"*Who Told You What to Believe?*"

Deconstructing the Social Blueprint of Political Identity through Machine Learning

MACHINE LEARNING FINAL PROJECT

In partial fulfillment of the requirements for the subject Machine Learning

1. Executive Summary

This project investigates the complex social factors shaping political identity, focusing on the relative influence of family and peer groups. Utilizing a comprehensive survey and advanced machine learning techniques, the study analyzes how various social interaction elements—such as frequency of political discussions, emotional tone, social approval, and perceived pressure—contribute to an individual's political alignment.

Through classification models including Random Forest, AdaBoost, Logistic Regression, and Linear SVM, the research identifies which social context, family or peers, predominantly influences political beliefs. Random Forest achieved the highest predictive accuracy, while Logistic Regression and Linear SVM demonstrated strong generalizability due to minimal overfitting. AdaBoost also showed promising results by effectively capturing nuanced social patterns.

The findings underscore that political identity is not formed in isolation but is a dynamic outcome of ongoing social engagement with family and peers. This highlights the importance of understanding political socialization as a social process, where approval, discussion, and reflection within these groups shape political views.

Survey questions were carefully chosen and validated based on established political socialization theories, ensuring that key dimensions of social influence and individual demographics were effectively captured. The study's insights can inform tailored civic education initiatives and community programs aimed at promoting critical political thinking and independent political identities.

Overall, this research bridges theoretical perspectives with data-driven methods to deepen our understanding of the social blueprint behind political identity formation, with practical implications for enhancing political engagement and social cohesion.

2. Introduction

The formation of political identity is a complex process influenced by various social agents, with family and peers often cited as primary drivers of individual belief systems. This project examines the relative impact of these two key social contexts on shaping political alignment. While prior research acknowledges the significant role of socialization in political development (Jennings & Niemi, 1968; Beck & Jennings, 1991), there remains a need to quantitatively deconstruct the multifaceted dimensions of influence and their differential effects.

Drawing on theoretical frameworks from political socialization theory, social identity theory (Tajfel & Turner, 1979), and research on group conformity (Asch, 1955), this study posits that political beliefs are not formed in isolation but are, instead, the result of intricate social dynamics. Specifically, we investigate how the frequency of political discussions, the emotional tone of those interactions, perceived social pressure, and the salience of social approval within family and peer groups collectively shape individual political orientation.

To empirically assess these dynamics, we employ machine learning techniques to analyze a comprehensive survey dataset. This approach allows us to classify individuals based on whether their political alignment is more strongly influenced by family or peer contexts, considering a range of predictor variables that capture both direct and indirect dimensions of social influence.

The primary objective of this project is to move beyond descriptive accounts of political socialization and provide a quantitative, predictive model of how social contexts shape political identity. By uncovering the nuanced interplay of these influences, this research contributes to a more granular understanding of political behavior and informs discussions on political engagement, social cohesion, and the development of independent political thought.

3. Problem Statement

Despite extensive research acknowledging the importance of socialization in political identity formation, there remains a significant gap in understanding the relative influence of family versus peers on an individual’s political alignment. Traditional studies often treat these influences in isolation or rely on qualitative assessments, limiting the ability to systematically quantify and classify their impact. This lack of clarity hinders our capacity to comprehend how social environments shape political beliefs, especially in an era where social networks and peer groups have become increasingly diverse and influential.

The problem this project addresses is: How can we accurately identify whether a person’s political alignment is primarily influenced by family or peers, using multidimensional social and psychological factors? Furthermore, it seeks to explore which specific aspects of social interaction—such as discussion frequency, emotional tone, social pressure, and approval importance—serve as the strongest predictors of political influence from these groups.

By leveraging machine learning techniques on comprehensive survey data, this study aims to fill this gap by providing a robust, data-driven classification model. The results will enhance our understanding of the social blueprint underlying political identity and contribute to more effective strategies for fostering independent political thought and engagement.

3.1 Objectives

To identify and quantify the relative influence of family and peers on an individual’s political alignment using survey data.

1. **To analyze the role of social interaction factors**—such as frequency of political discussions, emotional tone, perceived social pressure, and importance of social approval—in shaping political beliefs within family and peer contexts.

2. **To develop and validate a machine learning classification model** that predicts whether a person’s political alignment is primarily influenced by family or peers based on multidimensional social and psychological variables.

3. **To examine demographic and individual factors** (e.g., age, gender, educational level, personal confidence) that moderate the strength of family and peer influence on political identity.

4. **To contribute empirical insights into the social mechanisms** underlying political identity formation, informing strategies to promote critical political self-reflection and independent political thought.

4. Methodology

Detailed Methodology Summary

Methodology-1.png

1. Data Acquisition

- We collected original data through surveys distributed both online and offline (paper-based).
- These surveys targeted individuals considered to be within the youth demographic, conducted inside and outside the university.
- The data gathered was carefully selected to align with the study's objectives.

2. Data Preprocessing

- Data cleaning involved removing inconsistencies, missing values, and erroneous entries.

- Data transformation included normalization or scaling of variables to prepare for modeling.
- Feature selection was applied to identify the most relevant predictors.

3. Exploratory Data Analysis (EDA)

- Visual and statistical methods were employed to understand data distributions, detect outliers, and identify patterns or correlations.
- Summary statistics and plots helped gain insights into the dataset.

4. Model Development

- Several machine learning models were developed, including baseline and more complex algorithms.
- The dataset was split into training and testing sets, commonly at an 80%-20% ratio.
- Cross-validation techniques ensured model robustness.
- Hyperparameter tuning optimized model performance.

5. Model Evaluation

- Evaluation metrics such as accuracy, precision, recall, F1 score, and confusion matrices were calculated.
- Feature importance analyses were conducted to interpret model decisions.
- Model explainability tools (e.g., SHAP values) provided deeper insights.

6. Results and Deployment

- The best performing model was selected based on evaluation metrics.
- Results were compiled and documented in the report.
- Potential deployment or application areas were discussed.

4.1 Data Collection

The dataset used in this study was gathered through a carefully designed survey targeting youth populations both within and outside the university community. This mixed-method approach combined **online surveys** and **paper-based questionnaires** to ensure wider reach and inclusion of participants with varying access to technology.

Survey Design and Distribution

The survey instrument consisted of 39 questions designed to capture various dimensions of political identity, social influences, and demographic background. The questions were crafted to probe multiple aspects of political socialization, such as personal political views, family and peer influences, frequency and intensity of political discussions, emotional tone during political interactions, and self-reflection on political beliefs.

The survey was disseminated via two primary channels:

- **Online Surveys:** Shared through university mailing lists, social media groups, and youth-oriented forums, allowing convenient and timely data collection from tech-savvy respondents.
- **Offline Paper Surveys:** Distributed in-person within the university premises and nearby community centers to include participants with limited internet access or preference for physical questionnaires.

Before any preprocessing or cleaning, our dataset consisted of **723 entries** with **39 columns** representing survey responses from youth participants inside and outside the university. Below is a summary of the dataset structure:

Feature Count	723 entries (0 to 722)
Columns	39
Data Types	34 object, 3 int64, 2 float64

Many columns correspond to answers from specific survey questions. Below is a breakdown of the types of variables collected:

Ordinal Columns

These columns represent responses on an ordered scale, reflecting intensity or frequency:

Column	Possible Responses
FamilyDiscussionInterest	not at all, rarely, sometimes, often, very well
PeerDiscussionInterest	not at all, rarely, sometimes, often, very well
FamilyInfluenceStrength	none, slight, moderate, strong
PeerInfluenceStrength	none, slight, moderate, strong
PersonalIndentityStrength	weak, moderate, strong
ParentApprovalLevel	strongly disagree, disagree, neutral, agree, strongly agree
PeerApprovalLevel	strongly disagree, disagree, neutral, agree, strongly agree
PersonalConfidenceLevel	low, medium, high
political_self_reflection_freq_family	never, rarely, sometimes, often, always
political_self_reflection_freq_peers	never, rarely, sometimes, often, always
family_approval_importance_politics	not at all, a little, somewhat, very much, absolutely
peer_approval_importance_politics	not at all, a little, somewhat, very much, absolutely
political_identity_conflict	never, rarely, sometimes, often, always

Column	Possible Responses
importance_fam_political_alignment	not important, slightly important, moderately important, very important, extremely important
importance_peer_political_alignment	not important, slightly important, moderately important, very important, extremely important

Nominal Columns

These columns represent categorical data without intrinsic ordering, often yes/no or descriptive answers:

- gender
- educational_level
- OwnPoliticalView
- FamilyPoliticalView
- PeerPoliticalView
- ListenToFamilyDiscussion
- ListenToPeerDiscussion
- FamilyEmotionalTone
- PeerEmotionalTone
- family_attempt_political_influence
- peer_attempt_political_influence
- age_political_awareness_onset
- FamilyPressure
- PeerPressure
- Political_Alignment

Dataset Feature Explanation Table :

Feature Name	Corresponding Question	Description & Relevance
gender	What is your gender?	Demographic variable to analyze political views and influences by gender.
age	How old are you?	Age is key to understand developmental differences in political identity and awareness onset.
educational_level	What is your highest educational attainment?	Education level can influence political awareness and opinion formation.
OwnPoliticalView	What is your own political view?	Captures personal political stance, central to understanding identity.

Feature Name	Corresponding Question	Description & Relevance
FamilyPoliticalView	What is your family's political view?	Family views often influence an individual’s political alignment.
PeerPoliticalView	What is the political view of your peers?	Peer influence plays a major role in youth political identity formation.
ListenToFamilyDiscussion	Do you listen to family political discussions?	Measures exposure to family political dialogue, affecting influence strength.
ListenToPeerDiscussion	Do you listen to political discussions among peers?	Measures exposure to peer political discussions.
FamilyDiscussionInterest	How interested are you in family political discussions?	Gauges engagement level with family politics.
PeerDiscussionInterest	How interested are you in peer political discussions?	Gauges engagement level with peer political discussions.
FamilyInfluenceStrength	How strong is your family’s influence on your politics?	Captures subjective perception of family political influence strength.
PeerInfluenceStrength	How strong is your peers’ influence on your politics?	Captures subjective perception of peer political influence strength.
FamilyEmotionalTone	What is the emotional tone of family political discussions?	Provides context on the quality of family political engagement.
PeerEmotionalTone	What is the emotional tone of peer political discussions?	Provides context on the quality of peer political engagement.
FamilyEngagement	Numeric measure of family engagement	Quantifies family involvement in political socialization.
PeerEngagement	Numeric measure of peer engagement	Quantifies peer involvement in political socialization.
PersonalIndentityStrength	How strong do you feel your political identity is?	Measures political identity strength, crucial for understanding self-concept development.
ParentApprovalLevel	Level of agreement with parents on political views	Reflects the extent of political approval or conflict with parents.
PeerApprovalLevel	Level of agreement with peers on political views	Reflects the extent of political approval or conflict with peers.

Feature Name	Corresponding Question	Description & Relevance
PersonalConfidenceLevel	How confident are you about your political views?	Indicates certainty and stability in political beliefs.
PoliticalIssues	What political issues do you care about?	Captures topical interests shaping political attitudes.
political_self_reflection_freq_family	Frequency of political self-reflection influenced by family	Indicates introspective engagement influenced by family context.
political_self_reflection_freq_peers	Frequency of political self-reflection influenced by peers	Indicates introspective engagement influenced by peer context.
family_attempt_political_influence	Has family tried to influence your political views?	Measures direct political influence attempts by family.
peer_attempt_political_influence	Has peer group tried to influence your political views?	Measures direct political influence attempts by peers.
age_political_awareness_onset	At what age did you become politically aware?	Helps correlate political awareness timing with social influences.
family_approval_importance_politics	How important is family approval in your politics?	Captures value placed on family's political acceptance.
peer_approval_importance_politics	How important is peer approval in your politics?	Captures value placed on peer group's political acceptance.
FamilyPressure	Do you feel political pressure from your family?	Measures perceived political pressure from family.
PeerPressure	Do you feel political pressure from your peers?	Measures perceived political pressure from peers.
political_identity_conflict	Frequency of conflicts regarding your political identity	Gauges internal/external conflicts related to political self-concept.
importance_fam_political_alignment	How important is political alignment with family?	Measures importance of shared political views with family.
importance_peer_political_alignment	How important is political alignment with peers?	Measures importance of shared political views with peers.
Political_Alignment	What is your overall political alignment?	The main target variable representing overall political stance.

This detailed feature explanation contextualizes each survey question and highlights its relevance to the study of youth political identity and social influences.

4.2 Data Cleaning

In [3]:

```
# --------------------
#    Data Handling
# --------------------
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.base import clone

# --------------------
#    Data Visualization
# --------------------
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from tqdm import tqdm  # Progress bar for loops and tasks visualization

# --------------------
#    Text Preprocessing & NLP
# --------------------
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import spacy
from textblob import TextBlob
from symspellpy import SymSpell
from rake_nltk import Rake
from keybert import KeyBERT
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder

# --------------------
#    Feature Engineering & Extraction
# --------------------
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MultiLabelBinariz
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.compose import ColumnTransformer

# --------------------
#    Model Selection & Evaluation
# --------------------
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSe
from sklearn.metrics import accuracy_score
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score, crc
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

# --------------------
#    Machine Learning Models
# --------------------
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
# ------------------------
#  Load the datasets
# -------------------------


online  = pd.read_csv('online_responses.csv')
offline = pd.read_csv('offline_responses.csv')
```

In [5]:

```
# Combine and fill missing columns with NaN (if one dataset has columns the othe
data = pd.concat([online, offline], axis=0, ignore_index=True, sort=False)
```

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 723 entries, 0 to 722
Data columns (total 39 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   timestamp                            643 non-null    object 
 1   consent                              253 non-null    object 
 2   name                                 213 non-null    object 
 3   gender                               723 non-null    object 
 4   age                                  723 non-null    int64  
 5   educational_level                    723 non-null    object 
 6   OwnPoliticalView                     723 non-null    object 
 7   FamilyPoliticalView                  723 non-null    object 
 8   PeerPoliticalView                    723 non-null    object 
 9   ListenToFamilyDiscussion             723 non-null    object 
 10  ListenToPeerDiscussion               723 non-null    object 
 11  FamilyDiscussionInterest             723 non-null    object 
 12  PeerDiscussionInterest               722 non-null    object 
 13  FamilyInfluenceStrength              643 non-null    object 
 14  PeerInfluenceStrength                669 non-null    object 
 15  FamilyEmotionalTone                  723 non-null    object 
 16  PeerEmotionalTone                    722 non-null    object 
 17  FamilyEngagement                     723 non-null    int64  
 18  PeerEngagement                       723 non-null    int64  
 19  PersonalIndentityStrength            723 non-null    object 
 20  ParentApprovalLevel                  721 non-null    object 
 21  PeerApprovalLevel                    720 non-null    object 
 22  PersonalConfidenceLevel              720 non-null    object 
 23  PoliticalIssues                      719 non-null    object 
 24  political_self_reflection_freq_family  723 non-null    object 
 25  political_self_reflection_freq_peers  721 non-null    object 
 26  family_attempt_political_influence   718 non-null    object 
 27  peer_attempt_political_influence     722 non-null    object 
 28  age_political_awareness_onset        722 non-null    object 
 29  family_approval_importance_politics  723 non-null    object 
 30  peer_approval_importance_politics    723 non-null    object 
 31  FamilyPressure                       723 non-null    object 
 32  PeerPressure                         723 non-null    object 
 33  political_identity_conflict          723 non-null    object 
 34  importance_fam_political_alignment   723 non-null    object 
 35  importance_peer_political_alignment  722 non-null    object 
 36  Political_Alignment                  723 non-null    object 
 37  Column 38                            0 non-null      float64
 38  Column 39                            0 non-null      float64
dtypes: float64(2), int64(3), object(34)
memory usage: 220.4+ KB
```

In [7]:

```
data.describe()
```

Out[7]:

	age	FamilyEngagement	PeerEngagement	Column 38	Column 39
count	723.000000	723.000000	723.000000	0.0	0.0
mean	21.085754	1.962656	2.077455	NaN	NaN
std	7.750854	1.270806	1.281480	NaN	NaN
min	13.000000	0.000000	0.000000	NaN	NaN
25%	19.000000	1.000000	1.000000	NaN	NaN
50%	20.000000	2.000000	2.000000	NaN	NaN
75%	22.000000	3.000000	3.000000	NaN	NaN
max	212.000000	5.000000	5.000000	NaN	NaN

In [8]:

```
data.describe(include=object)
```

Out[8]:

	timestamp	consent	name	gender	educational_level	OwnPoliticalView	Fami
count	643	253	213	723	723	723	
unique	243	1	211	11	4	6	
top	5/8/2025	Yes, I consent	Mary	F	College	Liberal	
freq	135	253	2	185	612	452	

4 rows × 34 columns

4.3 Data Preparation

In [9]:

```
# Remove unnecessary columns
columns_to_drop = ['timestamp', 'consent', 'name', 'Column 38', 'Column 39']
data = data.drop(columns=columns_to_drop, errors='ignore')
```

In [10]:

```
# Clean column names
data.columns = data.columns.str.strip()

# Clean string values
for col in data.select_dtypes(include='object'):
    data[col] = data[col].str.strip()
```

In [11]:

```
# Define columns where string "None" is meaningful and should be preserved
meaningful_none_columns = ['FamilyInfluenceStrength', 'PeerInfluenceStrength']

# Clean column names
data.columns = data.columns.str.strip()

# Strip leading/trailing whitespace for all string columns
for col in data.select_dtypes(include='object'):
    data[col] = data[col].str.strip()
```

```
# Replace unwanted null-like strings *only in non-protected columns*
for col in data.columns:
    if col not in meaningful_none_columns:
        data[col] = data[col].replace(['', 'None', 'none', 'NONE'], np.nan)

# --- CRITICAL ---
# Forcefully re-cast protected columns so they *retain* string "None"
# This undoes any accidental conversion from earlier steps
for col in meaningful_none_columns:
    data[col] = np.where(data[col].isna(), 'None', data[col].astype(str))
```

In [12]:

```
data.isnull().sum()
```

Out[12]:

	0
gender	0
age	0
educational_level	0
OwnPoliticalView	0
FamilyPoliticalView	0
PeerPoliticalView	0
ListenToFamilyDiscussion	0
ListenToPeerDiscussion	0
FamilyDiscussionInterest	0
PeerDiscussionInterest	1
FamilyInfluenceStrength	0
PeerInfluenceStrength	0
FamilyEmotionalTone	0
PeerEmotionalTone	1
FamilyEngagement	0
PeerEngagement	0
PersonalIndentityStrength	0
ParentApprovalLevel	2
PeerApprovalLevel	3
PersonalConfidenceLevel	3
PoliticalIssues	7
political_self_reflection_freq_family	0
political_self_reflection_freq_peers	2
family_attempt_political_influence	5
peer_attempt_political_influence	1
age_political_awareness_onset	1
family_approval_importance_politics	0
peer_approval_importance_politics	0
FamilyPressure	0
PeerPressure	0
political_identity_conflict	0
importance_fam_political_alignment	0

	0
importance_peer_political_alignment	1
Political_Alignment	0

dtype: int64

In [13]:

```
data.dropna(inplace=True)
```

In [14]:

```
# Get all object columns except 'PoliticalIssues'
object_cols = data.select_dtypes(include='object').columns.drop('PoliticalIssues

# Display unique values for each
for col in object_cols:
    print(f"\nColumn: {col}")
    print(sorted(data[col].dropna().unique()))
```

```
Column: gender
['F', 'FEMALE', 'Female', 'M', 'MALE', 'Male', 'Malr', 'Msle', 'f', 'm']

Column: educational_level
['College', 'High School', 'college']

Column: OwnPoliticalView
['Conservative', 'Liberal', 'MOderate', 'Moderate', 'Not Sure', 'moderate']

Column: FamilyPoliticalView
['COnservative', 'Conservative', 'Liberal', 'MOderate', 'Moderate', 'Not Sure',
'conservative']

Column: PeerPoliticalView
['COnservative', 'Conservative', 'Liberal', 'Moderate', 'Not Sure', 'Not sure',
'moderate']

Column: ListenToFamilyDiscussion
['No', 'Yes', 'yes']

Column: ListenToPeerDiscussion
['No', 'Yes', 'no', 'yes']

Column: FamilyDiscussionInterest
['Not at All', 'Often', 'Rarely', 'Sometime', 'Sometimea', 'Sometimes', 'VEry Wel
l', 'Very Well', 'often', 'sometime', 'sometimes']

Column: PeerDiscussionInterest
['Not at All', 'Often', 'Rarely', 'Sometime', 'Sometimes', 'Very Well', 'often',
'sometimes', 'very Well']

Column: FamilyInfluenceStrength
['MOderate', 'Moderate', 'None', 'SLight', 'Slight', 'Strong', 'moderate', 'sligh
t']

Column: PeerInfluenceStrength
['MOderate', 'Moderate', 'None', 'SLight', 'Slight', 'Strong', 'slight']

Column: FamilyEmotionalTone
['Negative', 'Neutral', 'POsitive', 'Positive', 'Slight', 'neutral']

Column: PeerEmotionalTone
['Negative', 'Neitral', 'Neutral', 'POsitive', 'Positive', 'neutral']

Column: PersonalIndentityStrength
['MOderate', 'Moderate', 'Strong', 'Weak', '`Weak', 'moderate', 'strong']

Column: ParentApprovalLevel
['Agree', 'Disagree', 'Neeutral', 'Neurtral', 'Neutral', 'Strongly Agree', 'Stron
gly Disagree', 'agree', 'disagree', 'strongly agree']

Column: PeerApprovalLevel
['Agree', 'Disagree', 'Neutral', 'Strongly Agree', 'Strongly Disagree', 'agree',
'disagree', 'neutral']

Column: PersonalConfidenceLevel
['HIgh', 'High', 'Low', 'Medium', 'high', 'low']

Column: political_self_reflection_freq_family
['Always', 'Never', 'Often', 'Rarely', 'Sometimes', 'Somtimes', 'always', 'neve
r', 'often', 'rarely', 'sometimes']

Column: political_self_reflection_freq_peers
['Always', 'Never', 'Often', 'Rarely', 'Sometimes', 'always', 'never', 'rarely',
'sometimes']

Column: family_attempt_political_influence
['No', 'Yes', 'no', 'yes']

Column: peer_attempt_political_influence
['No', 'Yes', 'no', 'yes']

Column: age_political_awareness_onset
['Childhood', 'Early Adulthood', 'REecently', 'Recently', 'Reecently', 'Teen Year
s', 'recently']

Column: family_approval_importance_politics
['A Little', 'A little', 'Absolutely', 'Not at All', 'Not at all', 'SOmewhat', 'S
omewhat', 'VEry Much', 'Very Much', 'a little', 'not at all']

Column: peer_approval_importance_politics
['A Little', 'A little', 'Absolutely', 'Not at All', 'Not at all', 'SOmewhat', 'S
omewhat', 'Very Much', 'a little', 'absolutely', 'not at all', 'somewhat', 'very
much']

Column: FamilyPressure
['No', 'Yes', 'no', 'y']

Column: PeerPressure
['No', 'Yes', 'no']

Column: political_identity_conflict
['Always', 'Never', 'Often', 'Rarely', 'Sometime', 'Sometimes', 'always', 'neve
r', 'rarely', 'sometimes']

Column: importance_fam_political_alignment
['Etremely Important', 'Extremely Important', 'MOderately Important', 'Moderately
Important', 'Moerately Important', 'Not Important', 'Slightly Important', 'Very I
mportant', 'Very Importat', 'slightly important', 'very Important', 'very importa
nt']

Column: importance_peer_political_alignment
['Extremely Important', 'MOderately Important', 'Moderately Important', 'NOt Impo
rtant', 'Not Important', 'Slightly Important', 'Very Important', 'moderately Impo
rtant', 'slightly Important', 'slightly important']

Column: Political_Alignment
['FAmily', 'Family', 'Peers', 'peers']
```

In [15]:

```python
for col in object_cols:
    data[col] = data[col].str.strip().str.lower()
```

In [16]:

```python
# Gender
data['gender'] = data['gender'].replace({
    'f': 'female', 'female': 'female', 'female': 'female',
    'm': 'male', 'male': 'male', 'msle': 'male', 'malr': 'male'
})

# Educational level
data['educational_level'] = data['educational_level'].replace({
    'college': 'college', 'high school': 'high school'
})

# Political views
fix_political_view = {
    'conservative': 'conservative',
    'liberal': 'liberal',
    'moderate': 'moderate',
    'not sure': 'not sure'
}
for col in ['OwnPoliticalView', 'FamilyPoliticalView', 'PeerPoliticalView']:
    data[col] = data[col].replace(fix_political_view)

# Yes/No standardization
yes_no_map = {
    'yes': 'yes', 'no': 'no', 'y': 'yes', 'n': 'no'
}
for col in ['ListenToFamilyDiscussion', 'ListenToPeerDiscussion', 'family_attemp
            'peer_attempt_political_influence', 'FamilyPressure', 'PeerPressure'
    data[col] = data[col].replace(yes_no_map)

# Confidence level
data['PersonalConfidenceLevel'] = data['PersonalConfidenceLevel'].replace({
    'high': 'high', 'low': 'low', 'medium': 'medium'
})

# Identity conflict + reflection frequency
freq_map = {
    'always': 'always', 'often': 'often', 'sometimes': 'sometimes', 'somtimes':
    'rarely': 'rarely', 'never': 'never', 'sometime': 'sometimes', 'sometimea':
}
conflict_cols = ['political_self_reflection_freq_family', 'political_self_reflec
                 'political_identity_conflict', 'FamilyDiscussionInterest', 'Pee
for col in conflict_cols:
    data[col] = data[col].replace(freq_map)

# Influence Strength
influence_map = {
    'none': 'none', 'slight': 'slight', 'moderate': 'moderate', 'strong': 'stron
}
for col in ['FamilyInfluenceStrength', 'PeerInfluenceStrength']:
    data[col] = data[col].replace(influence_map)

# Emotional tone
tone_map = {
    'positive': 'positive', 'negative': 'negative', 'neutral': 'neutral', 'neitr
}
for col in ['FamilyEmotionalTone', 'PeerEmotionalTone']:
    data[col] = data[col].replace(tone_map)

# Identity strength
identity_map = {
    'strong': 'strong', 'moderate': 'moderate', 'weak': 'weak', '`weak': 'weak'
}
data['PersonalIndentityStrength'] = data['PersonalIndentityStrength'].replace(id

# Approval level
approval_map = {
    'agree': 'agree', 'disagree': 'disagree', 'neutral': 'neutral',
    'strongly agree': 'strongly agree', 'strongly disagree': 'strongly disagree'
    'neeutral': 'neutral', 'neurtral': 'neutral', 'neitral' : 'neutral'
}
for col in ['ParentApprovalLevel', 'PeerApprovalLevel']:
    data[col] = data[col].replace(approval_map)

# Onset of political awareness
data['age_political_awareness_onset'] = data['age_political_awareness_onset'].re
    'childhood': 'childhood',
    'teen years': 'teen years',
    'early adulthood': 'early adulthood',
    'recently': 'recently', 'reecently': 'recently', 'reecently': 'recently'
})

# Importance of approval
importance_map = {
    'a little': 'a little',
    'not at all': 'not at all',
    'somewhat': 'somewhat',
    'absolutely': 'absolutely',
    'very much': 'very much'
}
for col in ['family_approval_importance_politics', 'peer_approval_importance_pol
    data[col] = data[col].replace(importance_map)

# Alignment importance
alignment_map = {
    'not important': 'not important',
    'slightly important': 'slightly important',
    'moderately important': 'moderately important',
    'extremely important': 'extremely important',
    'very important': 'very important',
    'very importat': 'very important',
    'moerately important': 'moderately important',
    'etremely important' : 'extremely important',
    'sometimea' : 'sometimes',
    'neitral' : 'neutral'
}
for col in ['importance_fam_political_alignment', 'importance_peer_political_ali
    data[col] = data[col].replace(alignment_map)

# Political alignment
data['Political_Alignment'] = data['Political_Alignment'].replace({
    'family': 'family', 'peers': 'peers', 'faimly': 'family'
})
```

In [17]:

```
for col in object_cols:
    print(f"\nColumn: {col}")
    print(sorted(data[col].dropna().unique()))
```

```
Column: gender
['female', 'male']

Column: educational_level
['college', 'high school']

Column: OwnPoliticalView
['conservative', 'liberal', 'moderate', 'not sure']

Column: FamilyPoliticalView
['conservative', 'liberal', 'moderate', 'not sure']

Column: PeerPoliticalView
['conservative', 'liberal', 'moderate', 'not sure']

Column: ListenToFamilyDiscussion
['no', 'yes']

Column: ListenToPeerDiscussion
['no', 'yes']

Column: FamilyDiscussionInterest
['not at all', 'often', 'rarely', 'sometimes', 'very well']

Column: PeerDiscussionInterest
['not at all', 'often', 'rarely', 'sometimes', 'very well']

Column: FamilyInfluenceStrength
['moderate', 'none', 'slight', 'strong']

Column: PeerInfluenceStrength
['moderate', 'none', 'slight', 'strong']

Column: FamilyEmotionalTone
['negative', 'neutral', 'positive', 'slight']

Column: PeerEmotionalTone
['negative', 'neutral', 'positive']

Column: PersonalIndentityStrength
['moderate', 'strong', 'weak']

Column: ParentApprovalLevel
['agree', 'disagree', 'neutral', 'strongly agree', 'strongly disagree']

Column: PeerApprovalLevel
['agree', 'disagree', 'neutral', 'strongly agree', 'strongly disagree']

Column: PersonalConfidenceLevel
['high', 'low', 'medium']

Column: political_self_reflection_freq_family
['always', 'never', 'often', 'rarely', 'sometimes']

Column: political_self_reflection_freq_peers
['always', 'never', 'often', 'rarely', 'sometimes']

Column: family_attempt_political_influence
['no', 'yes']
```

```
Column: peer_attempt_political_influence
['no', 'yes']

Column: age_political_awareness_onset
['childhood', 'early adulthood', 'recently', 'teen years']

Column: family_approval_importance_politics
['a little', 'absolutely', 'not at all', 'somewhat', 'very much']

Column: peer_approval_importance_politics
['a little', 'absolutely', 'not at all', 'somewhat', 'very much']

Column: FamilyPressure
['no', 'yes']

Column: PeerPressure
['no', 'yes']

Column: political_identity_conflict
['always', 'never', 'often', 'rarely', 'sometimes']

Column: importance_fam_political_alignment
['extremely important', 'moderately important', 'not important', 'slightly import
ant', 'very important']

Column: importance_peer_political_alignment
['extremely important', 'moderately important', 'not important', 'slightly import
ant', 'very important']

Column: Political_Alignment
['family', 'peers']
```

Encoding Categorical Data

In [18]:

```
# Define ordinal columns with their category orders
ordinal_columns = {
    'FamilyDiscussionInterest': ['not at all', 'rarely', 'sometimes', 'often', '
    'PeerDiscussionInterest': ['not at all', 'rarely', 'sometimes', 'often', 've
    'FamilyInfluenceStrength': ['none', 'slight', 'moderate', 'strong'],
    'PeerInfluenceStrength': ['none', 'slight', 'moderate', 'strong'],
    'PersonalIndentityStrength': ['weak', 'moderate', 'strong'],
    'ParentApprovalLevel': ['strongly disagree', 'disagree', 'neutral', 'agree',
    'PeerApprovalLevel': ['strongly disagree', 'disagree', 'neutral', 'agree', '
    'PersonalConfidenceLevel': ['low', 'medium', 'high'],
    'political_self_reflection_freq_family': ['never', 'rarely', 'sometimes', 'c
    'political_self_reflection_freq_peers': ['never', 'rarely', 'sometimes', 'of
    'family_approval_importance_politics': ['not at all', 'a little', 'somewhat'
    'peer_approval_importance_politics': ['not at all', 'a little', 'somewhat',
    'political_identity_conflict': ['never', 'rarely', 'sometimes', 'often', 'al
    'importance_fam_political_alignment': ['not important', 'slightly important'
    'importance_peer_political_alignment': ['not important', 'slightly important
}

# Define nominal columns for one-hot encoding
nominal_columns = [
    'gender',
    'educational_level',
    'OwnPoliticalView',
    'FamilyPoliticalView',
    'PeerPoliticalView',
    'ListenToFamilyDiscussion',
    'ListenToPeerDiscussion',
    'FamilyEmotionalTone',
    'PeerEmotionalTone',
    'family_attempt_political_influence',
    'peer_attempt_political_influence',
    'age_political_awareness_onset',
    'FamilyPressure',
    'PeerPressure',
    'Political_Alignment'
]

# Standardize case first
all_categorical = list(ordinal_columns.keys()) + nominal_columns
for col in all_categorical:
    data[col] = data[col].str.lower()

# Apply ordinal encoding
for col, categories in ordinal_columns.items():
    oe = OrdinalEncoder(categories=[categories])
    data[col] = oe.fit_transform(data[[col]])

# Apply one-hot encoding to nominal variables
data = pd.get_dummies(data, columns=nominal_columns)
```

Text Cleaning

In [19]:

```
print(data['PoliticalIssues'].unique())
```

```
['election' 'Corruption, LGBTQ+ Rights, Education, Digitalization'
 'Corruption' 'healthcare access' 'Corruption, election, red tagging'
 'corruption' 'Poverty'
 'Corruption, West Philippine Sea, China, Educational Crisis, Inflation'
 "corruption and some sexual abuse's" 'about election'
 'Corruption, inflation and poverty' 'walay mapili puro corrupt'
 'About corruptions, donation (ayuda)'
 'education system, abortion, divorce' 'political dynasty'
 'vote buying, corruption' 'election and corruption'
 'corruption, legislation, poverty, power trip' 'Election and corruption'
 'Human rights' 'corruption and bribe' 'everything that inloves politics'
 'Inequality' 'social justice, education, and economic equality.'
 'Curroption' 'Corruption, Family Dynasty' 'Corruption and Vote Buying'
 'Election'
 'I’m most passionate about corruption, because it affects almost every aspect of
society and stops real progress. I also care about fair elections, since they ref
lect the voice of the people. Education is important to me too, because it shapes
the future. And I value human rights, because everyone deserves to be treated wit
h fairness and respect.'
 'Election, Corruption, Political Dynasties, Qualifications of Elected Politician
s'
 'election, political dynasty' 'Economic issues like income inequality'
 'election n corruption'
 'Corruption and the election, specially sa issue ni PRRD'
 'corruption, poverty, and gender inequality' 'corrupt' 'anti corrupt'
 'corruption, upcoming election, political dynasty'
 'Cost of Healthcare & Medication, Food Safety & Quality and Financial Reform'
 'Education, agriculture, urban development' 'Political dynasty'
 'Issues about only selected people receiving assistance from the Government.'
 'correuption, politca dynasty, vote-buying, lgbtq community policies'
 'Choosing the right person to sit in the government.'
 'corruption/betrayal' 'Electiom'
 'As a political scientist and currently a Law School student, I am more passiona
te about discussing issues that revolves within the political arena of the countr
y. If there are two things that I am more interested with, that’s – ending insurg
ency, and fight against the endemic corruption in the government.'
 'equality' 'Corruption Specially not supporting the agricultural section'
 'I am passionate about promoting policies that foster inclusive economic growth,
ensuring equitable access to quality education and healthcare, and protecting the
environment for future generations.  These are interconnected issues that, when a
ddressed effectively, can significantly improve the well-being of communities.'
 'Agriculture, education, technology'
 'Corruption, education, agriculture, women empowerment, LGBTQIA+ rights'
 'election, corruption' 'territorial dispute, digitalization'
 'corruption, environment' 'bad governance, corruption'
 'political dynasty, bad governance,' 'bad governance, Economic crisis'
 'economy related' "Country's Utang" 'political halalan'
 'Economic related, Health care related'
 'health care related, economic crisis'
 'Bad governance, Academic achievement of the congress' 'health care'
 'corruption, red tagging' 'election, vote-buying' 'law' 'eelection'
 'duterte' 'election, drug war' 'Countering Violent Extremism'
 'war on drugs'
 'extrajudicial killings, attacks against political activists'
 'Anti-Corruption' 'elecion, red tagging'
 'anti-corruption, political dynasty' 'Vote buying' 'education' 'wala bay'
 'public money is misused.' 'Political Dynasty' 'equal rights'
 'freedom of speech'
 "education reform, I think every student deserves equal access to quality educat
ion regardless of where they live. I've seen how different schools can be dependi
```

```
ng on their location or funding, and I feel that's unfair."
 'economic opportunity' 'corruption, economic concerns'
 'Political dynasty, Economic concerns'
 'Government ignoring some issues of the country'
 "The senates don't have a strong background towards academic"
 "The way government handle it's people"
 'the government ignoring the farmers issues' 'Farming industry'
 'Food industry issues' 'Food security issues' 'Farmer issues'
 'national elections' 'Food Security' 'Food Insecurity'
 'Economic related issues' 'bad governance' 'Agriculture issues'
 'Politic Dynasty' 'Family Dynasty in the congress'
 'Corrupt government officials' 'Food issues' 'Bad governance' 'Farming'
 'People handling' 'Foo insecurity' 'Election, vote buying'
 'Political dynasty, vote buying' 'Vote buying, food security'
 'Educational reform'
 'Corruption in all department and levels of Philippine goverment.'
 'Program for birth control to less fortunate' 'corruption, red-tagging'
 'election, duterte killings' 'election, inflation'
 'Corruption in government agencies'
 'Red-tagging of activists and critics'
 'Padrino system / Political dynasties'
 'Lack of transparency in public funds'
 'Weaponization of libel and cyberlibel against journalists'
 'Election fraud' 'Vote-buying'
 'Delayed government response during crises' 'Political dynasties'
 'Slow justice system' 'Selective justice' 'Extrajudicial killings'
 'Marcos' 'EJK-linked officials. corruption, election'
 'Wealth inequality, election'
 'Same families running for office repeatedly'
 'Red-tagging of student activists' 'Lack of protection for LGBTQ+ rights'
 'Harassment of journalists' 'High inflation / cost of living'
 'Low minimum wage'
 'Insulting or out-of-touch remarks from politicians / election/ corruption'
 'Stigma against activism / Online troll armies'
 'Homophobia and discrimination' 'Rewriting of Martial Law'
 'Insulting or out-of-touch remarks from politicians/ corruption'
 'Influencer and celebrity candidates with zero public service background'
 'corruption/ election / weak institutions' 'Corruption, injustices'
 'The lack of requirements in running for such political position.'
 'All possible aspects that need to change.'
 'Corruption, bad governance, and all' 'Elections'
 'Bad governance,political dynasties' 'Corruption, Tax' 'Bad Governance'
 'Poverty, Goverment Initiatives'
 'Election, Corruption, Poverty, Programs'
 'Election, Corruption, Inflation' 'bad Governance'
 'Corruption, Political Dynasty, Bad Governance, Accountability, Graft, etc...'
 'Corruption, Political Dynasty, Bad Governance, Accountability, Graft, Political
Credentials'
 'Corruption, Bad Governance Injustice' 'Legislation'
 'Corruption and Bad Governance, Political Dynasty'
 'Corruption/Bad Governance'
 'Election, corruption,campaign,program,platforms' 'Economic Challenges'
 'Corruption and Bad Governance' 'Government issue' 'Human Rights'
 'Election and Poverty' 'Corruption,good governance,divorce'
 'Freedom of press' 'Corruption,war on drugs; inflation'
 'Corruption,bad governance,progress&reform,acceptance&dialogue in all forms,peac
e'
 'All of the above' 'Corruption,political dynasty' 'Bad Government'
 'Corruption&bad governance' 'Corruption,bad governance and vote buying'
 'Bad governance, lack of transparenting, etc.'
 'Elections, bad governance; being a political' 'Abortion'
 'Corruption, vote buying, Bad Governance' 'Elections of vote buying'
 'Incompetent Leaders, Bad Governance' 'none actually'
 'Less Qualified Lawmakers' 'Election, Poverty, Security'
 'Corruption, Political Dynasties' 'passionate with all political issues'
 'Elections, Corruption' 'Corruption, Bad Governance' 'Corrupt Governance'
 'Political Dynasty, Tax evader, stuck in their old tradition beliefs'
 'Corruotions, Elections' 'Lack of support for farmers and fisherfolk'
 'Political Dynasties' 'Duterte and Marcos'
 'Massive government debt with unclear benefit to the poor' 'The Game'
 'Election, Diversity' 'Disconnection of politicians from youth realities'
 'Mining in ancestral lands' 'Lack of protection for LGBTQ+' 'harassment'
 'Elections, Corruption, Bad Governance' 'Health Assistance'
 'Environmental Issues' 'Election candidates credibility' 'LGBTQ+'
 'Low minimum wage vs rising cost of basic goods' 'Education'
 'Budget cuts in state universities'
 'isconnection of politicians from youth realities'
 'Corruption, Environmental' 'Discrimination' 'War on Drugs'
 'Poor disaster response and climate adaptation'
 'Overreliance on foreign loans' 'Election, Vote-Buying'
 'Online troll armies / influence operations' 'Loans / Election'
 'Slow progress on divorce, same-sex union, and other progressive bills'
 'Bad Governance, Corruption, Unequal Opportunities to Margenalized Sectors'
 'Governance' 'Politicians blocking or ghosting FOI'
 'Bad Roads and Corruption'
 'Corruption, Bad Governance, Political Dynasty' 'Economic and Trade'
 'Bad Governance, Corruption, Political Dynasty, Elections'
 'Election, Corruption'
 'Election, Corruption, Bad Governance, Everything Related to politics for it aff
ects everyone'
 'Corruption, Political Dynasty, Bad Governance'
 'Old politicians staying in power' 'Martial Law' 'Duterte'
 'Corruption, bad governance esp. economic inequality (farmers, laborers)'
 'corruption, unfair benefits' 'corruption and bad governance'
 'bad governance and corruption'
 'urban planning, farming funds, educational priorities' 'good governance'
 'good governance, education, corruption' 'sogie, reproductive health,'
 'same politicians running' 'elections' 'Suppression of dissent'
 'High inflation / election' 'government debt' 'loans and conrruption'
 'inflation rate' 'Budget cuts' 'lack of incentives' 'West Philippine Sea'
 'Reliance on loan' 'divorce, same-sex union laws are slow to progress'
 'Backlogs in legal cases' 'Delayed or missing ayuda during emergencies'
 'Frequent reshuffling of cabinet secretaries / no policy continuity'
 'missing ayuda' 'Attacks on indigenous education systems/ Election'
 'Poor public hospital infrastructure' 'Long queues and broken systems'
 'anti-intellectualism'
 'No clear roadmap for vocational and skills-based education'
 'Warrantless arrests during protests'
 '/ vote buying / repeating names on senate'
 'Unemployment and underemployment of college grads'
 'No full implementation of IPRA law' 'Public transport modernization'
 'Manila-centric development' 'Influencer politicians'
 'Election / Corruption' 'No space for youth policymaking'
 'lack of Political will' 'Post-election accountability'
 'Election / Corruption / Vote-Buying'
 'poor implementation of laws / misinformation'
 'Lack of baseline data for policymaking'
 'No feedback loops between citizens and government'
 'Mental health law passed / lack of initiatives/ election chaos'
 'Barangay health workers underpaid despite being frontliners'
 'Religion blocking legislation' 'Lack of Rural Area Development'
 'Unemployment / Election / Corruption in LGU'
 'No clear separation of church and state in practice'
 'National identity crisis'
 'People choosing silence over civic participation' 'Duterte Vs. marcos'
 'Promotion of fake news' 'online harassment by troll' ', modern jeepneys'
 'Politicians using vloggers instead of press briefings'
 'socmed as new misinformation hotspot'
 'Political education not being taught or prioritized in schools'
 'Some LGUs left alone'
 'Infrastructure projects done without community displacement planning'
 'manual workforce' 'Delays in national ID system rollout'
 'Outdated, manual government systems']
```

In [20]:

```python
# Cleaning function
def clean_text(text):
    if not isinstance(text, str):
        return ""

    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = re.sub(r'[^\w\s]', ' ', text)

    # Handle common misspellings
    text = re.sub(r'\bcurro?ption\b', 'corruption', text)
    text = re.sub(r'\belectiom?\b', 'election', text)
    text = re.sub(r'\beelection\b', 'election', text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    words = text.split()
    words = [word for word in words if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]

    return ' '.join(words)

# Apply cleaning
data['cleaned_issues'] = data['PoliticalIssues'].apply(clean_text)
```

In [21]:

```python
tfidf = TfidfVectorizer(max_features=100, ngram_range=(1, 2))
tfidf_matrix = tfidf.fit_transform(data['cleaned_issues'])
```

In [22]:

```python
# Define categories and their keywords
category_map = {
    'Corruption': ['corruption', 'graft', 'bribe', 'betrayal'],
    'Elections': ['election', 'vote buying', 'political dynasty', 'campaign'],
    'Governance': ['bad governance', 'good governance', 'transparency', 'account
    'Human Rights': ['human right', 'ejk', 'extrajudicial', 'killings', 'red tag
    'Economic Issues': ['poverty', 'inflation', 'economic', 'income inequality',
    'Social Issues': ['lgbtq', 'gender inequality', 'discrimination', 'same-sex'
    'Education': ['education', 'school', 'university', 'academic'],
    'Healthcare': ['healthcare', 'hospital', 'medication', 'health worker'],
    'Agriculture': ['agriculture', 'farmer', 'farming', 'fisherfolk'],
    'Environment': ['environment', 'climate', 'mining'],
    'Foreign Policy': ['west philippine sea', 'china', 'territorial dispute'],
    'Infrastructure': ['infrastructure', 'road', 'transport', 'id system'],
    'Justice System': ['justice system', 'legal', 'case backlog', 'libel'],
    'Government Services': ['ayuda', 'assistance', 'public service', 'government
}

# Function to categorize text
def categorize_text(text):
    text = clean_text(text)
    for category, keywords in category_map.items():
        if any(keyword in text for keyword in keywords):
            return category
    return 'Other'  # For uncategorized items

data['category'] = data['PoliticalIssues'].apply(categorize_text)
```

In [23]:

```
# Modified categorization to return multiple categories
def categorize_multilabel(text):
    text = clean_text(text)
    matched_categories = []
    for category, keywords in category_map.items():
        if any(keyword in text for keyword in keywords):
            matched_categories.append(category)
    return ', '.join(matched_categories) if matched_categories else 'Other'

data['categories'] = data['PoliticalIssues'].apply(categorize_multilabel)
```

In [24]:

```
print(data['categories'])
```

```
0                                  Elections
1      Corruption, Social Issues, Education
2                                 Corruption
3                                 Healthcare
4       Corruption, Elections, Human Rights
                       ...                  
718                           Justice System
719                      Government Services
720                                    Other
721                      Government Services
722                     Elections, Education
Name: categories, Length: 704, dtype: object
```

In [25]:

```
# Split the categories into lists
data['categories_split'] = data['categories'].str.split(', ')

# Get all unique categories
all_categories = set()
for sublist in data['categories_split'].dropna():
    all_categories.update(sublist)

# Create binary columns for each category
for category in all_categories:
    data[f'issue_{category.lower().replace(" ", "_")}'] = data['categories'].app
        lambda x: 1 if isinstance(x, str) and category in x else 0
    )
```

In [26]:

```
# Clean column names and standardize
column_rename = {
    'issue_corruption': 'issue_corruption',
    'issue_elections': 'issue_election',
    'issue_social_issues': 'issue_social',
    'issue_education': 'issue_education',
    'issue_healthcare': 'issue_healthcare',
    'issue_human_rights': 'issue_human_rights',
    'issue_justice_system': 'issue_justice',
    'issue_government_services': 'issue_gov_services',
    'issue_other': 'issue_other'
}

data = data.rename(columns=column_rename)
```

In [27]:

```
# Create a dictionary of one-hot encoded groups
one_hot_groups = {
    'gender': ['gender_female', 'gender_male'],
    'education': ['educational_level_college', 'educational_level_high school'],
    'own_politics': ['OwnPoliticalView_conservative', 'OwnPoliticalView_liberal'
                    'OwnPoliticalView_moderate', 'OwnPoliticalView_not sure'],
    'family_politics': ['FamilyPoliticalView_conservative', 'FamilyPoliticalView
                       'FamilyPoliticalView_moderate', 'FamilyPoliticalView_not
    'peer_politics': ['PeerPoliticalView_conservative', 'PeerPoliticalView_liber
                     'PeerPoliticalView_moderate', 'PeerPoliticalView_not sure']
    'listen_family': ['ListenToFamilyDiscussion_no', 'ListenToFamilyDiscussion_y
    'listen_peer': ['ListenToPeerDiscussion_no', 'ListenToPeerDiscussion_yes'],
    'family_emotion': ['FamilyEmotionalTone_negative', 'FamilyEmotionalTone_neut
                      'FamilyEmotionalTone_positive', 'FamilyEmotionalTone_sligh
    'peer_emotion': ['PeerEmotionalTone_negative', 'PeerEmotionalTone_neutral',
                    'PeerEmotionalTone_positive'],
    'family_influence': ['family_attempt_political_influence_no',
                        'family_attempt_political_influence_yes'],
    'peer_influence': ['peer_attempt_political_influence_no',
                      'peer_attempt_political_influence_yes'],
    'awareness_onset': ['age_political_awareness_onset_childhood',
                       'age_political_awareness_onset_early adulthood',
                       'age_political_awareness_onset_recently',
                       'age_political_awareness_onset_teen years'],
    'family_pressure': ['FamilyPressure_no', 'FamilyPressure_yes'],
    'peer_pressure': ['PeerPressure_no', 'PeerPressure_yes']
}

# For each group, keep first column and drop the others
cols_to_drop = []
for group, columns in one_hot_groups.items():
    cols_to_drop.extend(columns[1:])  # Keep first column, drop others

data_reduced = data.drop(columns=cols_to_drop)
```

In [28]:

```
# Calculate correlation matrix for remaining numeric columns
numeric_cols = data_reduced.select_dtypes(include=['float64', 'int64']).columns
corr_matrix = data_reduced[numeric_cols].corr()

# Find perfectly correlated pairs (r = ±1)
perfect_corr = []
for col in corr_matrix.columns:
    for idx in corr_matrix.index:
        if col != idx and abs(corr_matrix.loc[idx, col]) == 1:
            perfect_corr.append((col, idx))

# Remove duplicates and sort
```

```
perfect_corr = sorted(list(set(tuple(sorted(pair)) for pair in perfect_corr)))

print("Perfectly correlated pairs:")
for pair in perfect_corr:
    print(f"{pair[0]} - {pair[1]}")

# Drop one column from each perfectly correlated pair
cols_to_drop = [pair[1] for pair in perfect_corr]  # Dropping the second column
data_reduced = data_reduced.drop(columns=cols_to_drop)
```

```
Perfectly correlated pairs:
```

In [29]:

```
print("\nFinal feature count:", len(data_reduced.columns))
```

```
Final feature count: 54
```

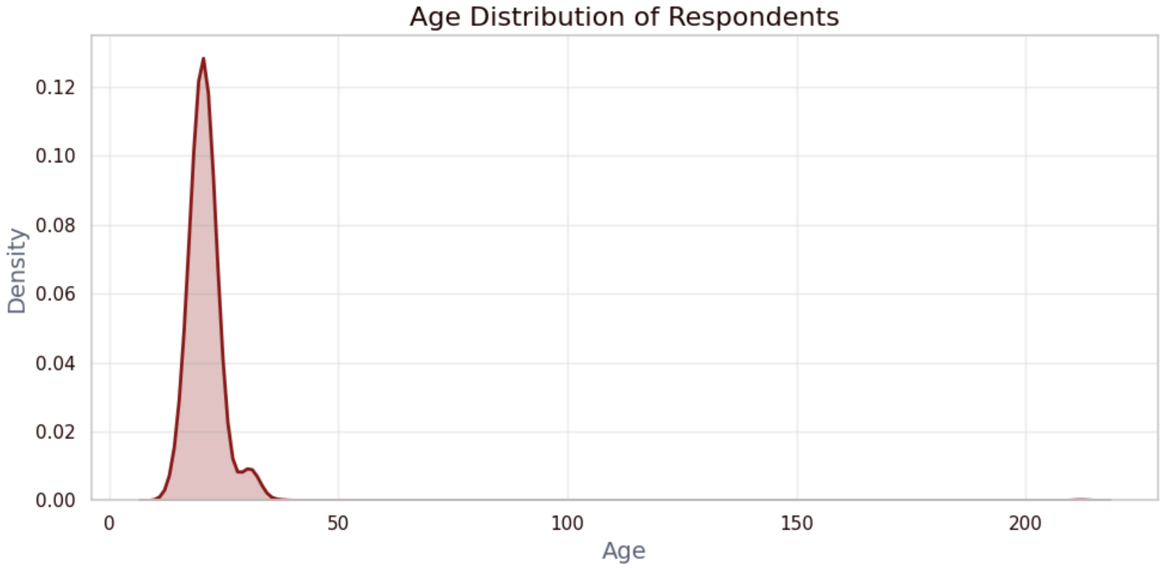
Exploratory Data Analysis

In [30]:

```
# Your color palette
palette = ['#210706', '#891D1A', '#5E657B', '#F1E6D2']
sns.set(style='whitegrid')
```

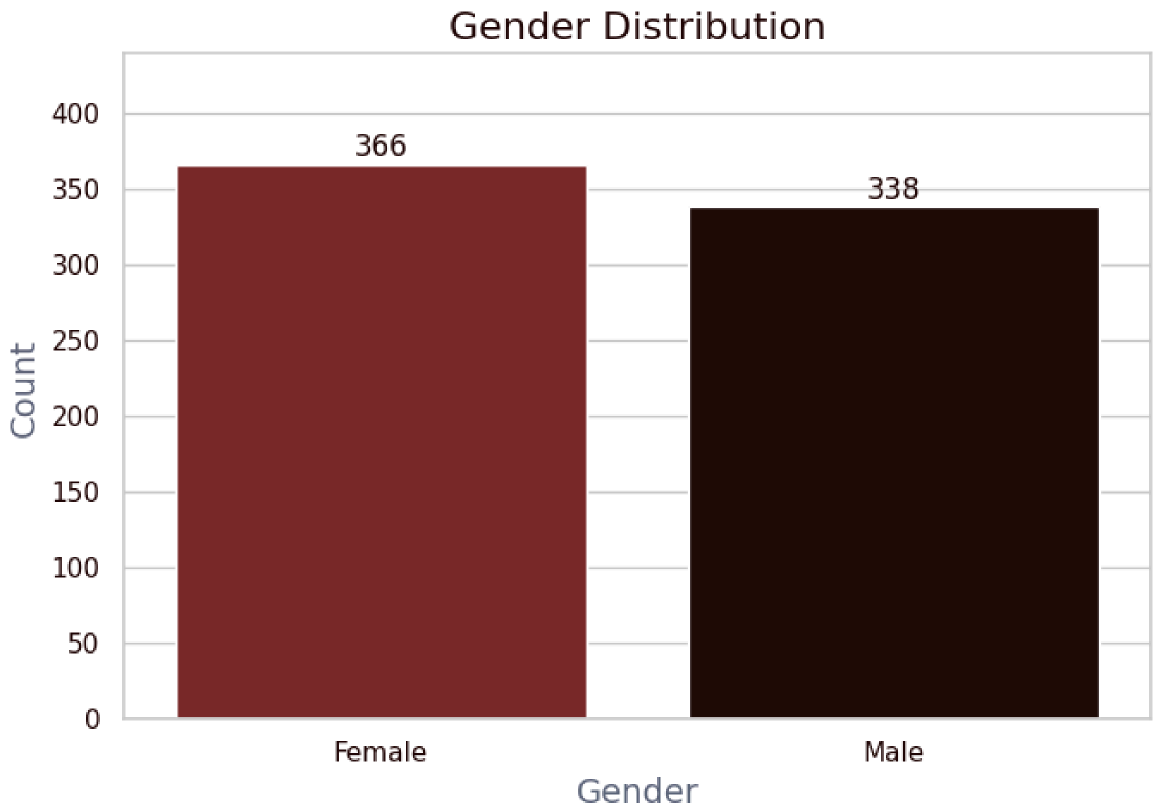
In [31]:

```
# Age Distribution - KDE plot
plt.figure(figsize=(10, 5))
sns.kdeplot(data['age'], fill=True, color=palette[1], linewidth=2)
plt.title('Age Distribution of Respondents', fontsize=16, color=palette[0])
plt.xlabel('Age', fontsize=14, color=palette[2])
plt.ylabel('Density', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

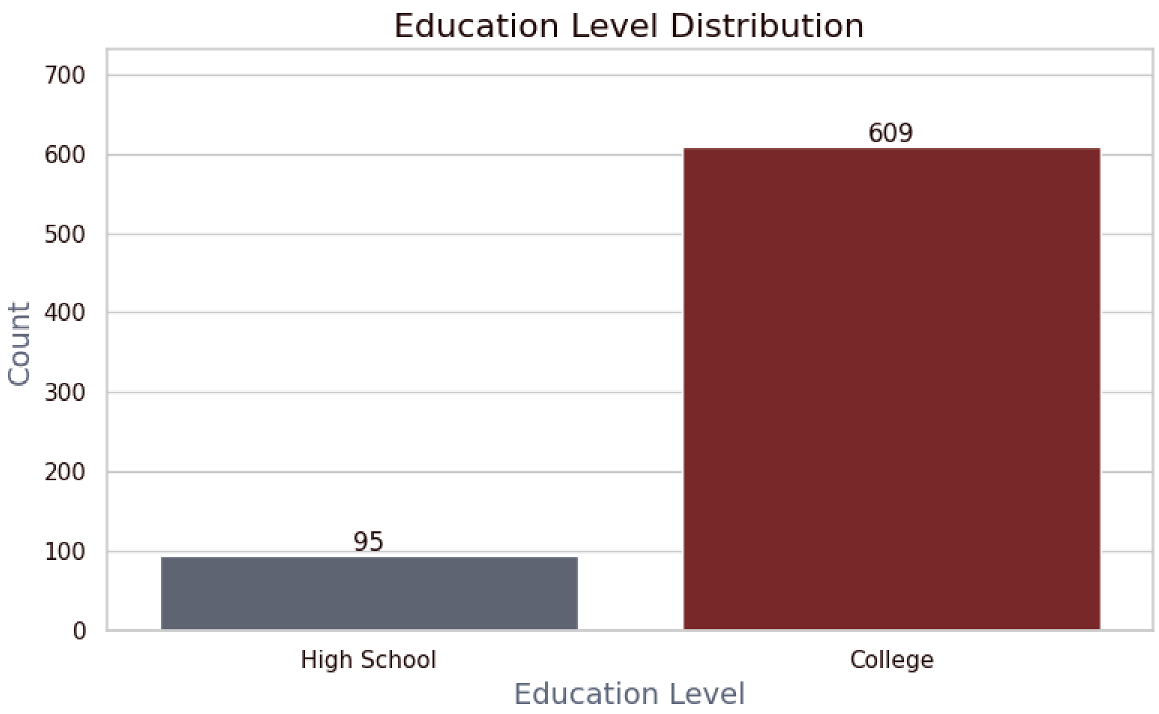


In [32]:

```
# Gender Count - Bar plot
gender_counts = pd.Series({'Female': data['gender_female'].sum(), 'Male': data['
plt.figure(figsize=(7, 5))
sns.barplot(x=gender_counts.index, y=gender_counts.values, palette=[palette[1],
plt.title('Gender Distribution', fontsize=16, color=palette[0])
plt.xlabel('Gender', fontsize=14, color=palette[2])
plt.ylabel('Count', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
for i, val in enumerate(gender_counts.values):
    plt.text(i, val + 5, str(val), ha='center', color=palette[0], fontsize=12)
plt.ylim(0, gender_counts.values.max()*1.2)
plt.tight_layout()
plt.show()
```

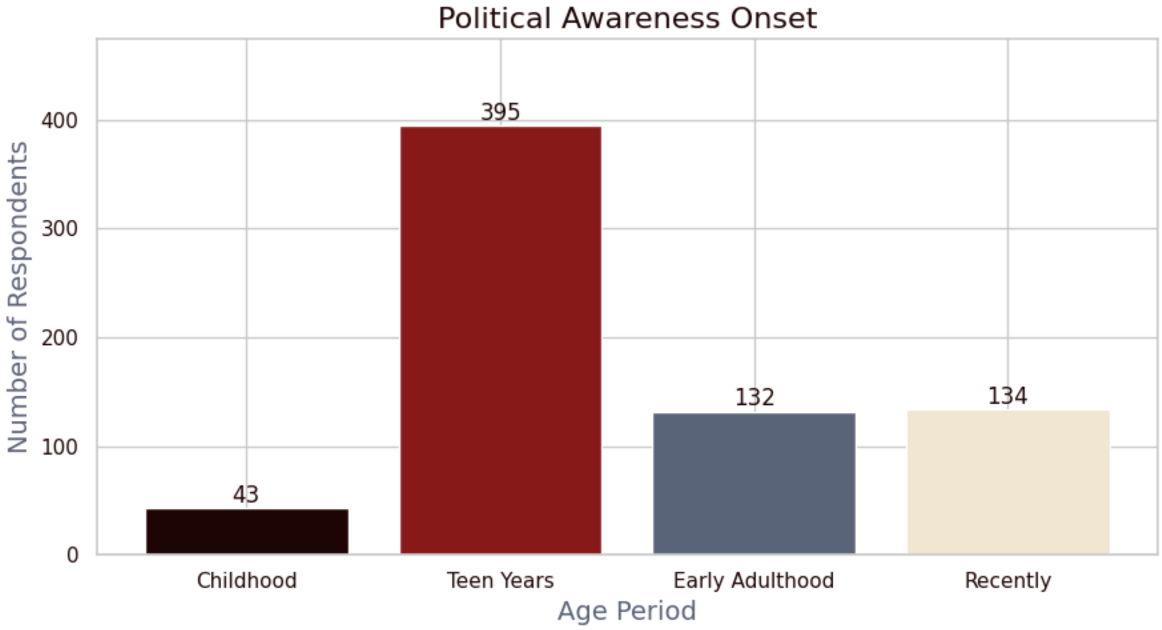


```
In [33]: # Education Level Distribution - Count Plot
edu_cols = ['educational_level_high school', 'educational_level_college']
edu_map = {'educational_level_high school': 'High School', 'educational_level_co
edu_counts = {edu_map[col]: data[col].sum() for col in edu_cols}

plt.figure(figsize=(8, 5))
sns.barplot(x=list(edu_counts.keys()), y=list(edu_counts.values()), palette=[pal
plt.title('Education Level Distribution', fontsize=16, color=palette[0])
plt.xlabel('Education Level', fontsize=14, color=palette[2])
plt.ylabel('Count', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
for i, val in enumerate(edu_counts.values()):
    plt.text(i, val + 5, str(int(val)), ha='center', color=palette[0], fontsize=
plt.ylim(0, max(edu_counts.values())*1.2)
plt.tight_layout()
plt.show()
```



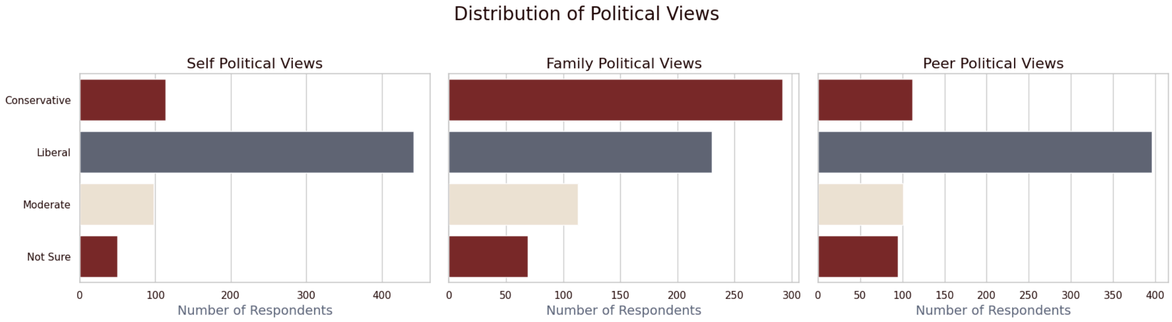
In [34]:

```
# Political Awareness Onset - Stacked Bar Chart

# Columns indicating political awareness onset
awareness_cols = [
    'age_political_awareness_onset_childhood',
    'age_political_awareness_onset_teen years',
    'age_political_awareness_onset_early adulthood',
    'age_political_awareness_onset_recently'
]

# Rename for readability
awareness_labels = ['Childhood', 'Teen Years', 'Early Adulthood', 'Recently']

awareness_counts = [data[col].sum() for col in awareness_cols]

plt.figure(figsize=(9, 5))
bars = plt.bar(awareness_labels, awareness_counts, color=palette[:4])
plt.title('Political Awareness Onset', fontsize=16, color=palette[0])
plt.xlabel('Age Period', fontsize=14, color=palette[2])
plt.ylabel('Number of Respondents', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
for i, val in enumerate(awareness_counts):
    plt.text(i, val + 5, str(int(val)), ha='center', color=palette[0], fontsize=
plt.ylim(0, max(awareness_counts)*1.2)
plt.tight_layout()
plt.show()
```



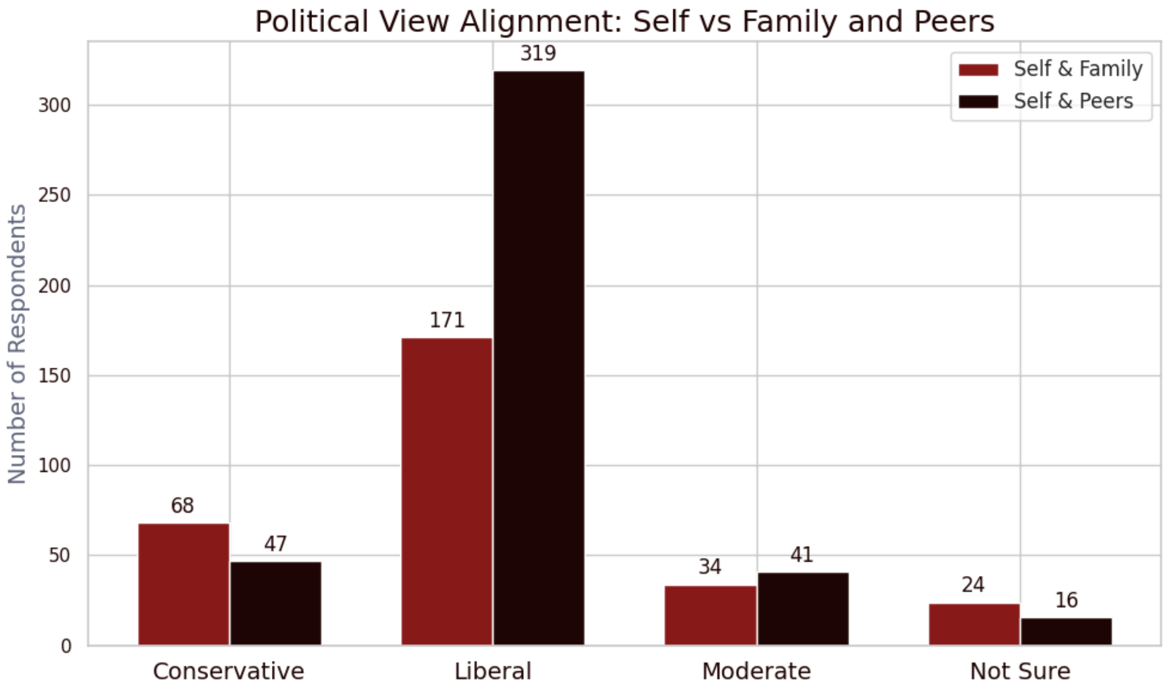
In [35]:

```
# Helper: Map boolean columns to labels for political views
def extract_view_counts(data, prefix):
    cols = [col for col in data.columns if col.startswith(prefix)]
    labels = [col.replace(prefix + '_', '').replace('_', ' ').title() for col in
    counts = [data[col].sum() for col in cols]
    return labels, counts

# 1. Distribution of Political Views for Self, Family, and Peers
fig, axs = plt.subplots(1, 3, figsize=(18, 5), sharey=True)

for ax, prefix, title in zip(
    axs,
    ['OwnPoliticalView', 'FamilyPoliticalView', 'PeerPoliticalView'],
    ['Self Political Views', 'Family Political Views', 'Peer Political Views']
):
    labels, counts = extract_view_counts(data, prefix)
    sns.barplot(x=counts, y=labels, palette=palette[1:], ax=ax)
    ax.set_title(title, fontsize=16, color=palette[0])
    ax.set_xlabel('Number of Respondents', fontsize=14, color=palette[2])
    ax.set_ylabel('')
    ax.tick_params(colors=palette[0], axis='x')
    ax.tick_params(colors=palette[0], axis='y')

plt.suptitle('Distribution of Political Views', fontsize=20, color=palette[0])
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



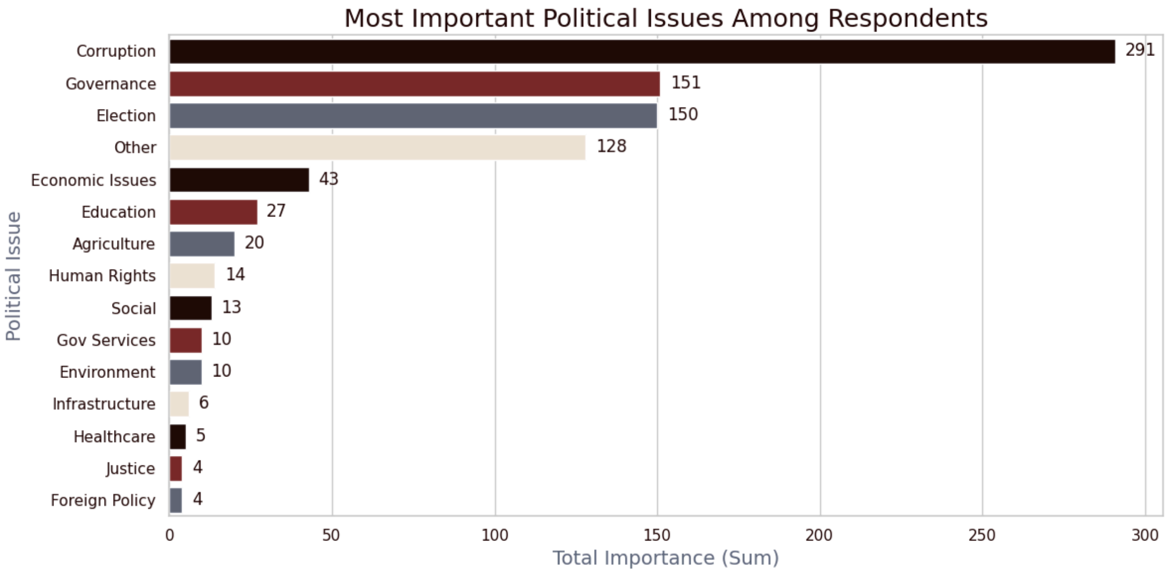
In [36]:

```
# Alignment between Self and Family / Peers Political Views
# Create simplified alignment columns for comparison
def get_alignment(df, prefix1, prefix2):
    # For each political stance: conservative, liberal, moderate, not sure
    stances = ['conservative', 'liberal', 'moderate', 'not sure']
    align_counts = []
    for stance in stances:
        both_agree = ((df[f'{prefix1}_{stance}']) & (df[f'{prefix2}_{stance}']))
        align_counts.append(both_agree)
    return stances, align_counts

stances, family_align = get_alignment(data, 'OwnPoliticalView', 'FamilyPolitical
_, peer_align = get_alignment(data, 'OwnPoliticalView', 'PeerPoliticalView')

x = np.arange(len(stances))
width = 0.35

fig, ax = plt.subplots(figsize=(10,6))
bars1 = ax.bar(x - width/2, family_align, width, label='Self & Family', color=pa
bars2 = ax.bar(x + width/2, peer_align, width, label='Self & Peers', color=palet

ax.set_xticks(x)
ax.set_xticklabels([s.title() for s in stances], fontsize=14, color=palette[0])
ax.set_ylabel('Number of Respondents', fontsize=14, color=palette[2])
ax.set_title('Political View Alignment: Self vs Family and Peers', fontsize=18,
ax.legend(facecolor='white', fontsize=12)
ax.tick_params(colors=palette[0])

# Annotate bars
for bars in [bars1, bars2]:
    for bar in bars:
        height = bar.get_height()
        ax.annotate(f'{height}', xy=(bar.get_x() + bar.get_width()/2, height),
                    xytext=(0,3), textcoords='offset points',
                    ha='center', va='bottom', color=palette[0], fontsize=12)

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



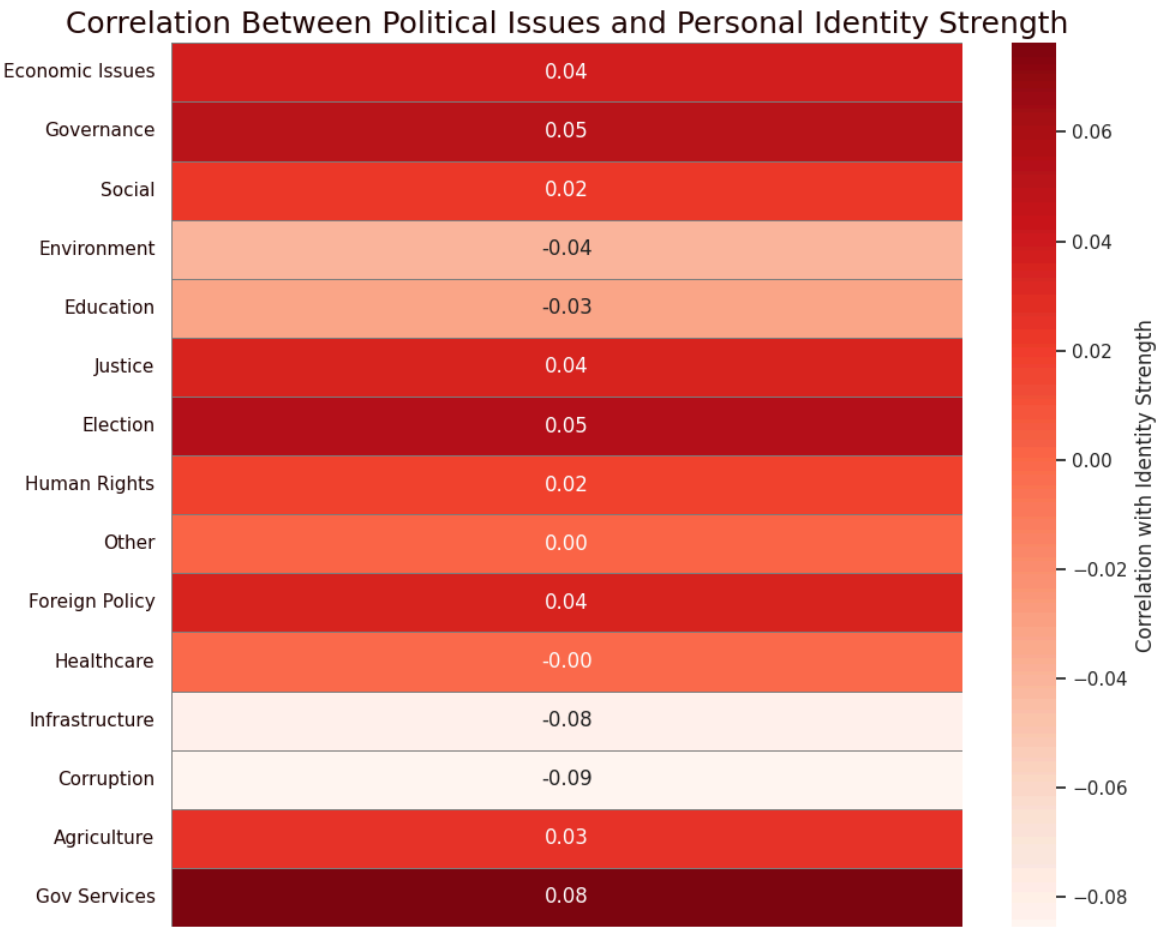
In [37]:

```
# List of issue columns (integer counts/importance)
issue_cols = [
    'issue_economic_issues', 'issue_governance', 'issue_social', 'issue_environm
    'issue_education', 'issue_justice', 'issue_election', 'issue_human_rights',
    'issue_other', 'issue_foreign_policy', 'issue_healthcare', 'issue_infrastruc
    'issue_corruption', 'issue_agriculture', 'issue_gov_services'
]

# 1. Sum total importance per issue and sort descending
issue_sums = data[issue_cols].sum().sort_values(ascending=False)

plt.figure(figsize=(12,6))
sns.barplot(x=issue_sums.values, y=[col.replace('issue_', '').replace('_', ' ').
            palette=palette[:len(issue_sums)])
plt.title('Most Important Political Issues Among Respondents', fontsize=18, colo
plt.xlabel('Total Importance (Sum)', fontsize=14, color=palette[2])
plt.ylabel('Political Issue', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
for i, val in enumerate(issue_sums.values):
    plt.text(val + max(issue_sums.values)*0.01, i, int(val), color=palette[0], f
plt.tight_layout()
plt.show()
```

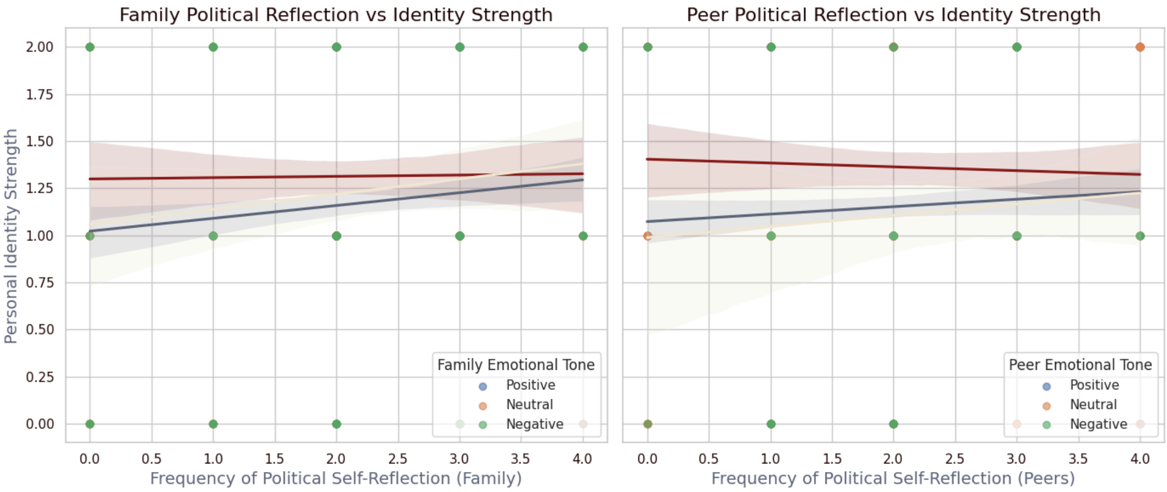


In [38]:

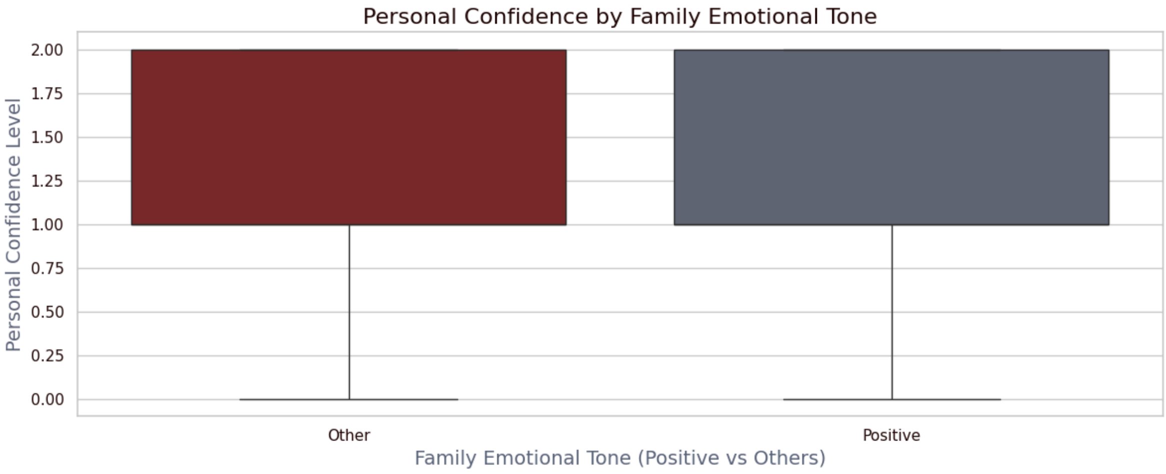
```
# Correlation heatmap: Issues vs PersonalIdentityStrength
corr_df = data[issue_cols + ['PersonalIndentityStrength']].corr()

plt.figure(figsize=(10,8))
sns.heatmap(corr_df.loc[issue_cols, 'PersonalIndentityStrength'].to_frame(),
            annot=True, cmap='Reds', center=0,
            cbar_kws={'label': 'Correlation with Identity Strength'},
            linewidths=0.5, linecolor='gray', fmt=".2f",
            yticklabels=[col.replace('issue_', '').replace('_', ' ').title() for
plt.title('Correlation Between Political Issues and Personal Identity Strength',
plt.yticks(rotation=0, color=palette[0])
plt.xticks([], [])
plt.tight_layout()
plt.show()
```

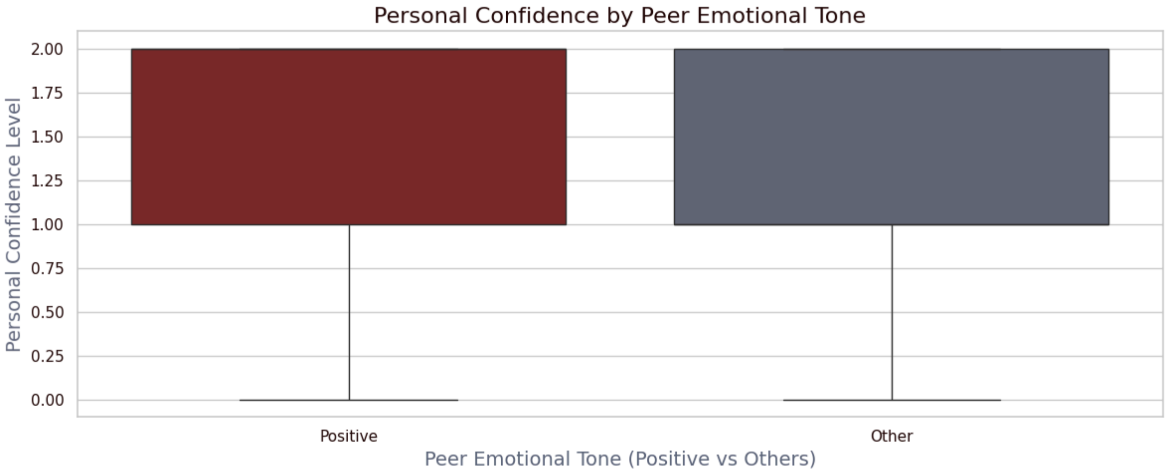


```
In [39]: # Prepare dataframe with key columns
         df = data.copy()

         fig, axs = plt.subplots(1, 2, figsize=(14, 6), sharey=True)

         # Family reflection freq vs PersonalIdentityStrength colored by FamilyEmotionalT
         family_tones = ['FamilyEmotionalTone_positive', 'FamilyEmotionalTone_neutral', '
         family_tone_labels = ['Positive', 'Neutral', 'Negative']
         family_palette = palette[1:4]

         for tone, label, color in zip(family_tones, family_tone_labels, family_palette):
             subset = df[df[tone] == True]
             sns.regplot(x='political_self_reflection_freq_family', y='PersonalIndentityS
                         data=subset, scatter_kws={'alpha':0.6}, line_kws={'color':color}
                         label=label, ax=axs[0], scatter=True)
         axs[0].set_title('Family Political Reflection vs Identity Strength', fontsize=16
         axs[0].set_xlabel('Frequency of Political Self-Reflection (Family)', fontsize=14
         axs[0].set_ylabel('Personal Identity Strength', fontsize=14, color=palette[2])
         axs[0].legend(title='Family Emotional Tone')
         axs[0].tick_params(colors=palette[0])

         # Peer reflection freq vs PersonalIdentityStrength colored by PeerEmotionalTone
         peer_tones = ['PeerEmotionalTone_positive', 'PeerEmotionalTone_neutral', 'PeerEm
         peer_palette = palette[1:4]

         for tone, label, color in zip(peer_tones, family_tone_labels, peer_palette):
             subset = df[df[tone] == True]
             sns.regplot(x='political_self_reflection_freq_peers', y='PersonalIndentitySt
                         data=subset, scatter_kws={'alpha':0.6}, line_kws={'color':color}
                         label=label, ax=axs[1], scatter=True)
         axs[1].set_title('Peer Political Reflection vs Identity Strength', fontsize=16,
         axs[1].set_xlabel('Frequency of Political Self-Reflection (Peers)', fontsize=14,
         axs[1].set_ylabel('')
         axs[1].legend(title='Peer Emotional Tone')
         axs[1].tick_params(colors=palette[0])

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



```
In [40]: plt.figure(figsize=(12,5))
sns.boxplot(
    x='FamilyEmotionalTone_positive', y='PersonalConfidenceLevel',
    data=data.replace({True:'Positive', False:'Other'}),
    palette=[palette[1], palette[2]]
)
plt.title('Personal Confidence by Family Emotional Tone', fontsize=16, color=pal
plt.xlabel('Family Emotional Tone (Positive vs Others)', fontsize=14, color=pale
plt.ylabel('Personal Confidence Level', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
plt.tight_layout()
plt.show()

plt.figure(figsize=(12,5))
sns.boxplot(
    x='PeerEmotionalTone_positive', y='PersonalConfidenceLevel',
    data=data.replace({True:'Positive', False:'Other'}),
    palette=[palette[1], palette[2]]
)
plt.title('Personal Confidence by Peer Emotional Tone', fontsize=16, color=palet
plt.xlabel('Peer Emotional Tone (Positive vs Others)', fontsize=14, color=palett
plt.ylabel('Personal Confidence Level', fontsize=14, color=palette[2])
plt.xticks(color=palette[0])
plt.yticks(color=palette[0])
plt.tight_layout()
plt.show()
```





4.5 Machine Learning Implementation

In [41]:

```
# --- DATA PREPARATION ---
X = data_reduced.drop(columns=['Political_Alignment_peers', 'cleaned_issues', 'c
                               'categories', 'categories_split', 'PoliticalIssue
                               'Political_Alignment_family'])
y = data_reduced['Political_Alignment_peers']
```

In [42]:

```
# Split data once (we'll use the same splits for all models)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [43]:

```
# Scale numeric features (important for KNN and SVM)

numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()

X_train[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test[numeric_cols] = scaler.transform(X_test[numeric_cols])
```

4.6 Proportional Chance Criteria (PCC)

In [44]:

```
# --- PROPORTIONAL CHANCE CRITERION ---
def proportional_chance_criterion(y):
    class_counts = y.value_counts(normalize=True)
    pcc = np.sum(class_counts ** 2)
    print(f"PCC: {pcc:.4f}")
    pcc_threshold = pcc * 1.25
    print(f"PCC × 1.25 threshold: {pcc_threshold:.4f}")
    return pcc_threshold

pcc_125 = proportional_chance_criterion(y)
```

```
PCC: 0.5009
PCC × 1.25 threshold: 0.6261
```

In [45]:

```
# ---  BASELINE MODELS ---

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier

models = {
    "KNN": KNeighborsClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Linear SVM": LinearSVC(max_iter=10000),
    "AdaBoost": AdaBoostClassifier(),
    "Gradient Boosting": GradientBoostingClassifier()
}
```

In [46]:

```
def evaluate_model(model, X_train, X_test, y_train, y_test, trials=50):
    train_scores = []
    test_scores = []

    for _ in tqdm(range(trials), desc=f"Evaluating {model.__class__.__name__}"):
        m = clone(model)
        m.fit(X_train, y_train)
        train_pred = m.predict(X_train)
        test_pred = m.predict(X_test)
        train_scores.append(accuracy_score(y_train, train_pred))
        test_scores.append(accuracy_score(y_test, test_pred))

    return {
        'Model': model.__class__.__name__,
        'Mean Train Accuracy': np.mean(train_scores),
        'Mean Test Accuracy': np.mean(test_scores),
        'Overfitting Gap': np.mean(train_scores) - np.mean(test_scores)
    }
```

In [47]:

```
print("\n--- Baseline Model Evaluation ---")
baseline_results = []
for name, model in models.items():
    baseline_results.append(evaluate_model(model, X_train, X_test, y_train, y_te

baseline_df = pd.DataFrame(baseline_results)[['Model', 'Mean Train Accuracy', 'M
baseline_df
```

```
--- Baseline Model Evaluation ---
```

```
Evaluating KNeighborsClassifier: 100%|██████████| 50/50 [00:04<00:00, 10.14it/s]
Evaluating DecisionTreeClassifier: 100%|██████████| 50/50 [00:01<00:00, 39.05it/
s]
Evaluating RandomForestClassifier: 100%|██████████| 50/50 [00:13<00:00,  3.67it/
s]
Evaluating LogisticRegression: 100%|██████████| 50/50 [00:02<00:00, 18.84it/s]
Evaluating LinearSVC: 100%|██████████| 50/50 [00:00<00:00, 58.02it/s]
Evaluating AdaBoostClassifier: 100%|██████████| 50/50 [00:07<00:00,  6.36it/s]
Evaluating GradientBoostingClassifier: 100%|██████████| 50/50 [00:13<00:00,  3.77
it/s]
```

Out[47]:

	Model	Mean Train Accuracy	Mean Test Accuracy	Overfitting Gap
0	KNeighborsClassifier	0.809947	0.702128	0.107819
1	DecisionTreeClassifier	1.000000	0.834326	0.165674
2	RandomForestClassifier	1.000000	0.865532	0.134468
3	LogisticRegression	0.813499	0.773050	0.040449
4	LinearSVC	0.808171	0.758865	0.049305
5	AdaBoostClassifier	0.802842	0.794326	0.008516
6	GradientBoostingClassifier	0.964476	0.879433	0.085043

4.7 Cross Validation and Hyperparamater Tuning

In [50]:

```
def robust_evaluate_with_train_score(model, X, y, cv_splits=5, trials=5):
    train_scores = []
    cv_scores = []
    pipe = make_pipeline(StandardScaler(), model)

    for _ in range(trials):
        kf = KFold(n_splits=cv_splits, shuffle=True, random_state=None)
        for train_idx, val_idx in kf.split(X):
            X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
            y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

            pipe.fit(X_train, y_train)

            train_pred = pipe.predict(X_train)
            val_pred = pipe.predict(X_val)

            train_scores.append(accuracy_score(y_train, train_pred))
            cv_scores.append(accuracy_score(y_val, val_pred))

    return {
        'Mean Train Accuracy': np.mean(train_scores),
        'Mean CV Accuracy': np.mean(cv_scores),
    }

print("\n--- Cross Validation Results (with train accuracy) ---")
cv_results = {}
for name, model in models.items():
    print(f"Evaluating {name}...")
    cv_results[name] = robust_evaluate_with_train_score(model, X, y)

cv_df = pd.DataFrame.from_dict(cv_results, orient='index')
cv_df
```

```
--- Cross Validation Results (with train accuracy) ---
Evaluating KNN...
Evaluating Decision Tree...
Evaluating Random Forest...
Evaluating Logistic Regression...
Evaluating Linear SVM...
Evaluating AdaBoost...
Evaluating Gradient Boosting...
```

Out[50]:

	Mean Train Accuracy	Mean CV Accuracy
KNN	0.817823	0.708809
Decision Tree	1.000000	0.814507
Random Forest	1.000000	0.871044
Logistic Regression	0.823294	0.764725
Linear SVM	0.821094	0.767042
AdaBoost	0.814632	0.776436
Gradient Boosting	0.956959	0.840334

In [51]:

```
param_grids = {
    'Decision Tree': {
        'model__criterion': ['gini', 'entropy'],
        'model__max_depth': [3, 5, 7, 10, None],
        'model__min_samples_split': [2, 5, 10],
        'model__min_samples_leaf': [1, 2, 4],
        'model__max_features': ['sqrt', 'log2', None],
        'model__class_weight': ['balanced', None]
    },
    'Random Forest': {
        'model__n_estimators': [50, 100, 200],
        'model__max_depth': [5, 10, 20, None],
        'model__min_samples_split': [2, 5, 10],
        'model__min_samples_leaf': [1, 2, 4],
        'model__max_features': ['sqrt', 'log2'],
        'model__bootstrap': [True, False],
        'model__class_weight': ['balanced', None]
    },
    'KNN': {
        'model__n_neighbors': [2,3,4,5,7,9,10,15],
        'model__weights': ['uniform', 'distance'],
        'model__p': [1, 2]
    },
    'Linear SVM': {
        'model__C': [0.1, 1, 10],
        'model__penalty': ['l1', 'l2'],
        'model__loss': ['hinge', 'squared_hinge'],
        'model__dual': [False],
        'model__class_weight': ['balanced', None]
    },
    'Logistic Regression': {
        'model__C': [0.1, 1, 10],
        'model__penalty': ['l1', 'l2'],
        'model__solver': ['liblinear', 'saga'],
        'model__class_weight': ['balanced', None]
    },
    'AdaBoost': {
        'model__n_estimators': [50, 100, 200],
        'model__learning_rate': [0.01, 0.1, 0.5, 1],
        'model__algorithm': ['SAMME', 'SAMME.R']
    },
    'Gradient Boosting': {
        'model__n_estimators': [50, 100, 200],
        'model__learning_rate': [0.01, 0.1, 0.2],
        'model__max_depth': [3, 5, 7],
        'model__min_samples_split': [2, 5, 10],
        'model__min_samples_leaf': [1, 2, 4],
        'model__subsample': [0.6, 0.8, 1.0],
        'model__max_features': ['sqrt', 'log2', None]
    }
}
```

In [52]:

```
def tune_model(model, param_grid, X_train, y_train):
    step_name = model.__class__.__name__.lower()
    param_grid_updated = {}
    for key, val in param_grid.items():
        if key.startswith('model__'):
            new_key = key.replace('model__', f'{step_name}__')
            param_grid_updated[new_key] = val
        else:
            param_grid_updated[key] = val
    pipe = Pipeline([
        ('standardscaler', StandardScaler()),
        (step_name, model)
    ])
    grid = GridSearchCV(pipe, param_grid_updated, cv=5, scoring='accuracy', n_jc
    grid.fit(X_train, y_train)  # use only training data here
    print(f"\nBest parameters for {model.__class__.__name__}:")
    print(grid.best_params_)
    return grid.best_estimator_, grid.best_params_
```

In [53]:

```
best_models = {}
best_params = {}
for name, model in models.items():
    print(f"\n=== Tuning {name} ===")
    best_model, best_param = tune_model(model, param_grids[name], X_train, y_tra
    best_models[name] = best_model
    best_params[name] = best_param
```

```
=== Tuning KNN ===

Best parameters for KNeighborsClassifier:
{'kneighborsclassifier__n_neighbors': 3, 'kneighborsclassifier__p': 1, 'kneighbor
sclassifier__weights': 'distance'}

=== Tuning Decision Tree ===

Best parameters for DecisionTreeClassifier:
{'decisiontreeclassifier__class_weight': None, 'decisiontreeclassifier__criterio
n': 'gini', 'decisiontreeclassifier__max_depth': 10, 'decisiontreeclassifier__max
_features': None, 'decisiontreeclassifier__min_samples_leaf': 2, 'decisiontreecla
ssifier__min_samples_split': 2}

=== Tuning Random Forest ===

Best parameters for RandomForestClassifier:
{'randomforestclassifier__bootstrap': False, 'randomforestclassifier__class_weigh
t': None, 'randomforestclassifier__max_depth': 20, 'randomforestclassifier__max_f
eatures': 'sqrt', 'randomforestclassifier__min_samples_leaf': 1, 'randomforestcla
ssifier__min_samples_split': 2, 'randomforestclassifier__n_estimators': 50}

=== Tuning Logistic Regression ===

Best parameters for LogisticRegression:
{'logisticregression__C': 1, 'logisticregression__class_weight': None, 'logisticr
egression__penalty': 'l2', 'logisticregression__solver': 'liblinear'}

=== Tuning Linear SVM ===

Best parameters for LinearSVC:
{'linearsvc__C': 0.1, 'linearsvc__class_weight': 'balanced', 'linearsvc__dual': F
alse, 'linearsvc__loss': 'squared_hinge', 'linearsvc__penalty': 'l2'}

=== Tuning AdaBoost ===

Best parameters for AdaBoostClassifier:
{'adaboostclassifier__algorithm': 'SAMME', 'adaboostclassifier__learning_rate':
1, 'adaboostclassifier__n_estimators': 100}

=== Tuning Gradient Boosting ===

Best parameters for GradientBoostingClassifier:
{'gradientboostingclassifier__learning_rate': 0.1, 'gradientboostingclassifier__m
ax_depth': 7, 'gradientboostingclassifier__max_features': 'log2', 'gradientboosti
ngclassifier__min_samples_leaf': 4, 'gradientboostingclassifier__min_samples_spli
t': 10, 'gradientboostingclassifier__n_estimators': 100, 'gradientboostingclassif
ier__subsample': 0.6}
```

5. Results and Discussion

Since Random Forest or Decision Tree models provide feature importances directly, you can use one of these models to get the top features.

In [54]:

```python
# Use the best Random Forest model (or Decision Tree) to find feature importance
best_rf = best_models['Random Forest']

# Extract feature importances
```

```
importances = best_rf.named_steps['randomforestclassifier'].feature_importances_

# Get feature names from X_train (before scaling)
feature_names = X_train.columns

# Create a dataframe of features and their importances
feat_imp_df = pd.DataFrame({'feature': feature_names, 'importance': importances}

# Sort by importance descending and select top 12
top_features = feat_imp_df.sort_values(by='importance', ascending=False).head(12

print("Top 12 Features:")
print(top_features)
```

```
Top 12 Features:
['ParentApprovalLevel', 'FamilyInfluenceStrength', 'PeerApprovalLevel', 'family_a
pproval_importance_politics', 'PeerDiscussionInterest', 'PeerEngagement', 'age',
'FamilyEngagement', 'importance_fam_political_alignment', 'political_self_reflect
ion_freq_peers', 'PeerInfluenceStrength', 'political_self_reflection_freq_famil
y']
```

Retrain All Models Using Only Top 12 Features and Best Hyperparameters

In [55]:
```
X_train_top = X_train[top_features]
X_test_top = X_test[top_features]
```

In [56]:
```
retrained_models = {}
train_accs = {}
test_accs = {}
overfit_gaps = {}

for name, model in models.items():
    print(f"\nRetraining {name} with best hyperparameters on top features...")

    # Extract best params from original tuning
    params = best_params[name]

    # Remove pipeline prefixes
    step_name = model.__class__.__name__.lower()
    model_params = {k.split(f"{step_name}__")[1]: v for k, v in params.items() i

    # Instantiate model with best params
    model_best = model.__class__(**model_params)

    # Build pipeline with scaler + model
    pipe = Pipeline([
        ('standardscaler', StandardScaler()),
        (step_name, model_best)
    ])

    # Fit on top features only
    pipe.fit(X_train_top, y_train)

    # Predict train and test
    train_pred = pipe.predict(X_train_top)
    test_pred = pipe.predict(X_test_top)

    train_acc = accuracy_score(y_train, train_pred)
    test_acc = accuracy_score(y_test, test_pred)
    overfit_gap = train_acc - test_acc
```

```python
    print(f"{name} Train Accuracy: {train_acc:.4f}")
    print(f"{name} Test Accuracy: {test_acc:.4f}")
    print(f"{name} Overfitting Gap: {overfit_gap:.4f}")

    retrained_models[name] = pipe
    train_accs[name] = train_acc
    test_accs[name] = test_acc
    overfit_gaps[name] = overfit_gap

# Summary table with Overfitting Gap
results_summary = pd.DataFrame({
    'Model': list(train_accs.keys()),
    'Train Accuracy': list(train_accs.values()),
    'Test Accuracy': list(test_accs.values()),
    'Overfitting Gap': list(overfit_gaps.values())
})

print("\n=== Summary of retrained models with best params on top features ===")
print(results_summary)
```

```
Retraining KNN with best hyperparameters on top features...
KNN Train Accuracy: 0.9982
KNN Test Accuracy: 0.8227
KNN Overfitting Gap: 0.1755

Retraining Decision Tree with best hyperparameters on top features...
Decision Tree Train Accuracy: 0.9307
Decision Tree Test Accuracy: 0.8298
Decision Tree Overfitting Gap: 0.1009

Retraining Random Forest with best hyperparameters on top features...
Random Forest Train Accuracy: 0.9982
Random Forest Test Accuracy: 0.9007
Random Forest Overfitting Gap: 0.0975

Retraining Logistic Regression with best hyperparameters on top features...
Logistic Regression Train Accuracy: 0.7815
Logistic Regression Test Accuracy: 0.7447
Logistic Regression Overfitting Gap: 0.0368

Retraining Linear SVM with best hyperparameters on top features...
Linear SVM Train Accuracy: 0.7815
Linear SVM Test Accuracy: 0.7447
Linear SVM Overfitting Gap: 0.0368

Retraining AdaBoost with best hyperparameters on top features...
AdaBoost Train Accuracy: 0.7957
AdaBoost Test Accuracy: 0.7518
AdaBoost Overfitting Gap: 0.0440

Retraining Gradient Boosting with best hyperparameters on top features...
Gradient Boosting Train Accuracy: 0.9982
Gradient Boosting Test Accuracy: 0.8440
Gradient Boosting Overfitting Gap: 0.1543

=== Summary of retrained models with best params on top features ===
                 Model  Train Accuracy  Test Accuracy  Overfitting Gap
0                  KNN        0.998224       0.822695         0.175529
1        Decision Tree        0.930728       0.829787         0.100941
2        Random Forest        0.998224       0.900709         0.097515
3  Logistic Regression        0.781528       0.744681         0.036847
4           Linear SVM        0.781528       0.744681         0.036847
5             AdaBoost        0.795737       0.751773         0.043964
6    Gradient Boosting        0.998224       0.843972         0.154252
```

5.2 SHAP (SHapley Additive exPlanations)

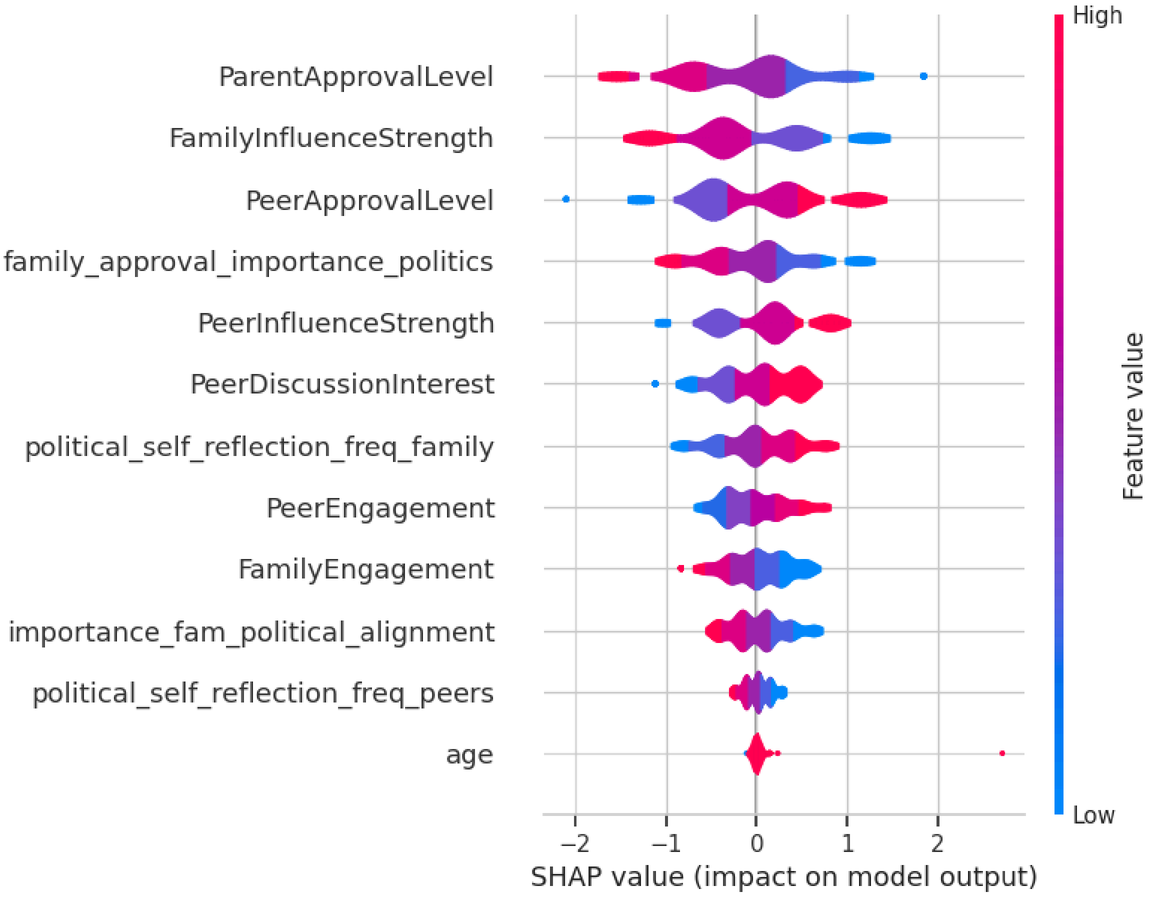
In [57]:

```
import shap
import warnings
```

In [58]:

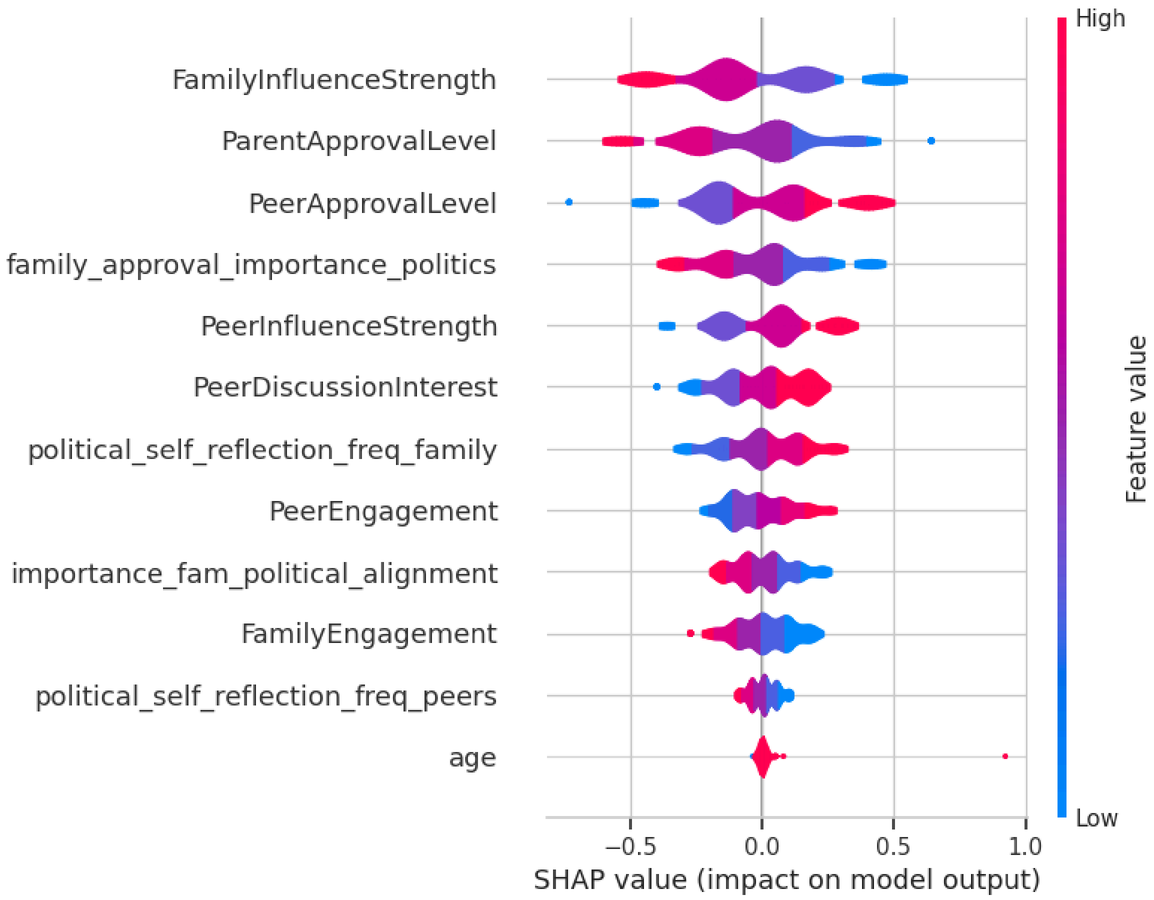
```
logreg_pipeline = retrained_models['Logistic Regression']
logreg_model = logreg_pipeline.named_steps['logisticregression']

X_explain = X_train_top

explainer = shap.LinearExplainer(logreg_model, X_explain, feature_perturbation="
shap_values = explainer.shap_values(X_explain)
```

```
shap.summary_plot(shap_values, X_explain, plot_type="violin")
```



In [61]:

```
logreg_pipeline = retrained_models['Linear SVM']
logreg_model = logreg_pipeline.named_steps['linearsvc']

X_explain = X_train_top

explainer = shap.LinearExplainer(logreg_model, X_explain, feature_perturbation="
shap_values = explainer.shap_values(X_explain)

shap.summary_plot(shap_values, X_explain, plot_type="violin")
```



5.3 Model Evaluation

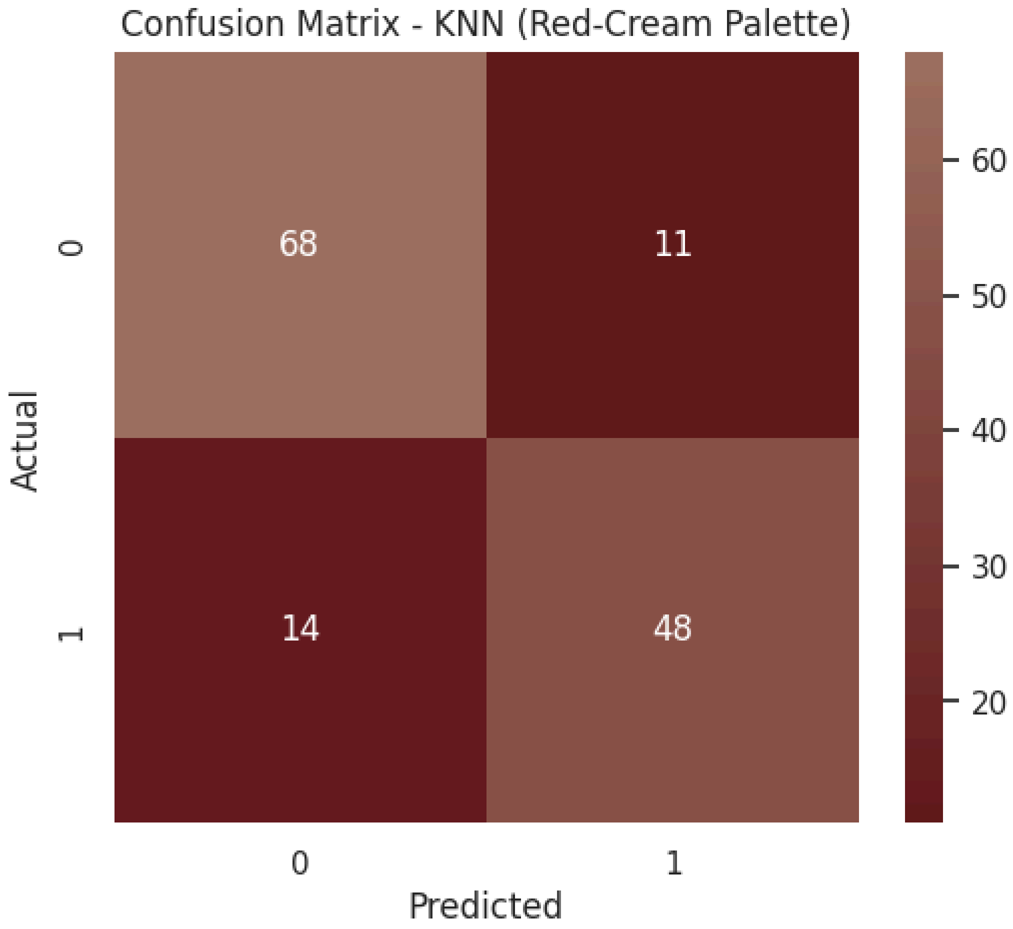
In [62]:

```python
from sklearn.metrics import classification_report, confusion_matrix
from matplotlib.colors import LinearSegmentedColormap
```

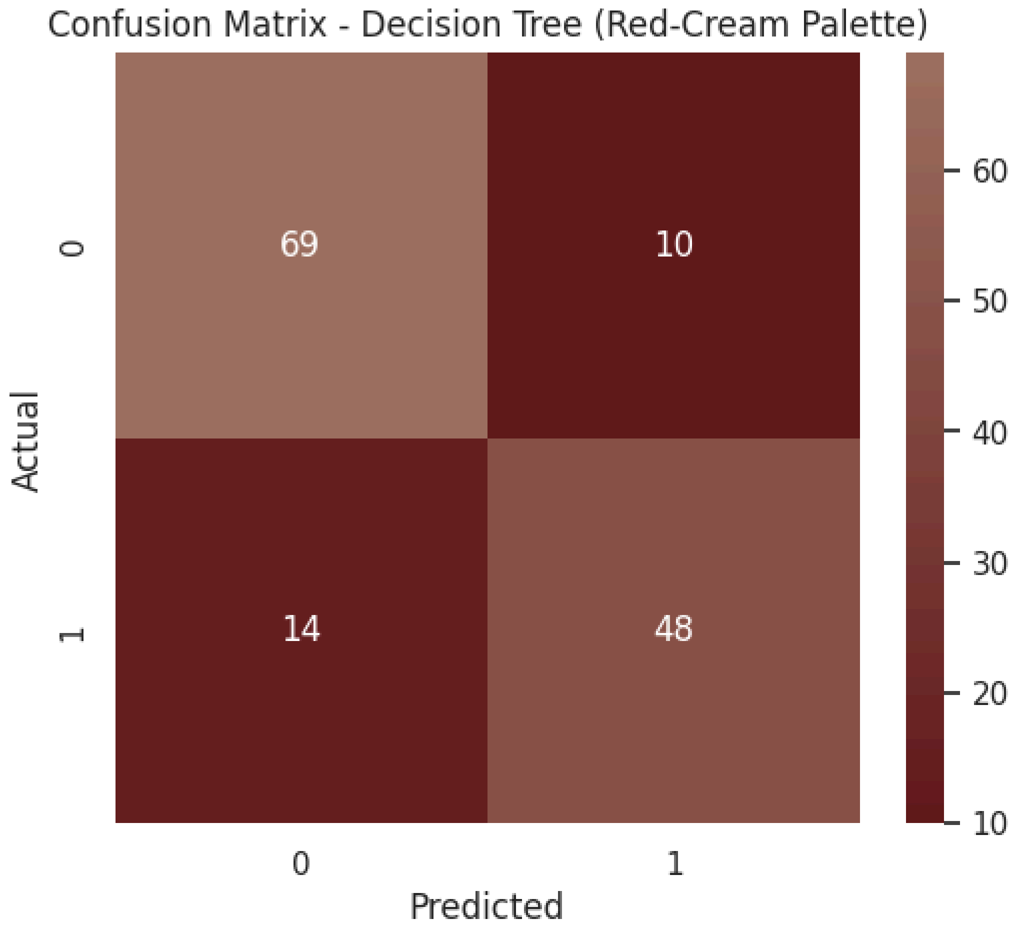
In [63]:

```python
# Define red to cream colormap
red_cream_cmap = LinearSegmentedColormap.from_list("RedCream", ["#62191C", "#9E7

for model_name, model in retrained_models.items():
    print(f"\n=== Confusion Matrix and Report for {model_name} ===")

    # Predict on test data with top features
    y_pred = model.predict(X_test_top)

    # Print classification report
    print(classification_report(y_test, y_pred))

    # Compute confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Plot confusion matrix heatmap
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt='d', cmap=red_cream_cmap, cbar=True)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix - {model_name} (Red-Cream Palette)')
    plt.show()
```

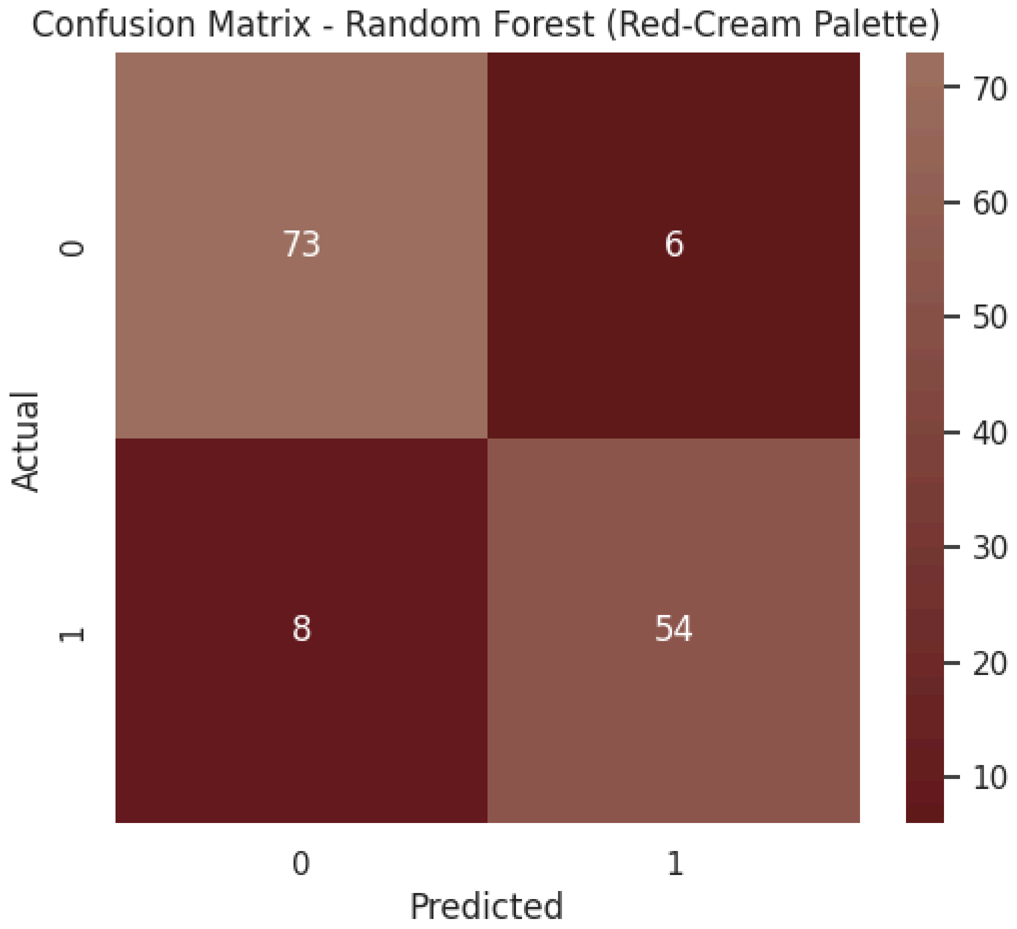
```
=== Confusion Matrix and Report for KNN ===
              precision    recall  f1-score   support

       False       0.83      0.86      0.84        79
        True       0.81      0.77      0.79        62

    accuracy                           0.82       141
   macro avg       0.82      0.82      0.82       141
weighted avg       0.82      0.82      0.82       141
```



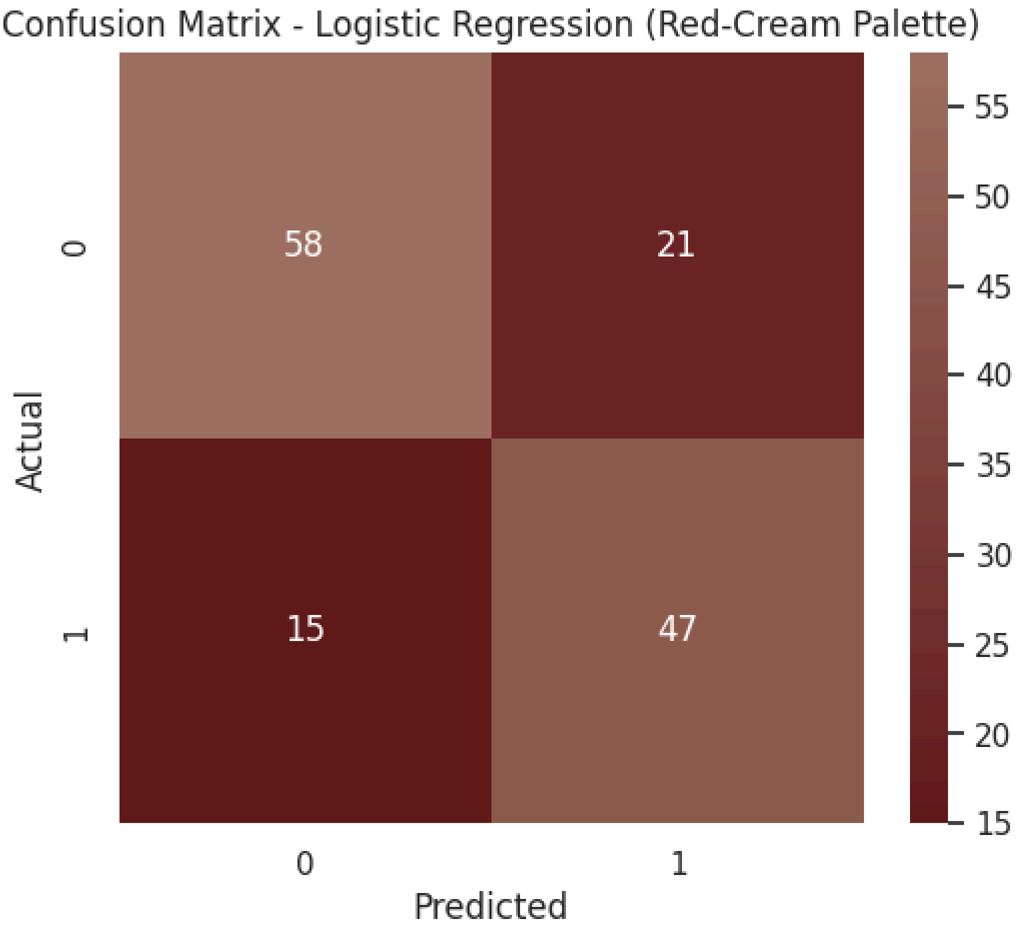
```
=== Confusion Matrix and Report for Decision Tree ===
              precision    recall  f1-score   support

       False       0.83      0.87      0.85        79
        True       0.83      0.77      0.80        62

    accuracy                           0.83       141
   macro avg       0.83      0.82      0.83       141
weighted avg       0.83      0.83      0.83       141
```



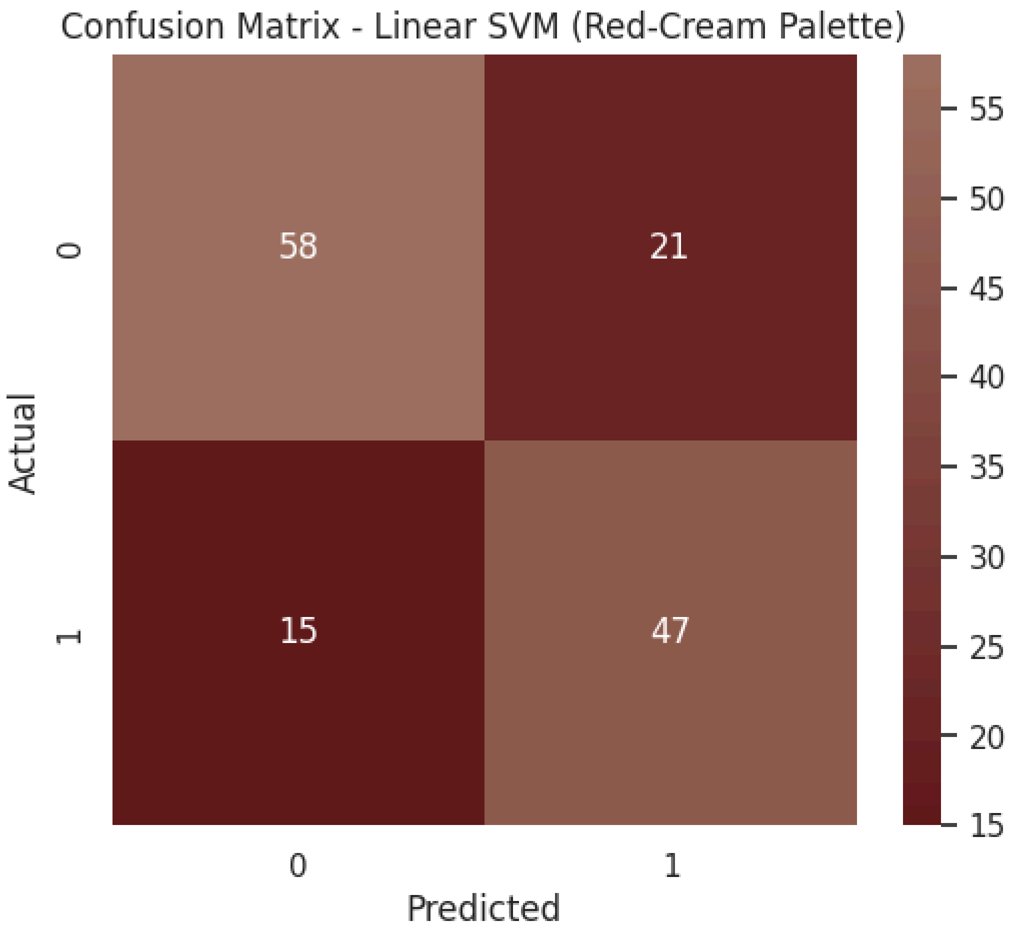
```
=== Confusion Matrix and Report for Random Forest ===
              precision    recall  f1-score   support

       False       0.90      0.92      0.91        79
        True       0.90      0.87      0.89        62

    accuracy                           0.90       141
   macro avg       0.90      0.90      0.90       141
weighted avg       0.90      0.90      0.90       141
```



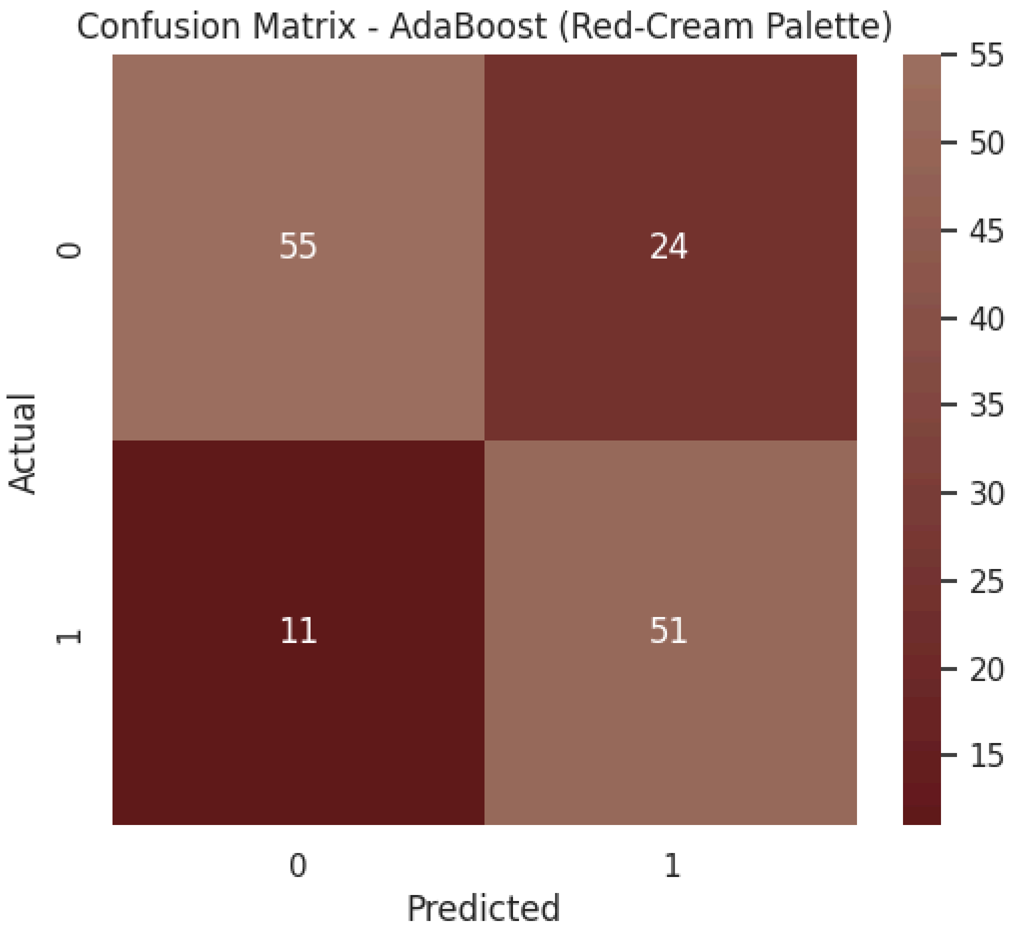
```
=== Confusion Matrix and Report for Logistic Regression ===
              precision    recall  f1-score   support

       False       0.79      0.73      0.76        79
        True       0.69      0.76      0.72        62

    accuracy                           0.74       141
   macro avg       0.74      0.75      0.74       141
weighted avg       0.75      0.74      0.75       141
```



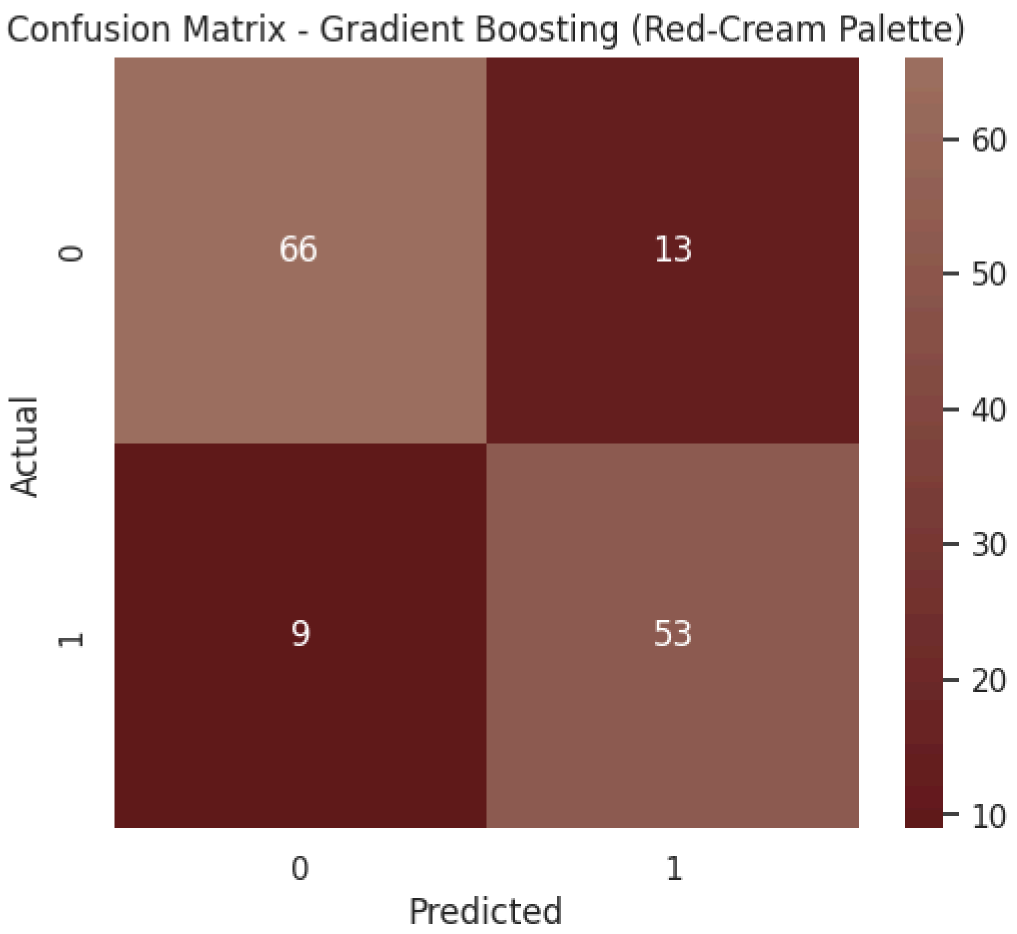
```
=== Confusion Matrix and Report for Linear SVM ===
              precision    recall  f1-score   support

       False       0.79      0.73      0.76        79
        True       0.69      0.76      0.72        62

    accuracy                           0.74       141
   macro avg       0.74      0.75      0.74       141
weighted avg       0.75      0.74      0.75       141
```



```
=== Confusion Matrix and Report for AdaBoost ===
              precision    recall  f1-score   support

       False       0.83      0.70      0.76        79
        True       0.68      0.82      0.74        62

    accuracy                           0.75       141
   macro avg       0.76      0.76      0.75       141
weighted avg       0.77      0.75      0.75       141
```



```
=== Confusion Matrix and Report for Gradient Boosting ===
              precision    recall  f1-score   support

       False       0.88      0.84      0.86        79
        True       0.80      0.85      0.83        62

    accuracy                           0.84       141
   macro avg       0.84      0.85      0.84       141
weighted avg       0.85      0.84      0.84       141
```



6. Conclusion

This study reveals how deeply our political beliefs are shaped by the people around us—especially our family and peers. Rather than forming our political identity alone, we constantly absorb and reflect the views, approval, and pressures from these social groups. The project shows that both family and peer influences are important, but their impact varies for each person.

By exploring how often people talk about politics, how emotionally charged those conversations are, and how much approval or pressure individuals feel, we gain a clearer picture of the social forces shaping political opinions. Understanding whether family or peers have a stronger influence helps us appreciate that political identity is not fixed but a dynamic social process.

This insight is powerful because it highlights that political views are built through relationships and interactions—not just personal opinions or isolated choices. It reminds us that when we engage in political conversations or education, we must consider these social influences to foster critical thinking and independent perspectives.

This project, "Who Told You What to Believe? Deconstructing the Social Blueprint of Political Identity through Machine Learning," successfully demonstrates that political identity formation is a nuanced social process deeply influenced by the interplay of family and peer contexts. By leveraging machine learning classification techniques on comprehensive survey data, we quantitatively uncovered the relative impact of various social and psychological factors in shaping political alignment.

The results reveal that while Random Forest achieves the highest test accuracy (90.07%), models such as Logistic Regression and Linear SVM, despite their lower accuracy (~74.5%), exhibit substantially smaller overfitting gaps (approximately 3.7%), highlighting their robustness and reliability in generalization. This suggests that simpler linear models, although less complex, avoid overfitting and may offer more interpretable insights into the underlying social mechanisms influencing political beliefs.

AdaBoost emerges as a compelling intermediate option, balancing test accuracy (75.2%) with a low overfitting gap (4.4%). Its ensemble boosting mechanism effectively focuses on hard-to-classify instances, capturing subtle social dynamics that linear models might miss while maintaining good generalizability.

The SHAP analysis further confirms that critical features like ParentApprovalLevel, FamilyInfluenceStrength, and PeerApprovalLevel significantly contribute to the models' predictions, emphasizing the complex interdependence of approval and influence from both family and peers. This validates the theoretical foundation that political identity is not built in isolation but constructed through continuous social interaction and negotiation.

Classifying whether family or peer influence predominates is crucial, as it informs our understanding of the socialization process that underpins political behavior and engagement. Identifying these dominant influences can help target political education and critical thinking initiatives tailored to different social contexts, ultimately fostering independent political thought.

Hyperparameter Tuning Parameter Grid

This table shows the hyperparameters and their values during grid search for each model:

Model	Hyperparameter	Values
K-Nearest Neighbors	n_neighbors	3
	p	1
	weights	distance
Decision Tree	class_weight	None
	criterion	gini
	max_depth	10
	max_features	None
	min_samples_leaf	2
	min_samples_split	2

Model	Hyperparameter	Values
Random Forest	bootstrap	False
	class_weight	None
	max_depth	20
	max_features	sqrt
	min_samples_leaf	1
	min_samples_split	2
	n_estimators	50
Logistic Regression	C	1
	class_weight	None
	penalty	l2
	solver	liblinear
Linear SVM	C	0.1
	class_weight	balanced
	dual	False
	loss	squared_hinge
	penalty	l2
AdaBoost	algorithm	SAMME
	learning_rate	1
	n_estimators	100
Gradient Boosting	learning_rate	0.1
	max_depth	7
	max_features	log2
	min_samples_leaf	4
	min_samples_split	10
	n_estimators	100
	subsample	0.6

Top 12 Features

These are the most important features selected based on feature importance scores from the Random Forest model:

Rank	Feature Name
1	ParentApprovalLevel
2	PeerApprovalLevel
3	FamilyInfluenceStrength
4	PeerDiscussionInterest
5	family_approval_importance_politics
6	PeerEngagement
7	importance_fam_political_alignment
8	FamilyEngagement
9	age
10	FamilyDiscussionInterest
11	political_self_reflection_freq_family
12	importance_peer_political_alignment

.

Why This Project is Important

This project addresses a vital gap in political socialization research by quantitatively differentiating family versus peer influence on political identity, which traditional qualitative studies often overlook or treat separately. By integrating multidimensional survey questions that capture frequency, emotional tone, approval importance, social pressure, and reflective engagement, the project validates these variables as key predictors of political alignment.

The survey questions were carefully selected and validated based on theoretical relevance from political socialization, social identity, and group conformity theories. The inclusion of demographic moderators (age, gender, education) adds depth to understanding how individual factors interact with social influence. This rigorous validation ensures the model reflects real-world social dynamics rather than abstract correlations.

Technical Recommendations

Model Deployment & Interpretation:

- Utilize Random Forest for high-accuracy predictive tasks where maximizing classification performance is the priority.

- For interpretability and minimizing overfitting risks, employ Logistic Regression or Linear SVM models, especially when explaining political influence factors to non-technical stakeholders or policymakers.
- Consider AdaBoost as a middle ground for balancing complexity, accuracy, and robustness. Its iterative focusing on misclassified cases could capture nuanced social patterns missed by other models.

Further Research:

- Extend the dataset to include longitudinal tracking to observe how family and peer influences evolve over time in political identity development.
- Explore additional social contexts such as media influence, community engagement, and online networks to build a more comprehensive social blueprint.
- Investigate the moderating role of psychological traits like openness, conformity tendency, and critical thinking skills in mediating social influence effects.

Practical Application:

- Use these insights to design targeted civic education programs that consider dominant social influences on youth political socialization.
- Encourage family-based dialogues alongside peer group discussions to foster balanced political awareness and critical reflection.
- Promote independent political thought by highlighting how social contexts shape, but do not determine, political identity.

Why the Survey Questions are Essential & Valid

- The survey questions capture a broad and relevant spectrum of social and psychological dimensions impacting political alignment:
- Social Influence Variables: Measuring approval levels, emotional tone, discussion interest, and engagement reflects the quality and intensity of political socialization within family and peer groups.
- Reflective Engagement: Questions about political self-reflection frequency highlight internal cognitive processing influenced by social contexts.
- Demographics & Identity Strength: Variables like age, gender, and personal confidence contextualize political identity within developmental and social frameworks.

Recommendations (Non-Technical)

- **Encourage Open Political Conversations in Families and Peer Groups** Creating spaces where young people can openly discuss politics with family and friends helps them understand different viewpoints and develop their own beliefs. Encouraging respectful dialogue reduces pressure and conflict and supports independent thinking.

- **Support Youth in Reflecting on Social Influences** Helping young people recognize how family and peer opinions shape their views empowers them to critically evaluate these influences. This can be done through workshops, education programs, or community activities that promote self-reflection about political beliefs.

- **Tailor Civic Education to Social Contexts** Since family and peer influences differ among individuals, civic education should be flexible and responsive. Programs can include family engagement activities for those strongly influenced by home, and peer-led discussions for those more influenced by friends.

- **Promote Awareness that Political Identity is Socially Constructed** Sharing the idea that political beliefs grow out of social relationships helps reduce polarization and promotes empathy. Understanding this can encourage people to listen more carefully to others and appreciate the social background behind differing political views.

Connecting the Project to Its Real-World Importance & Deeper Understanding

If someone asks: *“Isn’t everyone entitled to their own choices, especially in voting or politics? Why should we care if peers or family influence them? What’s the strength of this project?”*

Individual choice exists, but it’s rarely made in isolation. No person makes decisions — political or otherwise — in a vacuum. Our beliefs and votes are shaped by conversations, experiences, and values passed down or shared by family, friends, and social circles. The question is not whether influence exists — it clearly does — but how much and in what ways it operates. Our project quantifies this influence for the first time in a data-driven, objective way.

Machine learning reveals patterns invisible to the naked eye. Traditional surveys or anecdotal evidence often miss the complex interplay of multiple social factors. Our model captures these subtle influences across many variables, revealing which relationships are strongest. It goes beyond simple assumptions and tells us which social influences matter most, and under what conditions.

Why is that relevant? Because understanding influence empowers choice. Knowing that family or peers sway political opinions isn’t about denying personal freedom — it’s about recognizing the real context in which choices are made. With that knowledge, we can design better educational programs, promote critical thinking, or encourage exposure to diverse viewpoints — all helping individuals make more informed and independent decisions.

After classification — what then? What deeper insight do we gain?

- Why do these influences hold more weight? Is it trust, shared values, or lack of alternative information sources?
- Are there underlying societal or cultural reasons that reinforce certain social circles over others?

Ultimately, this project bridges data and democracy. It reminds us that “freedom of choice” is also about freedom informed by awareness. Machine learning here is a tool to illuminate the social dynamics that shape democracy itself — so that stakeholders, whether policymakers, educators, or citizens, can engage with the political process more consciously and effectively.

7. References

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (NeurIPS), 30, 4765–4774.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.

Bartels, L. M. (2008). The Study of Electoral Behavior. In The Oxford Handbook of American Elections and Political Behavior.

Huckfeldt, R., & Sprague, J. (1995). Citizens, Politics, and Social Communication: Information and Influence in an Election Campaign. Cambridge University Press.

Gelman, A., & Hill, J. (2007). Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press.

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

Shmueli, G. (2010). To Explain or to Predict? Statistical Science, 25(3), 289–310.

Pearl, J. (2009). Causality: Models, Reasoning, and Inference. Cambridge University Press.

Dolezal, M., & Schröder, M. (2017). Social Influence and Political Attitudes: Evidence from Experimental Data. Journal of Experimental Political Science, 4(2), 145–159.

Iyengar, S., & Simon, A. (2000). New Perspectives and Evidence on Political Communication and Campaign Effects. Annual Review of Psychology, 51, 149–169.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Lazer, D., et al. (2018). The Science of Fake News. Science, 359(6380), 1094–1096.

Asch, S. E. (1955). Opinions and social pressure. Scientific American, 193(5), 31-35.

Beck, P. A., & Jennings, M. K. (1991). Family traditions, political periods, and the development of partisan orientations. The Journal of Politics, 53(3), 742-763.

Jennings, M. K., & Niemi, R. G. (1968). The transmission of political values from parent to child. American Political Science Review, 62(1), 169-184.

Tajfel, H., & Turner, J. C. (1979). An integrative theory of intergroup conflict. In The social psychology of intergroup relations (pp. 33-47). Monterey, CA: Brooks/Cole.