Malaysian Political Dynamics Predictor (MPDP) - Project Update

Overview

The Malaysian Political Dynamics Predictor (MPDP) is a cutting-edge, data-driven initiative dedicated to analyzing and predicting critical aspects of Malaysia's political landscape. This project is particularly focused on forecasting party-switching behavior among politicians, pinpointing key influencers in these shifts, and projecting potential coalition formations. It utilizes extensive historical data to understand the complex political dynamics that emerged during the 2020–2022 Malaysian political crisis.

Project Code

The MPDP's source code is accessible on GitHub at https://github.com/jeonghin/MPDP.

Data Sources

The project's data is primarily sourced from Thevesh Theva's GitHub repository at https://github.com/Thevesh/analysis-election-msia/tree/main/data, which offers comprehensive data on Malaysian parliamentary election results. Additional data (if required) will be obtained from the Wikipedia entries that detail Malaysian political parties and the politicians' backgrounds.

Data Summary

A thorough summary of my data, including insights and preliminary findings, is available in the project's README. This document provides a snapshot of my approach and the initial outcomes of our analysis. The README is at https://github.com/jeonghin/MPDP/blob/main/README.md

Data Description

The are two main files that are currently being used for the project: results_parlimen_ge15.csv and candidates_ge15.csv. The description for each field for the dataset can be found in the README. The Malaysian parliament has 222 seats. There was a total of 945 candidates who participated in the election held in 2022.

Data Structure

I will structure the data in the form of graphs to clearly illustrate the relationships and connections between politicians. This approach enables me to effectively map party-switching patterns and influential networks. I created two classes: Voter and ElectionModel to be used in the Agent-Based Model (ABM) to predict voter's voting preferences and visualize the entire election simulation, including voters, influencers, and the dynamics of preference changes over election cycles. In the ABM, I included some important factors such as Race, Political Ideology, and Voter Category as these are some factors that are known to be influecing voters' preferences.

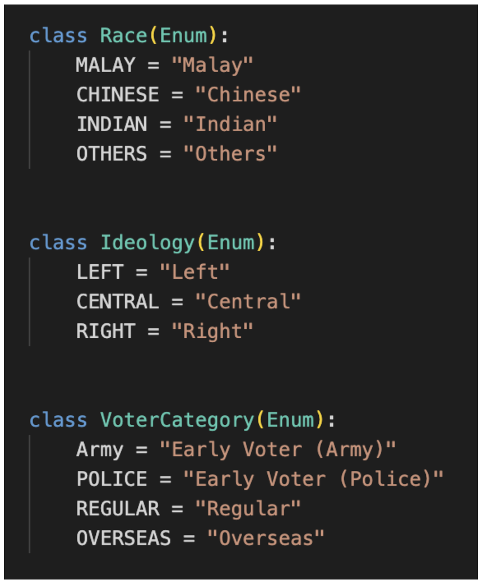
Interaction and Presentation

For an interactive experience, I am employing Plotly to visualize our data dynamically. These visualizations are designed to provide intuitive insights into complex political trends. The web interface, powered by Django, is currently under development. It will offer users an accessible platform to explore our findings and interact with the data.

Moving Forward

The next steps in the MPDP project include refining the machine learning models, enhancing our network analysis algorithms, and integrating more real-time data to keep our predictions current and relevant. I am also focusing on optimizing the user interface for a more engaging and informative experience.

Appendix A

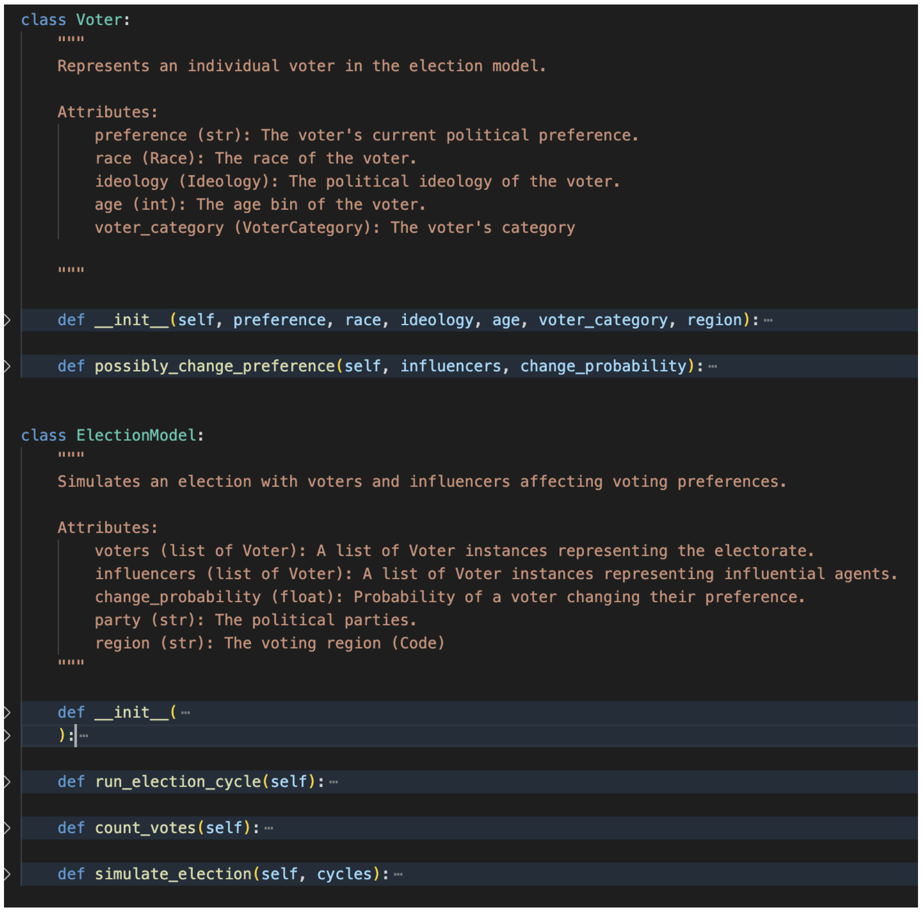


```python
class Race(Enum):
    MALAY = "Malay"
    CHINESE = "Chinese"
    INDIAN = "Indian"
    OTHERS = "Others"


class Ideology(Enum):
    LEFT = "Left"
    CENTRAL = "Central"
    RIGHT = "Right"


class VoterCategory(Enum):
    Army = "Early Voter (Army)"
    POLICE = "Early Voter (Police)"
    REGULAR = "Regular"
    OVERSEAS = "Overseas"
```

Figure A1. The three factors known to influence the voters' preferences in Malaysian elections.



```python
class Voter:
    """
    Represents an individual voter in the election model.

    Attributes:
        preference (str): The voter's current political preference.
        race (Race): The race of the voter.
        ideology (Ideology): The political ideology of the voter.
        age (int): The age bin of the voter.
        voter_category (VoterCategory): The voter's category

    """

    def __init__(self, preference, race, ideology, age, voter_category, region):···

    def possibly_change_preference(self, influencers, change_probability):···


class ElectionModel:
    """
    Simulates an election with voters and influencers affecting voting preferences.

    Attributes:
        voters (list of Voter): A list of Voter instances representing the electorate.
        influencers (list of Voter): A list of Voter instances representing influential agents.
        change_probability (float): Probability of a voter changing their preference.
        party (str): The political parties.
        region (str): The voting region (Code)
    """

    def __init__(···
    ):···

    def run_election_cycle(self):···

    def count_votes(self):···

    def simulate_election(self, cycles):···
```

Figure A2. The two classes used in the Agent-Based Model.