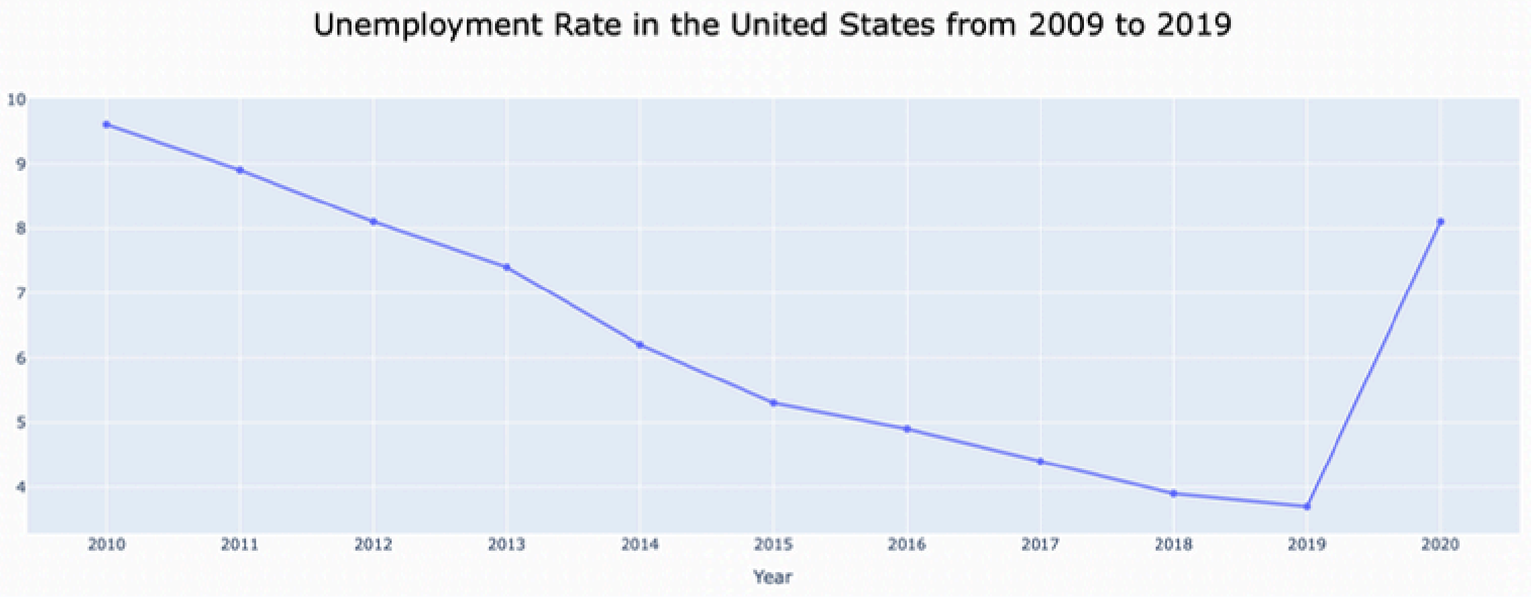
```
import plotly.express as px

fig = px.choropleth(
    unemploy,
    locations='code_state',  # Use the state abbreviations
    locationmode='USA-states',  # Match US states
    color=2020,  # Color by unemployment rate 2019
    scope='usa',  # Focus on the USA
    color_continuous_scale= 'sunset',  # Use a red-to-orange color map
    )

fig.update_layout(
    title={
        'text': "Unemployment Rate in 2020 per Region in the USA ",
        'font': {
            'size': 24,  # Adjust the font size as needed
            'color': 'black'  # Set the title color to black
        },
        'x': 0.5,  # Center the title
        'xanchor': 'center'  # Anchor the title at the center
    }
)

# Display the map
fig.show()
```

- **We assigned the Region Index to the data by mapping the code_state to illustrate the unemployment rate in the U.S. Regions..**
- **The darker map colors indicate a higher unemployment rate and the lighter is the lowesr**
- **From the data, The highest unemployment rate is in Nevada, while the lowest is in South Dakota**

THE US UNEMPLOYMENT RATE

We plot the unemployment rate data to illustrate the unemployment trend in the country

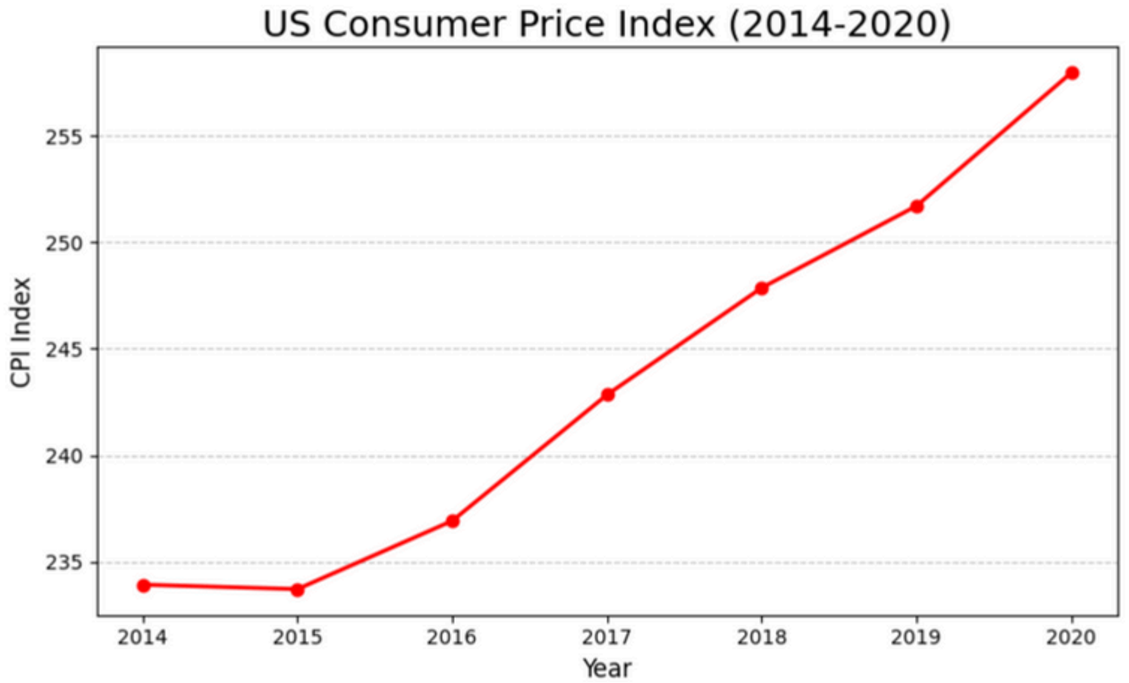
```
import plotly.express as px

unemploy_melted = unemploy.melt(id_vars=['Region', 'code_state'],
                                var_name='Year',
                                value_name='Unemployment Rate')

unemploy_melted['Year'] = unemploy_melted['Year'].astype(int)
unemploy_us = unemploy_melted[(unemploy_melted['Region'] == 'United States')
& unemploy_melted['Year'].between(2010, 2020)]

fig = px.line(
    unemploy_us,
    x='Year',
    y='Unemployment Rate',  #
    title="Unemployment Rate in the United States from 2010 to 2020",
    labels={'Unemployment Rate': 'Unemployment Rate (%)'},
    line_shape='linear',
    markers=True
)
fig.update_layout(
    title={
        'text': "Unemployment Rate in the United States from 2010 to 2020",
        'font': {
            'size': 24,
            'color': 'black'
        },
        'x': 0.5,
        'xanchor': 'center'
    },
    xaxis_title="Year",
    yaxis_title="Unemployment Rate (%)",
    showlegend=False,
    xaxis=dict(
        tickmode='linear',
        dtick=1
    )
)
fig.show()
```



The line grap shows that from 2010 to 2019 the unemployment rate in the U.S. gradually declined. However, in 2020 it rose significantly.

US-INFLATION

For the inflation data, we use the Consumer Price Index (CPI) provided by the Bureau of Labor Statistics.
The CPI data illustrates the inflation trends in the country.



After a slight decrease in 2015,
from 2016 to 2020, the U.S. CPI shows a consistent increase.

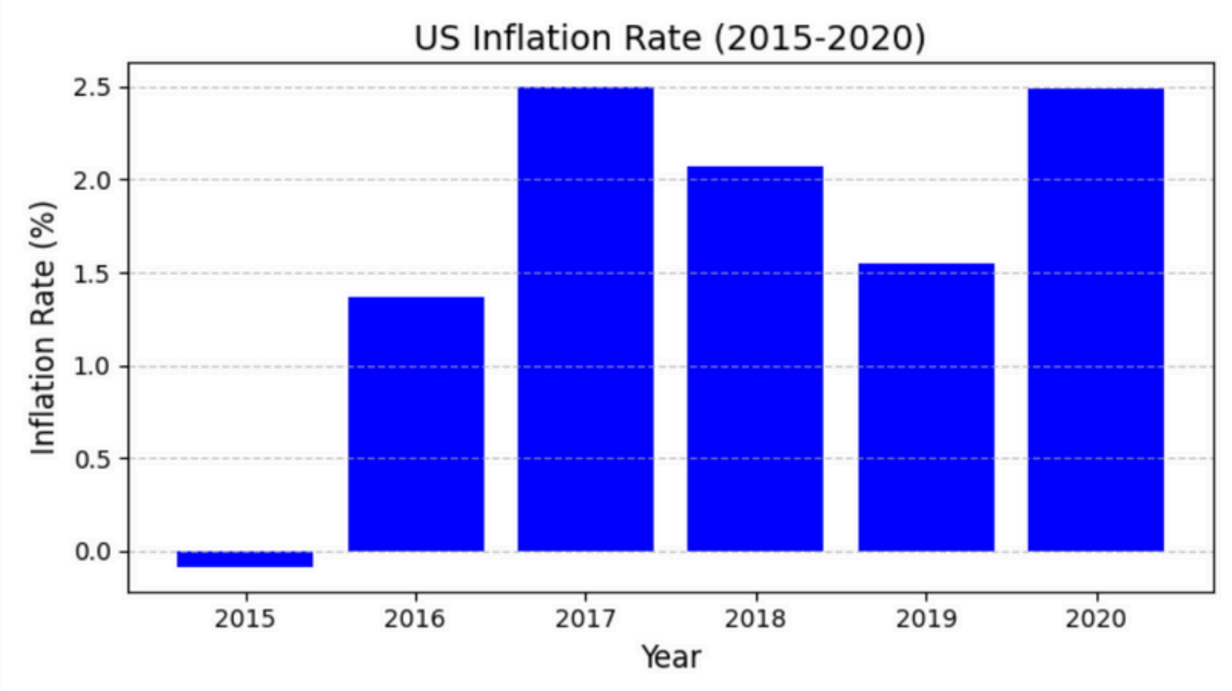
```
import pandas as pd

cpi = pd.read_excel('/content/CPI.xlsx', skiprows=11)
cpi.info()
```

```
#cpi['Inflation Rate'] = ((cpi['Jan'].shift(-1) - cpi['Jan']) / cpi['Jan']) * 100
cpi['Inflation Rate'] = ((cpi['Jan']) - cpi['Jan'].shift(+1)) / (cpi['Jan'].shift(+1)) * 100
cpi.loc[cpi['Year'] == 2014, 'Inflation Rate'] = 0
cpi['Inflation Rate'] = cpi['Inflation Rate'].round(2)
```

```
import matplotlib.pyplot as plt

# Filter the data to include only the years until 2020
inflation_df = cpi[cpi['Year'] <= 2020][['Year', 'Jan']]

# Plot the data as a line plot
plt.figure(figsize=(8, 5))
plt.plot(inflation_df['Year'].astype(str), inflation_df['Jan'],
         marker='o', color='red', linestyle='-', linewidth=2)

# Add labels and title
plt.title('US Consumer Price Index (2014-2020)', fontsize=18)
plt.xlabel('Year', fontsize=12)
plt.ylabel('CPI Index', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()
plt.show()
```

US-INFLATION

After a slight deflation in 2015, the inflation came back with fluctuated rate from 2016 to 2020.




```python
import matplotlib.pyplot as plt

# Filter the data for years <= 2020
inflation_df2 = cpi[(cpi['Year'] > 2014) & (cpi['Year'] <= 2020)][['Year', 'Inflation Rate']]

# Plot the data
plt.figure(figsize=(7, 4))
plt.bar(inflation_df2['Year'].astype(str), inflation_df2['Inflation Rate'], color='blue')

# Add labels and title
plt.title('US Inflation Rate (2015-2020)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Inflation Rate (%)', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()
plt.show()
```

03. PREDICTING VOTER PREFERENCES

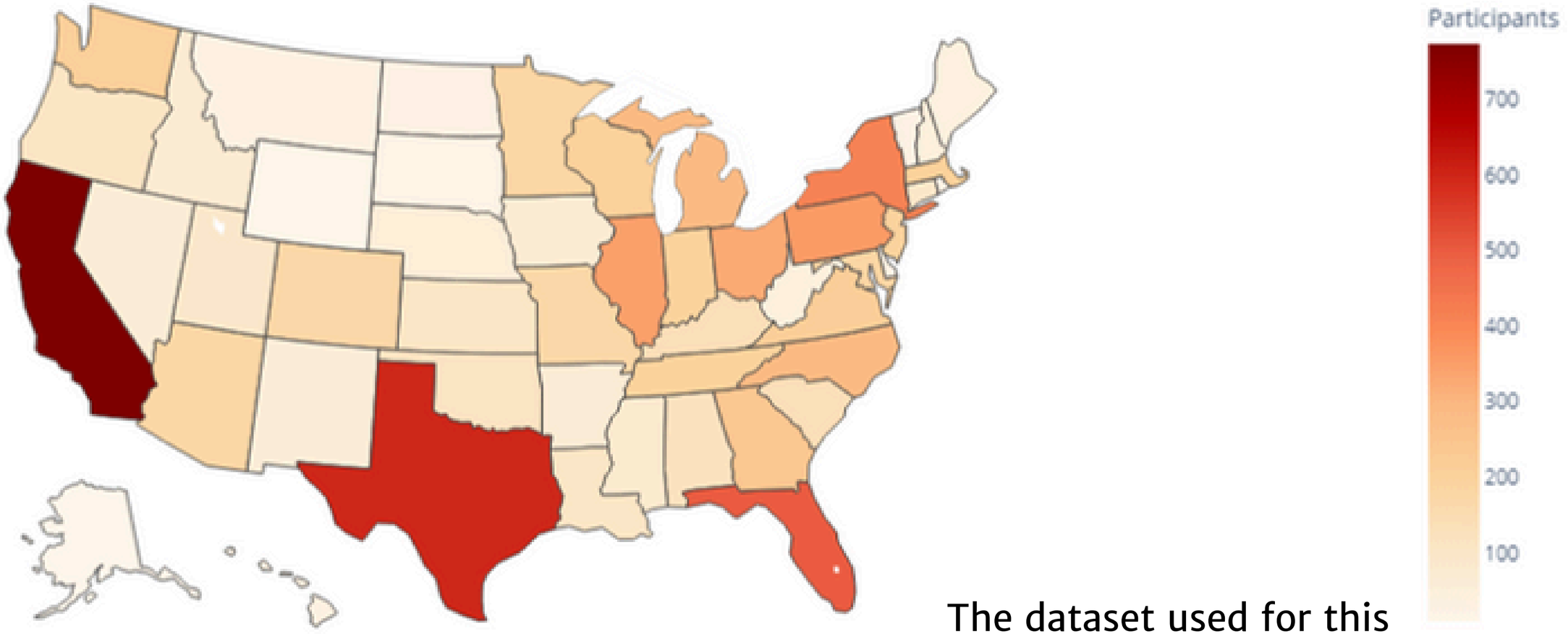
01 Data manipulation and preparation that involved constructing the necessary dataframes, encoding categorical variables, and ensuring the data was in the correct format.

02 Assessment of correlation and multicollinearity through the computation of the correlation matrix and the calculation of the VIF for the predictor variables.

03 Multivariate logistic regression analysis, followed by an assessment of model performance.

04 Identifying the most significant variables, then visualizing their impact on the response classes.

State-wise 2020 Presidential Election Questionnaire Participants



The dataset used for this section is the ***ANES 2020 Times Series Study***.

THE DATA

To load and prepare the data upon which our analysis is based, we ran the following commands.

```
# Loading the dataset containg the regressors we're going to use
regressors=pd.read_csv("Time_series_2020.csv", low_memory=False)
# Loading the dataset containg the two responses
df_y=pd.read_excel("Response.xlsx")
```

```
[148] variables=['V203000','V201004','V201016','V201127','V201130','V201133','V201136','V201139','V201142',

                 'V201300','V201303','V201306','V201309','V201312','V201318','V201321','V201324','V201328','V201331',
                 'V201334','V201350','V201351','V201352','V201360','V201377','V201380',

                 'V201554','V201596','V201603','V201617x']

      df_x=regressors[variables]
      df_x=df_x.rename(columns={'V203000':'State_cod'})
      # df_x
```

```
# Merging df_x and df_y
df=pd.merge(df_x,df_y,left_index=True,right_index=True,how='inner')
df
```

Firstly, we constructed the dataframe *df*. Then, we created two dataframes with the same variables but differing only in their response variables: *Pref_President* and *Pref_Party*.

```
[78] # Dataframe considering just the response Pref_President
     df_President = df.drop(['Pref_Party'], axis = 1)
     df_President
```

```
[79] # Dataframe considering just the response Pref_Party
     df_Party = df.drop(['Pref_President'], axis = 1)
     df_Party
```

THE DATA

This is the dataframe *df*, presenting information on 8280 US voters across 29 categorical variables (excluding the response variables).

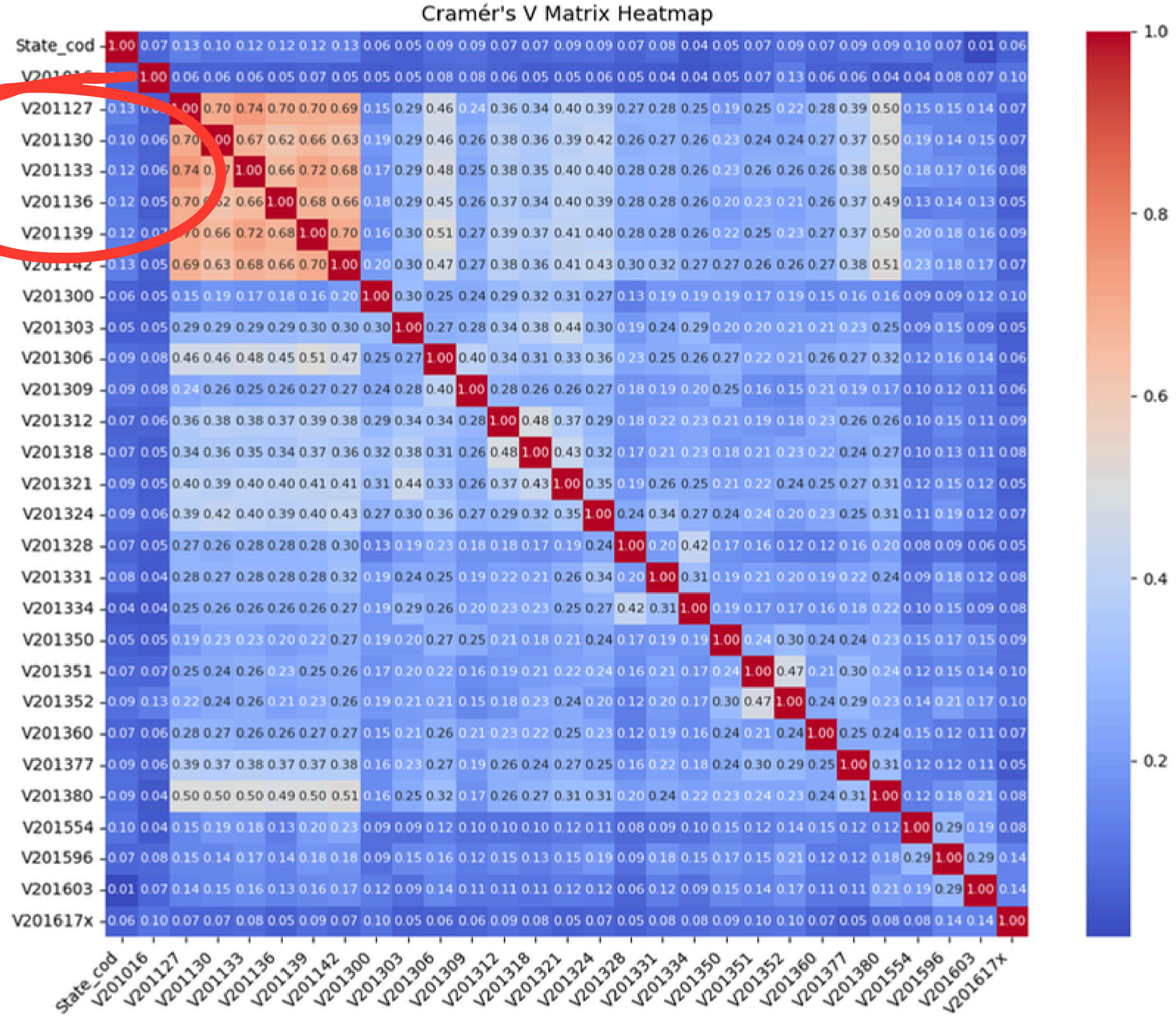
	State_cod	V201004	V201016	V201127	V201130	V201133	V201136	V201139	V201142	V201300	...	V201352	V201360	V201377	V201380	V201554	V201596	V201603	V201617x	Pref_President	Pref_Party
0	40	-1	3	1	1	1	1	1	1	3	...	3	2	1	2	4	1	1	21	2	1
1	16	-1	3	2	1	2	1	2	2	3	...	2	2	1	3	1	1	1	13	3	1
2	51	-1	3	2	2	2	2	2	2	1	...	4	1	3	1	1	2	3	17	1	0
3	6	-1	3	2	2	2	2	2	2	3	...	4	1	4	1	4	2	1	7	1	1
4	8	-1	2	1	1	1	2	1	2	3	...	2	1	2	1	1	2	1	22	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8275	12	2	1	1	1	1	1	1	1	1	...	2	1	1	1	4	2	1	8	2	1
8276	16	2	3	1	1	2	1	1	1	1	...	5	1	2	3	4	1	1	19	2	-1
8277	6	-1	2	2	2	2	2	2	2	3	...	3	1	3	1	4	1	1	16	1	-1
8278	51	-1	3	1	1	1	1	1	1	1	...	3	2	1	2	1	2	1	14	-1	1
8279	51	2	1	2	2	2	2	2	2	2	...	4	1	5	1	4	2	1	15	1	-1

```
# print(df.dtypes)
# Convert each column to 'category' type
categorical_columns = ['State_cod', 'V201016', 'V201127', 'V201130', 'V201133', 'V201136', 'V201139', 'V201142',
                       'V201300', 'V201303', 'V201306', 'V201309', 'V201312', 'V201318', 'V201321', 'V201324',
                       'V201328', 'V201331', 'V201334', 'V201350', 'V201351', 'V201352', 'V201377', 'V201360',
                       'V201554', 'V201596', 'V201617x', 'Pref_President', 'Pref_Party']

df[categorical_columns] = df[categorical_columns].apply(lambda x: x.astype('category'))
```

After creating the two dataframes, we grouped similar categories to reduce complexity and improve interpretability while ensuring consistency.

CORRELATION ANALYSIS

	Variable	VIF
0	const	12.882909
1	V201127	5.680028
2	V201130	3.138071
3	V201133	4.583188
4	V201136	3.645074
5	V201139	3.845255
6	V201142	3.596304



MULTIVARIATE LOGISTIC REGRESSION: DF_PARTY

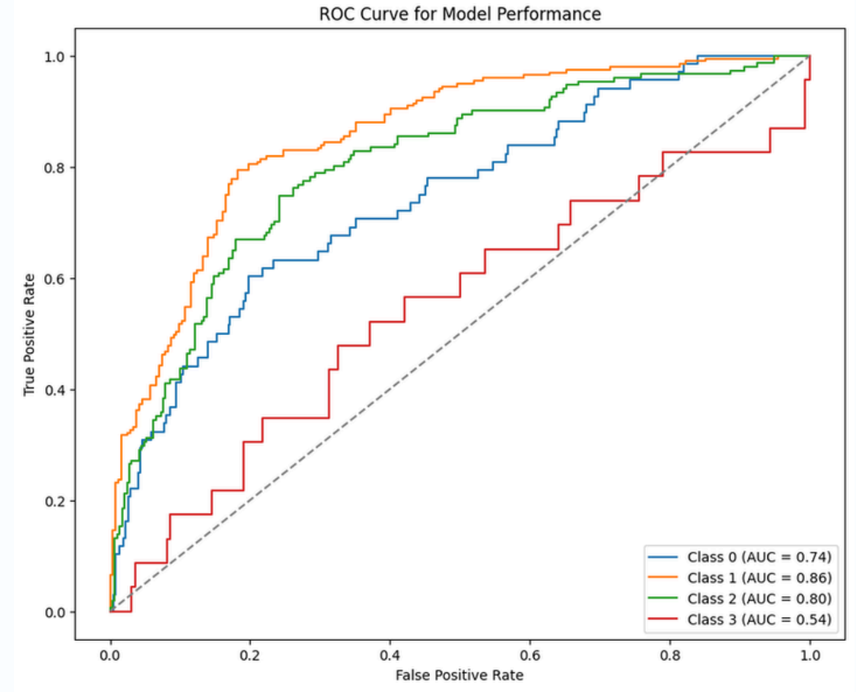
```
[177] df_Party['Pref_Party'] = pd.Categorical(df_Party['Pref_Party'], categories=[0, 1, 2, 3], ordered=True)
      # setting 'inapplicable' (class 0) as the baseline
      # this way, the model would assess how the other classes differ from those who did not respond

      X1 = df_Party.drop('Pref_Party', axis=1)
      y1 = df_Party['Pref_Party']
      X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.20, random_state=42)

      # Standardize the regressors so they have zero mean and variance = 1
      scaler = StandardScaler()
      X1_train = scaler.fit_transform(X1_train)
      X1_test = scaler.transform(X1_test)

      # Add the intercept beta0
      X1_train = sm.add_constant(X1_train)

      # Train multivariate logistic regression using the training data
      model = sm.MNLogit(y1_train, X1_train)
      result = model.fit()
      print(result.summary())
```

The report only reflects the test set

Classification Report:

	precision	recall	f1-score	support
0	0.61	0.36	0.45	336
1	0.71	0.84	0.77	696
2	0.68	0.76	0.72	561
3	0.00	0.00	0.00	63
accuracy			0.68	1656
macro avg	0.50	0.49	0.48	1656
weighted avg	0.65	0.68	0.66	1656

The model performs well overall: its metrics assign greater weight to the more frequent classes (*Democratic* and *Republican*). The remaining classes are not predicted as accurately (class imbalance).



MODEL PERFOMANCE

```
# The following results represent the probabilities of each observation (row) in df_Party belonging to each of the three response classes

name_mapping = {'x1':'State_cod', 'x2':'V201016', 'x3':'V201130', 'x4':'V201133', 'x5':'V201136', 'x6':'V201139', 'x7':'V201142',
                'x9':'V201303', 'x10':'V201306', 'x11':'V201309', 'x13':'V201318', 'x14':'V201321', 'x15':'V201324',
                'x16':'V201328', 'x19':'V201351', 'x20':'V201351', 'x21':'V201352', 'x22':'V201360', 'x23':'V201377', 'x24':'V201380',
                'x25':'V201554', 'x27':'V201603', 'x28':'V201617x'}

# Significant predictors for each class based on p-values < 0.5
significant_vars_class1 = ['x2', 'x3', 'x5', 'x6', 'x9', 'x11', 'x15', 'x19', 'x21', 'x22', 'x23', 'x28']
significant_vars_class2 = ['x2', 'x6', 'x7', 'x10', 'x13', 'x14', 'x15', 'x16', 'x20', 'x21', 'x24', 'x25', 'x27', 'x28']

# Retrieve the indices of the significant variables in df_Party
significant_vars_class1_indices = [list(df_Party.columns).index(name_mapping[var]) for var in significant_vars_class1]
significant_vars_class2_indices = [list(df_Party.columns).index(name_mapping[var]) for var in significant_vars_class2]

# Now, use the indices to access the data in the NumPy array
X_significant_class1 = X1_train[:, significant_vars_class1_indices]
X_significant_class2 = X1_train[:, significant_vars_class2_indices]

# Calculate log-odds for each class using the significant variables
#The line below is changed to select the correct parameters from result.params based on the significant variables for each class
log_odds_1 = result.params[1][significant_vars_class1_indices] @ X_significant_class1.T  # Log-odds for class 1
log_odds_2 = result.params[2][significant_vars_class2_indices] @ X_significant_class2.T  # Log-odds for class 2
#log_odds_3 = result.params[0][significant_vars_class3_indices] @ X_significant_class3.T  # Log-odds for class 3

# Stack the log-odds for the three classes together, including a baseline for Pref_Party=0 ('Inapplicable')
log_odds_all = np.column_stack([np.zeros(X_significant_class1.shape[0]), log_odds_1, log_odds_2])

# Apply the 'softmax' function to compute probabilities for each class
exp_log_odds = np.exp(log_odds_all)
probabilities = exp_log_odds / np.sum(exp_log_odds, axis=1, keepdims=True)

# Probabilities for Pref_Party=1 and Pref_Party=2
print(f"Pref_Party=1, Pref_Party=2:\n{probabilities[:, 1:]}")
# We can identify the most likely class for each observation by looking at which column (class) has the highest probability
```

Calculating the probability of each response class for every observation

```
# Now, we can compute how many observations are likely to fall into each of the three classes

# First of all, we have to identify the class with the highest probability for each observation (excluding the baseline)
predicted_classes = np.argmax(probabilities[:, 1:], axis=1) + 1  # '+1' to match the class labels

# Count how many observations fall into each class
class_labels = [1, 2]
counts = [np.sum(predicted_classes == label) for label in class_labels]

# Display the count for each class in the specified order
for class_label, count in zip(class_labels, counts):
    print(f"Class {class_label}: {count} observations")
```

```
Class 1: 3319 observations
Class 2: 3305 observations
```

The model might be biased toward the majority classes due to class imbalance, as Class 3 is rare

```
count_class_2 = (df_Party['Pref_Party'] == 2).sum()
count_class_1 = (df_Party['Pref_Party'] == 1).sum()
print(f"Number of observations where Pref_Party = 1: {count_class_1}")
print(f"Number of observations where Pref_Party = 2: {count_class_2}")
```

```
Number of observations where Pref_Party = 1: 3362
Number of observations where Pref_Party = 2: 2846
```

SIGNIFICANT REGRESSORS

```
[72] # Show only the significant variables for which the p-value < 0.05

     significant_vars = {}

     for category in result.pvalues.columns:
         significant_vars[category + 1] = result.pvalues[result.pvalues[category] < 0.05].index.tolist()

     for category, v in significant_vars.items():
         print(f"\nSignificant variables for class {category}: {v}")
```

Removing variables with p-values greater than 0.05.

DEMOCRATIC PARTY

Democratic supporters focus more on how Trump's handling key issues such as:

- the **economy** (V201130);
- **healthcare** (V201136);
- **immigration** (V201139).

Other important aspects are:

- **federal spending on public schoo**l (V201303);
- **federal spending on crime prevention** (V201309);
- **federal spending on international issu**es (V201350);
- **trust in media accuracy** (V201377);
- **election oversight trust** (V201352);
- **support for voting rights for convincted felons** (V201360).

Economic perceptions and **income** also play a role (V201324, V201617x).

REPUBLICAN PARTY

The most significant factors include how Trump is handling:

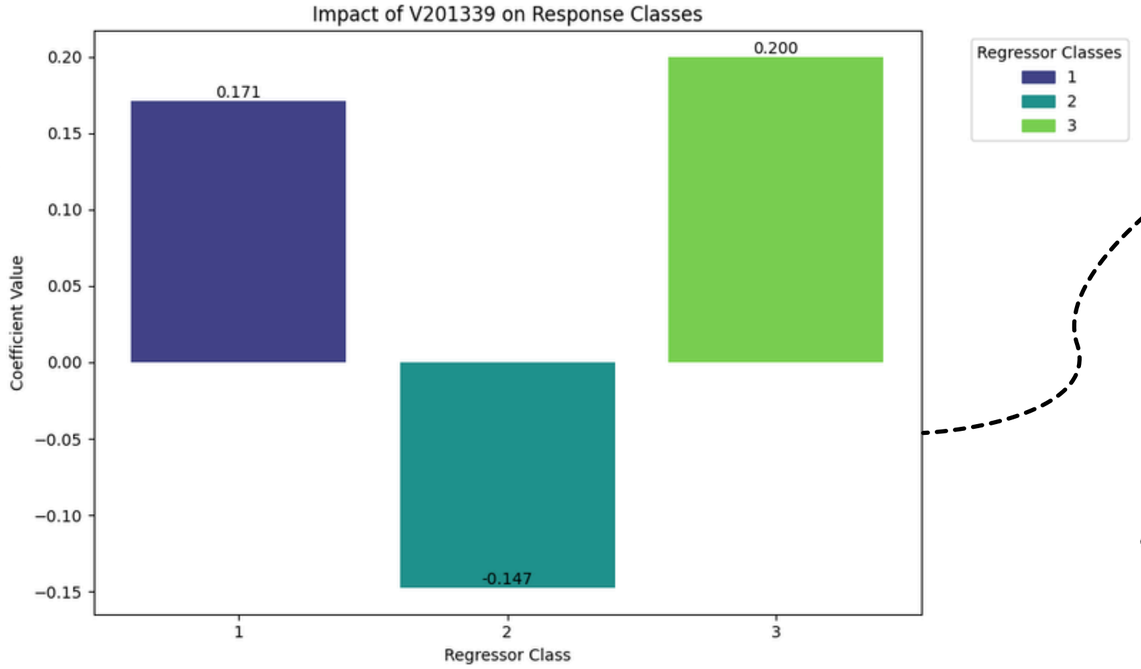
- **the economy** (V201324, V201328);
- **border security** (V201306);
- **the pandemic response** (V201142);
- **budget spending on aid to the poor** (V201318);
- **immigration** (V201139);
- **environmental protection** (V201321).

Other crucial points are:

- **trust in election integrity and officials** (V201351, V201352)
- **views on corruption since Trump** (V201380)
- **political violence** (V201603).

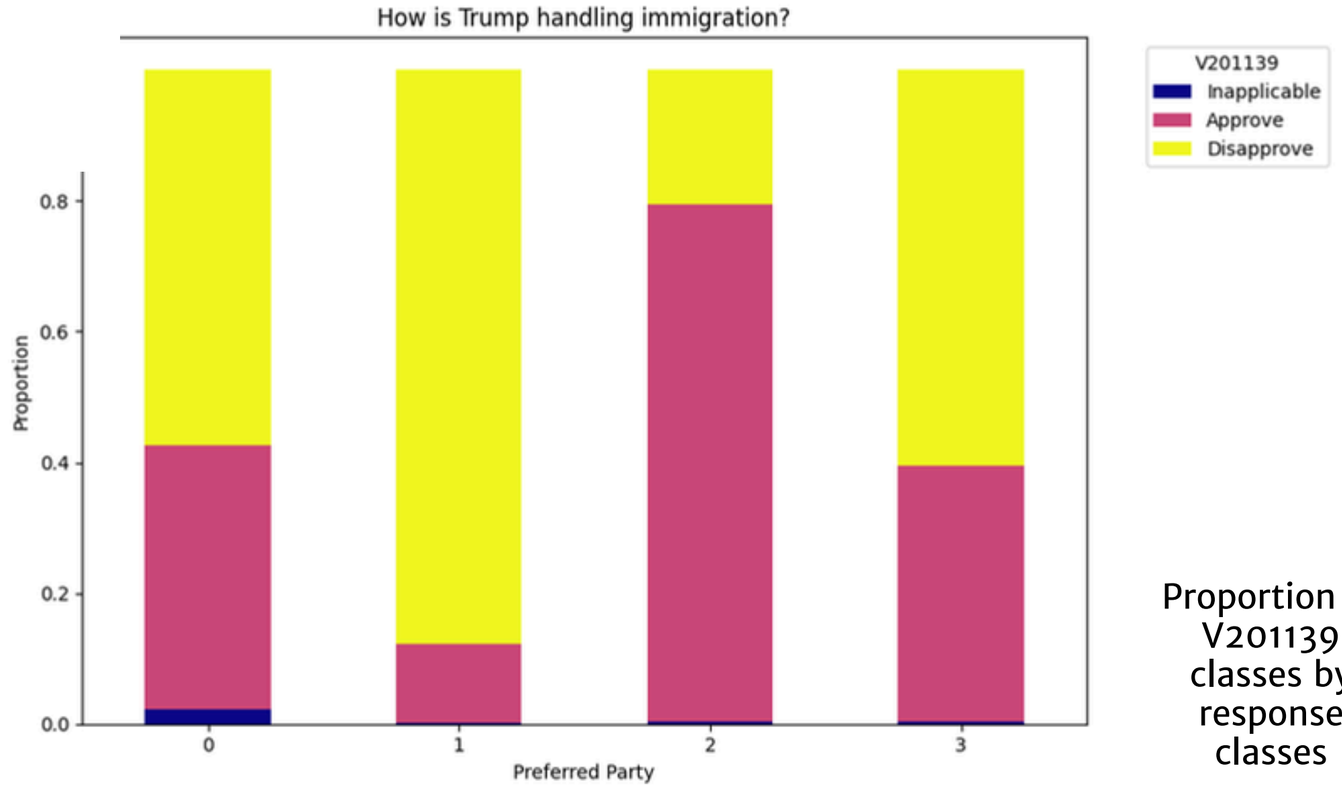
Family income (V201617x) and **country of birth** (V201554) are also notable factors.

Both groups share concerns about the economy, immigration, and family income.

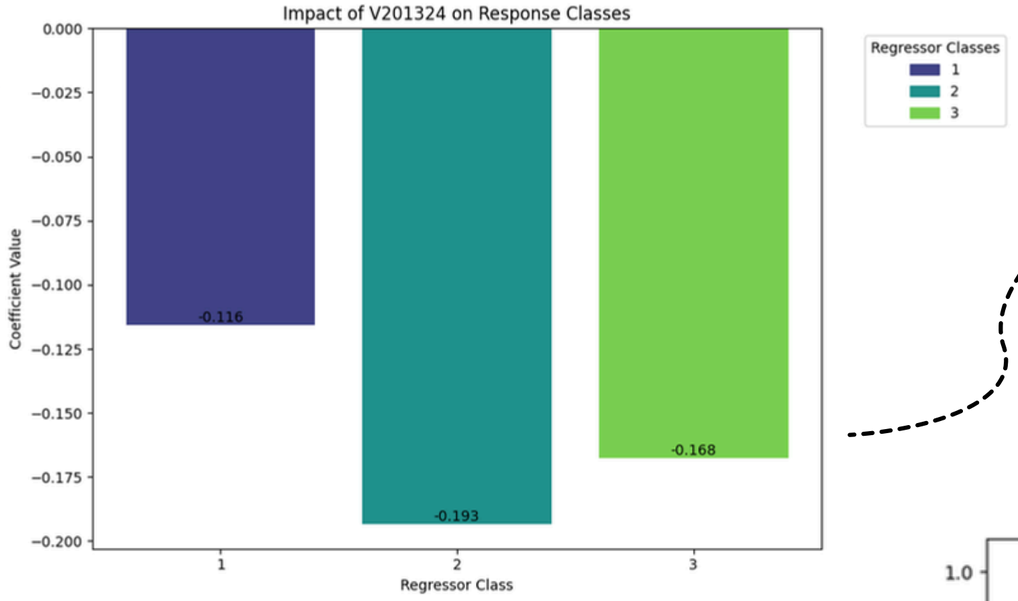


Change in the ***log-odds*** of being in a specific response class compared to the baseline class

OVERLAPPING VARIABLES

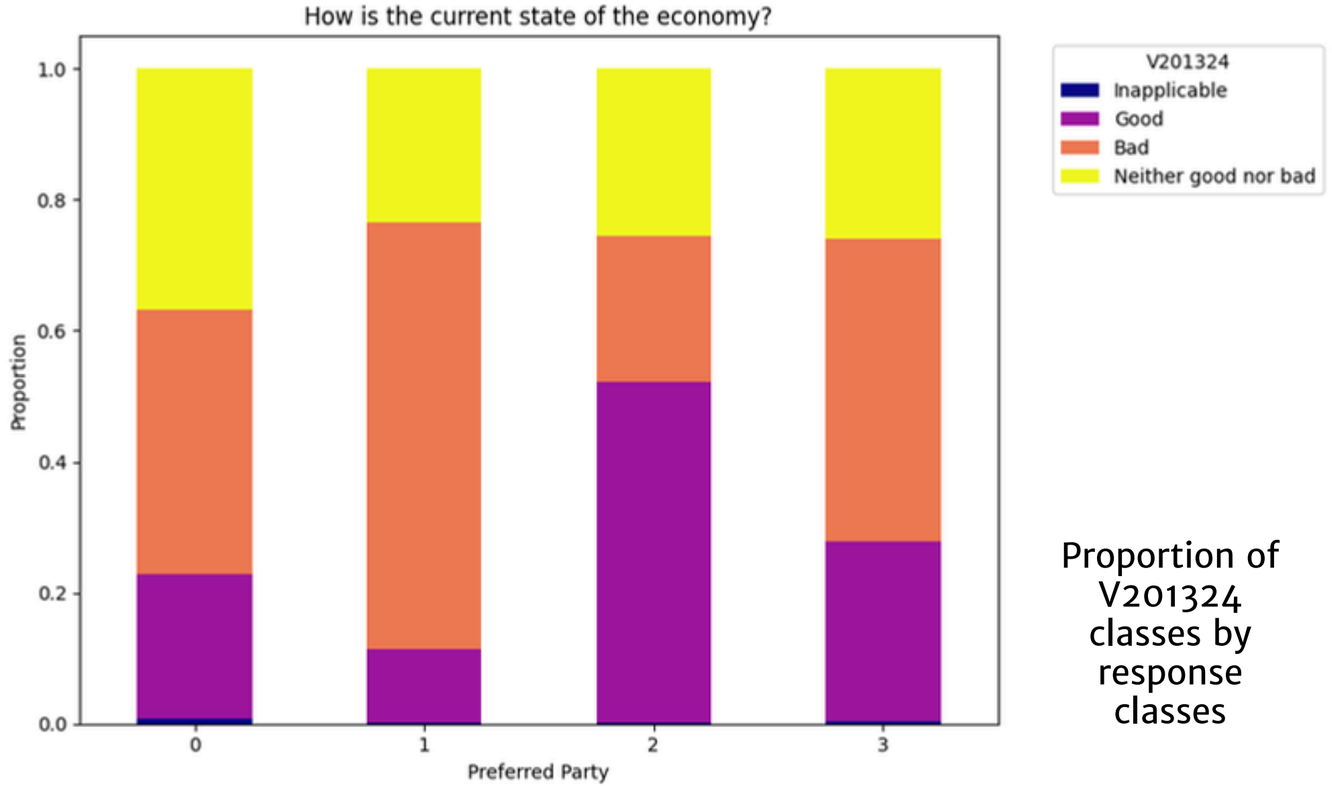


Proportion of V201139 classes by response classes



Change in the ***log-odds*** of being in a specific response class compared to the baseline class

OVERLAPPING VARIABLES

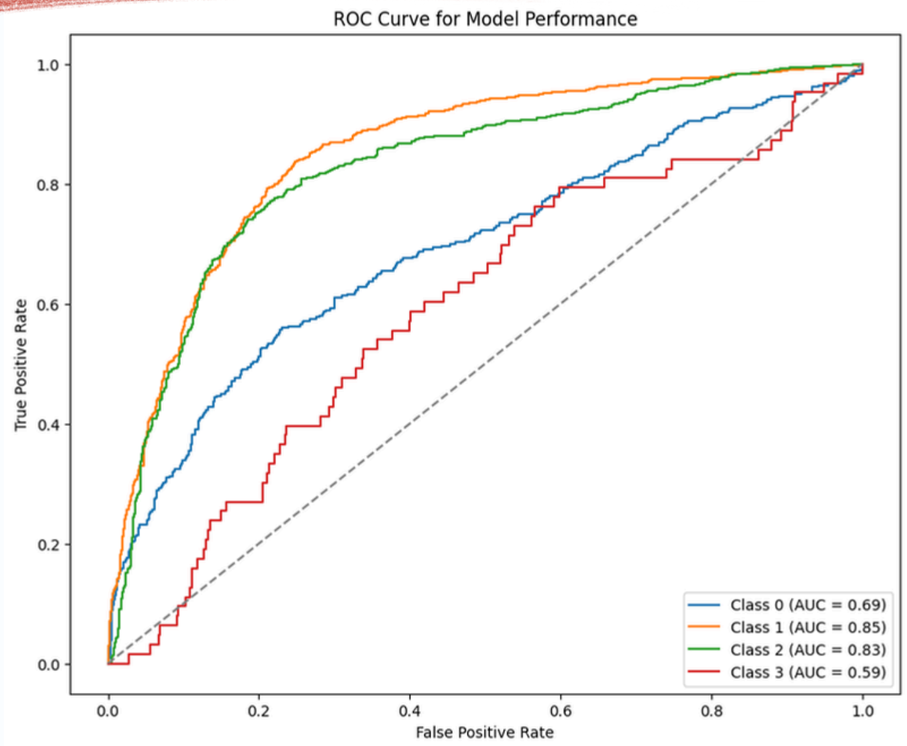


Proportion of V201324 classes by response classes

MULTIVARIATE LOGISTIC REGRESSION: DF_PRESIDENT

```
df_President['Pref_President'] = pd.Categorical(df_President['Pref_President'], categories=[0, 1, 2, 3], ordered=True)
# setting 'inapplicable' (class 0) as the baseline
# this way, the model would assess how the other classes differ from those who did not respond

X2 = df_President.drop('Pref_President', axis=1)
y2 = df_President['Pref_President']
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, random_state=42)

# Standardize the regressors so they have zero mean and variance = 1
scaler = StandardScaler()
X2_train = scaler.fit_transform(X2_train)
X2_test = scaler.transform(X2_test)

# Add the intercept beta0
X2_train = sm.add_constant(X2_train)

# Train multivariate logistic regression using the training data
model = sm.MNLogit(y2_train, X2_train)
result = model.fit()
print(result.summary())
```

The weighted average recall and weighted average F1-score are encouraging, suggesting that the model is performing well. However, as before, classes 0 and 3 are not predicted very well.



Classification Report:

	precision	recall	f1-score	support
0	0.35	0.06	0.11	207
1	0.82	0.97	0.88	777
2	0.84	0.95	0.89	611
3	0.00	0.00	0.00	61
accuracy			0.81	1656
macro avg	0.50	0.50	0.47	1656
weighted avg	0.74	0.81	0.76	1656

```
# The following results represent the probabilities of each observation (row) in df_President belonging to each of the three response classes

name_mapping = {'x1':'State_cod', 'x2':'V201016', 'x3':'V201130', 'x4':'V201133', 'x5':'V201136', 'x6':'V201139', 'x7':'V201142',
                'x9':'V201303', 'x10':'V201306', 'x11':'V201309', 'x13':'V201318', 'x14':'V201321', 'x15':'V201324',
                'x16':'V201328', 'x19':'V201351', 'x20':'V201351', 'x21':'V201352', 'x22':'V201360', 'x23':'V201377', 'x24':'V201380',
                'x25':'V201554', 'x27':'V201603', 'x28':'V201617x'}

# Significant predictors for each class based on p-values < 0.5
significant_vars_class1 = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x9', 'x14', 'x22', 'x24', 'x28']
significant_vars_class2 = ['x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x10', 'x13', 'x15', 'x19', 'x20', 'x24', 'x28']


# Retrieve the indices of the significant variables in df_President
significant_vars_class1_indices = [list(df_President.columns).index(name_mapping[var]) for var in significant_vars_class1]
significant_vars_class2_indices = [list(df_President.columns).index(name_mapping[var]) for var in significant_vars_class2]

# Now, use the indices to access the data in the NumPy array
X_significant_class1 = X2_train[:, significant_vars_class1_indices]
X_significant_class2 = X2_train[:, significant_vars_class2_indices]

# Calculate log-odds for each class using the significant variables
#The line below is changed to select the correct parameters from result.params based on the significant variables for each class
log_odds_1 = result.params[1][significant_vars_class1_indices] @ X_significant_class1.T  # Log-odds for class 1
log_odds_2 = result.params[2][significant_vars_class2_indices] @ X_significant_class2.T  # Log-odds for class 2
#log_odds_3 = result.params[0][significant_vars_class3_indices] @ X_significant_class3.T  # Log-odds for class 3
```

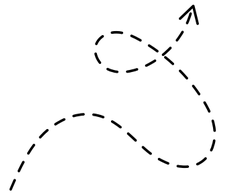
```
# Stack the log-odds for the three classes together, including a baseline for Pref_Party=0 ('Inapplicable')
log_odds_all = np.column_stack([np.zeros(X_significant_class1.shape[0]), log_odds_1, log_odds_2])

# Apply the 'softmax' function to compute probabilities for each class
exp_log_odds = np.exp(log_odds_all)
probabilities = exp_log_odds / np.sum(exp_log_odds, axis=1, keepdims=True)

# Probabilities for Pref_Party=1 and Pref_Party=2
print(f"Pref_President=1, Pref_President=2:\n{probabilities[:, 1:]}")
# We can identify the most likely class for each observation by looking at which column (class) has the highest probability
```

```
# Now, we can compute how many observations are likely to fall into each of the three classes

# First of all, we have to identify the class with the highest probability for each observation (excluding the baseline)
predicted_classes = np.argmax(probabilities[:, 1:], axis=1) + 1  # '+1' to match the class labels

# Count how many observations fall into each class
class_labels = [1, 2]
counts = [np.sum(predicted_classes == label) for label in class_labels]

# Display the count for each class in the specified order
for class_label, count in zip(class_labels, counts):
    print(f"Class {class_label}: {count} observations")
```

```
Class 1: 2978 observations
Class 2: 3646 observations
```

MODEL PERFOMANCE

Calculating the probability of each response class for every observation



The model might be biased toward the majority classes due to class imbalance, as Class 3 is rare

```
count_class_2 = (df_President['Pref_President'] == 2).sum()
count_class_1 = (df_President['Pref_President'] == 1).sum()
print(f"Number of observations where Pref_President = 1: {count_class_1}")
print(f"Number of observations where Pref_President = 2: {count_class_2}")
```

```
Number of observations where Pref_President = 1: 3843
Number of observations where Pref_President = 2: 3151
```

SIGNIFICANT REGRESSORS

```
[ ]  # Show only the significant variables for which the p-value < 0.05

     significant_vars = {}

     for category in result.pvalues.columns:
         significant_vars[category + 1] = result.pvalues[result.pvalues[category] < 0.05].index.tolist()

     for category, v in significant_vars.items():
         print(f"\nSignificant variables for class {category}: {v}")
```

Removing variables with p-values greater than 0.05.

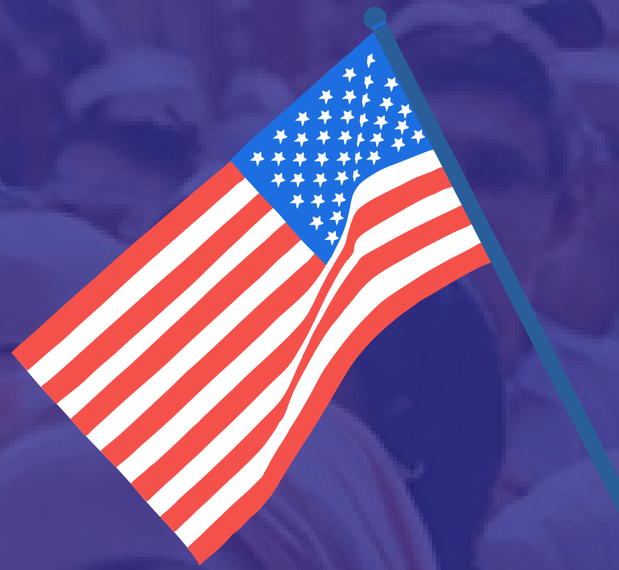
JOE BIDEN SUPPORTERS

Joe Biden supporters focus on how Trump's handling key issues, such as:

- the **economy** (V201130);
- **foreign relations** (V201133);
- **healthcare** (V201136);
- **immigration** (V201139;
- the **pandemic** (V201142).
-

They also pay attention to social factors such as:

- **trust in the media** (V201377);
- **voting rights for felons** (V201360).

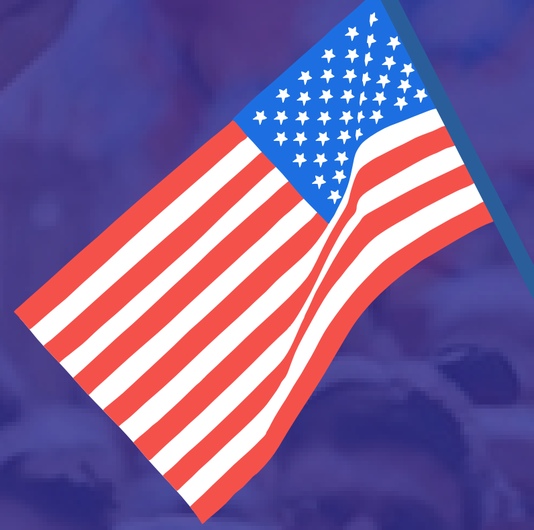


Additionaly, they are also concerned with public spending on services, such as:

- **education** (V201303);
- **crime prevention** (V201309).

Trust in election officials (V201352), the **general state of the economy** (V201324), and **total family income** (V291617x) are also significant.

DONALD TRUMP SUPPORTERS



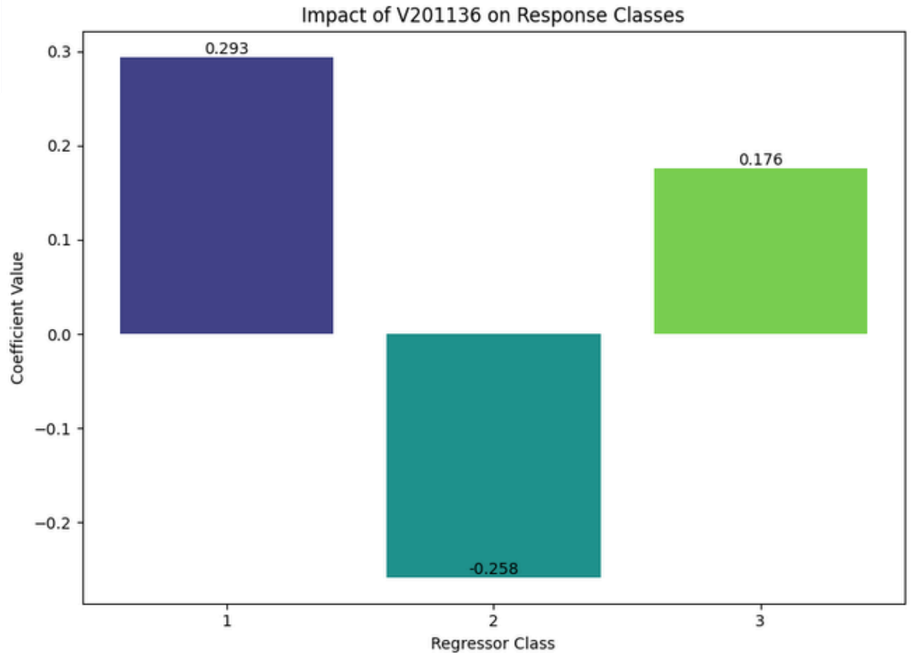
For Trump supporters, the most important factors focus on:

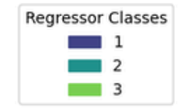
- **border security** (V201306);
- **immigration** (V201139);
- the **pandemic response** (V201142);
- **federal spending on aid to the poor** (V201318);
- **environmental protection** (V201321).

They are also concerned with:

- **corruption** (V201380);
- **political violence** (V201603), specifically in relation to Trump's leadership;
- **trust in election officials** (V201351);
- the **integrity of vote counting in the 2020 election** (V201351)

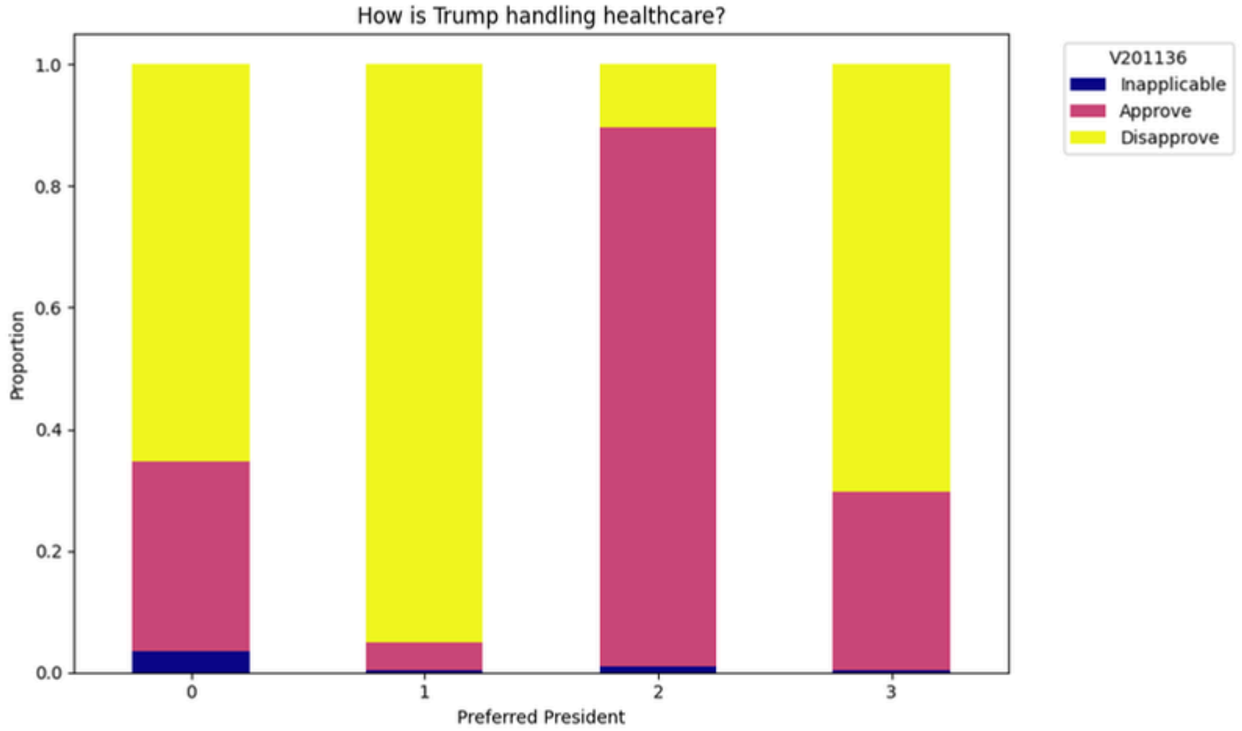
Economic concerns play a large role, including **perceptions of the current economy** (V201324) and **future outlook** (V201328), along with **total family income** (V201617x). Demographic factors like **country of birth** (V201554) are also relevant.



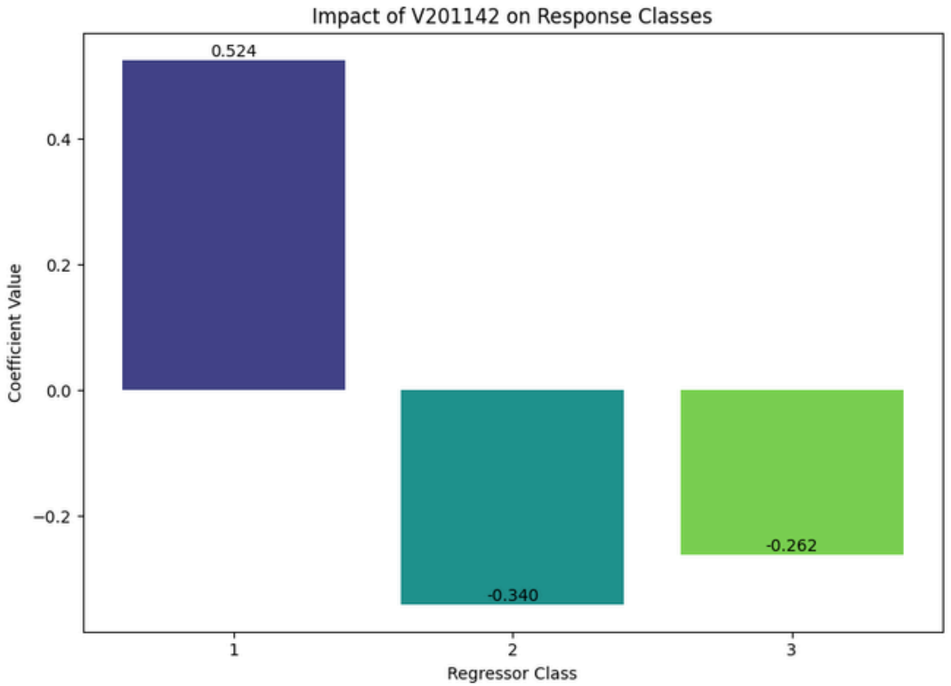


Change in the ***log-odds*** of being in a specific response class compared to the baseline class

OVERLAPPING VARIABLES

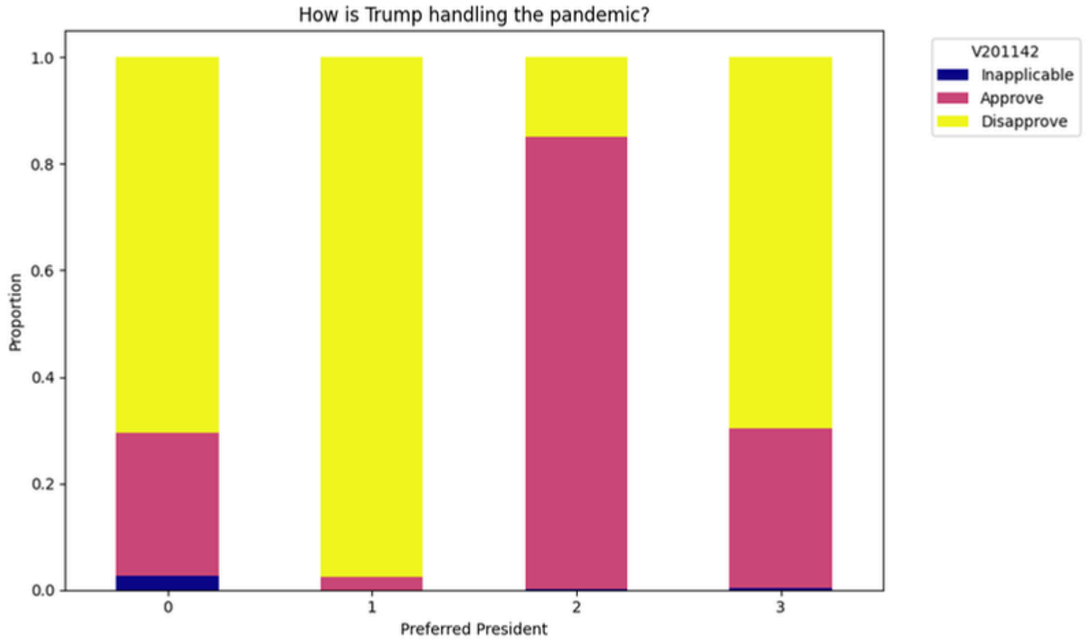


Proportion of V201136 classes by response classes



Change in the ***log-odds*** of being in a specific response class compared to the baseline class

OVERLAPPING VARIABLES



Proportion of V201142 classes by response classes

THE CONCLUSIONS

1. *The US Economy significantly affected Voter Preferences in the 2020 Election.*

2. *The US Healthcare Policy significantly affected Voter Preferences in the 2020 Election.*

3. *The US Foreign Policy significantly affected Voter Preferences in the 2020 Election.*

4. *The US Social Policy significantly affected Voter Preferences in the 2020 Election.*

5. *The other aspects that influenced Voter Preferences in the 2020 Election were:*

- **The Pandemic Handling/Response;**
- **Public Spending on Education;**
- **Public Spending on Crime Prevention;**
- **Trust in Election Officials;**
- **Total Family Income;**
- **Federal Spending on aid to the Poor;**
- **Corruption;**
- **Political violence (Leadership Style);**
- **The Integrity of Vote Counting;**
- **Demographic factors (e.g. Country of Birth).**

THANK YOU!

TRES ANALYTICS GROUP:

Lestari Suzi (908370)
Massano Elisa (905582)
Sanober (908664)