Machine Learning Final Project

Author: Oscar Hernandez Mata

"Predicting Voter Affiliation: Democrat, Republican, Independent, and Other in the U.S."

Abstract

Feedforward Neural Network (FNN) has been used with political datasets to predict election outcomes, classifying voter behaviors, and political sentiments such as identifying support for a particular candidate. In this project FNN was utilized on the ANES 2020 Time Series Study to predict voters party identification, more specifically whether an individual would identify with Democrats, Republicans, Independents or Other political party based on voter demographics, socio-economic status, and social issue opinions. FNN was well-suited for this project due to their capacity to capture learning complex patterns, their ability to generalize across heterogeneous data types, and scalability. The FNN model created a first input layer with 65 nodes, then a hidden layer with 128 nodes followed by a second hidden layer of 32 nodes and a final output layer of 4 nodes for Democrat, Republican, Independent and Other. The model achieved accuracy results of 91% and a good capacity of prediction for Democrats and Republicans. However, it did not perform well at predicting independent voters and people who identify with other political parties due to the lack of data for the model to be properly trained in those two categories.

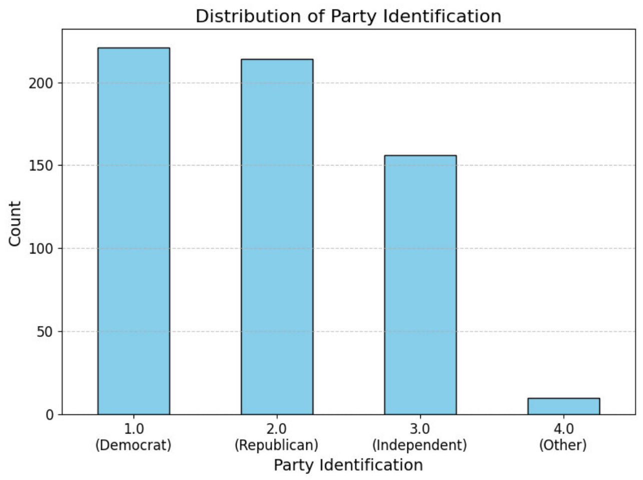
Introduction

Feedforward neural network is a foundational type of artificial neural network structured with multiple layers: an input layer, one or more hidden layers, and an output layer. Information flows in one direction (from input to output) without looping back, which distinguishes FNNs from recurrent neural networks. These networks are designed to learn patterns and relationships within data through a process of adjusting their free parameters, such as synaptic weights and biases, based on training data. FNNs commonly use supervised learning, where labeled input-output pairs guide the training process. The training is often achieved through the backpropagation algorithm, which operates in two phases: a forward phase, where the network processes input and calculates an error based on the difference between predicted and actual outputs, and a backward phase, where the error is used to adjust the network's parameters. This iterative process minimizes the error and improves the network's predictions. Feedforward neural networks are versatile, capable of handling tasks like pattern recognition, regression, and classification (Haykin, 1998).

In this project I used the ANES 2020 Time Series Study dataset. ANES, or the American National Election Studies, conducts socio-political research on public opinion and voting behavior during U.S. presidential elections (ANES 2020, p1). This dataset is essential for understanding public perceptions of the political, economic, cultural, and social realities of the United States. The goal of this project was to create a Feedforward neural network using the ANES dataset to predict voters party identification, more specifically whether an individual would identify with Democrats, Republicans, Independents or Other political party based on voter demographics, socio-economic status, and social issue opinions. FNN was well-suited for this project due to their capacity to capture non-linear relationships between features, which is essential when dealing with complex socio-political factors. In addition, FNNs can generalize across heterogeneous data types (e.g., categorical and numerical), making them effective at processing diverse variables. Moreover, FNNs can be scaled by adjusting the number of hidden layers, allowing for a balance between performance and computational efficiency (Heng-Tze Cheng 2016). Studies like mine help identify the concerns Americans face, their political and ideological tendencies, and how these factors may influence the outcome of a presidential election. Furthermore, they can be leveraged for political campaigns, enabling targeted messaging to specific populations to gain their support.

Methodology

Before conducting the Feedforward Neural Network, I performed data preprocessing on the ANES 2020 Time Series Study, which initially consisted of 8,280 rows and 1,771 columns. After careful analysis, I selected 66 relevant features that captured demographic information, political ideologies, financial data, and voting history. Cleaning the data involved removing responses such as "inapplicable," "don't know," and "refused," reducing the dataset to 601 rows while ensuring no missing values remained. The target variable, Party Identification, was refined by replacing category 5 ("Other") with category 4 for consistency. A bar plot of Party Identification revealed over 200 Democrat and Republican voters, slightly over 150 Independents, and fewer than 5 voters identifying with Other political parties, providing insight into the dataset's distribution.



Next, the target variable was one-hot encoded to represent each category numerically: 1.0 as [1. 0. 0. 0.], 2.0 as [0. 1. 0. 0.], 3.0 as [0. 0. 1. 0.], and 4.0 as [0. 0. 0. 1.]. This transformation facilitated compatibility with the neural network model. Subsequently, the dataset was split into training and testing sets, with the training set comprising 80% of the data and the testing set accounting for the remaining 20%, ensuring an appropriate balance for model evaluation and generalization.

The next step was creating an FNN. The model architecture included an input layer accepting 65 features:

Input Layer

$$\mathbf{a}^{(0)} = \mathbf{x}$$

(Where x is the input vector of size 65)

followed by a dense layer with 128 neurons and ReLU activation (Gupta & Raza 2020, p6):

Layer 1 (Dense with 128 neurons, ReLU activation)

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{a}^{(1)} = \max(0, \mathbf{z}^{(1)})$$

a hidden layer with 32 neurons and ReLU activation:

Layer 2 (Dense with 32 neurons, ReLU activation)

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}, \quad \mathbf{a}^{(3)} = \mathrm{softmax}(\mathbf{z}^{(3)})$$

and an output layer with 4 neurons and a softmax activation function for multi-class classification (Ajitesh Kumar 2022):

Layer 3 (Dense with 4 neurons, softmax activation)

$$\mathbf{z}^{(3)} = \mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)}, \quad \mathbf{a}^{(3)} = \mathrm{softmax}(\mathbf{z}^{(3)})$$

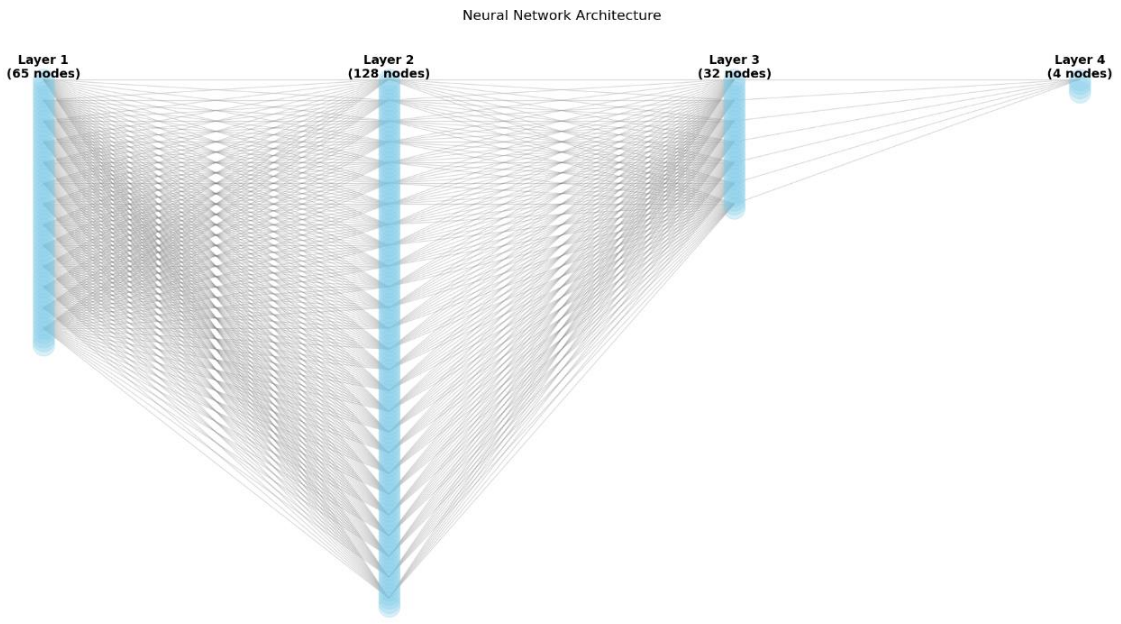
$$\mathrm{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{4}\exp(z_j)}$$

For i = 1, 2, 3, 4.

Final Output

$$\mathbf{a}^{(3)} = [a_1^{(3)}, a_2^{(3)}, a_3^{(3)}, a_4^{(3)}]$$

Where $a_i^{(3)}$ is the probability for class i.



Dropout layers with rates of 0.18 and 0.2 were added after the first and second dense layers, respectively, to reduce overfitting.

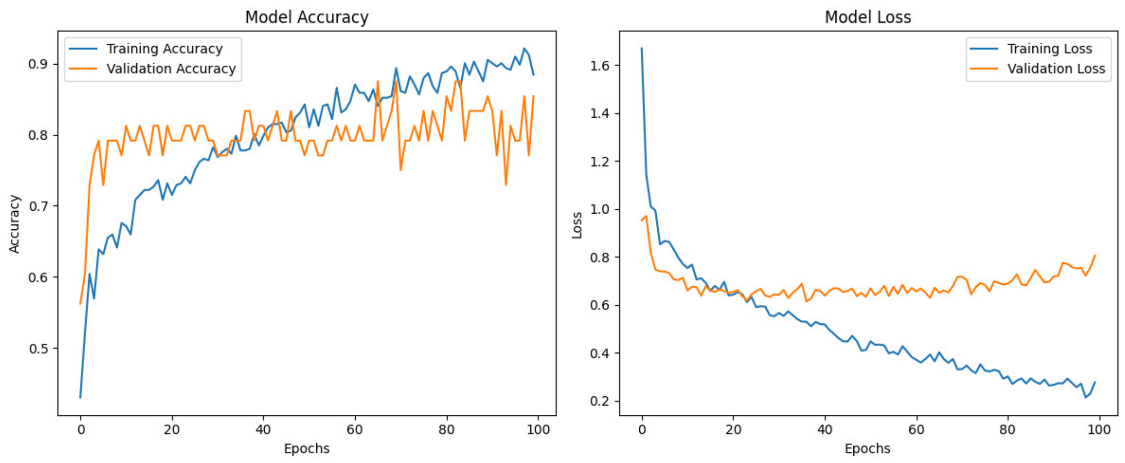
Experiment

Evaluation Metrics

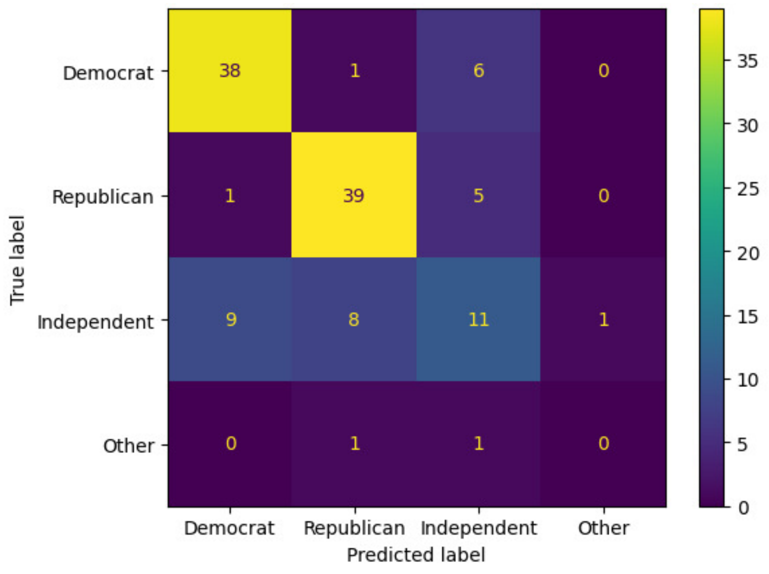
The model was compiled using the Adam optimizer with a learning rate of 0.001 and categorical crossentropy as the loss function, with accuracy set as the evaluation metric. It was trained over 100 epochs with a batch size of 32, utilizing 10% of the training data for validation. Various hyperparameter configurations were tested and fine-tuned to achieve the best possible performance.

Results

As I mentioned above, the Feedforward Neural Network was trained over 100 epochs with a batch size of 32, using 10% of the training data for validation. In the initial epochs, the model achieved a training accuracy of 39.4% and a loss of 1.8423, while the validation accuracy started at 56.25% with a loss of 0.9528. As training progressed, accuracy steadily improved, reaching a peak training accuracy of 92.3% by the 98th epoch and ending with 88.4% in the final epoch. Validation accuracy fluctuated, peaking at 87.5% in multiple epochs, and ended at 85.4%. The loss values for both training and validation consistently decreased in the early stages, with the training loss reducing from 1.8423 to 0.2821 and validation loss from 0.9528 to 0.8053. However, in later epochs, slight fluctuations in validation loss and accuracy indicated potential overfitting, as the training loss continued to decrease while validation loss slightly increased.



The next step in my experiment was to build a confusion matrix in order to get a detailed breakdown of the model's predictions versus the actual labels across the four classes: Democrat, Republican, Independent, and Other.



The diagonal entries represent correct predictions, while off-diagonal entries indicate misclassifications. For Democrats, 38 instances were correctly classified, while 7 were misclassified (1 as Republican and 6 as Independent), with no predictions in the Other category.

Similarly, for Republicans, 39 instances were correctly predicted, with 1 misclassified as Democrat and 5 as Independent, showing relatively strong performance for these two classes.

However, the Independent category presented a challenge, with only 11 correct classifications out of 29 instances. The model misclassified 9 as Democrat, 8 as Republican, and 1 as Other, indicating some confusion in distinguishing Independents from the other political affiliations. The Other category, representing the smallest group, has no correct classifications; out of 2 instances, 1 was misclassified as Republican and 1 as Independent.

The results indicated that the model performed well for the two largest classes, Democrat and Republican, due to the greater number of training examples for these groups. However, the performance declined for the smaller classes, Independent and Other, reflecting an imbalance in the dataset and the model's struggle to differentiate these minority categories.

The final step of the experiment involved creating a hypothetical voter with specific characteristics to test the model's predictive capabilities. Based on the provided attributes, the model predicted that this voter would identify with the Republican Party.

Conclusions

This project provided valuable insights into working with data and feedforward neural networks (FNNs). Through feature selection and data cleaning, I learned how to refine a large dataset to ensure the model receives meaningful inputs. Designing and experimenting with the FNN architecture deepened my understanding of its components, such as layers, activation functions, optimization algorithms, and techniques like dropout to prevent overfitting.

While the project demonstrated the potential of neural networks in classifying voter party identification, it also highlighted limitations, particularly the dataset's imbalance and lack of sufficient high-quality data. This shortcoming likely contributed to the model's inability to accurately predict Independents and voters identifying with Other political parties. Despite these challenges, the project showcased the model's ability to predict major political affiliations like Democrats and Republicans with reasonable accuracy.

Additionally, reviewing related research broadened my understanding of neural networks' expressive power and practical applications. Looking ahead, I aim to explore neural networks further, particularly in political datasets, by integrating advanced techniques like deep reinforcement learning (DRL). This approach could enable the analysis of dynamic decision-making processes, such as campaign strategy adaptation or voter turnout prediction, and open new avenues for optimizing political resource allocation.

References

American National Election Studies (ANES). ANES 2020 Time Series Study. University of Michigan, Center for Political Studies, 2022.

"Bar Plot Tutorial." *Matplotlib*, https://matplotlib.org/stable/gallery/index.html. Accessed 29 Nov. 2024.

Cheng, Heng-Tze, et al. "Wide & Deep Learning for Recommender Systems." *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, ACM, 2016.

Confusion Matrix Plot: Scikit-learn Documentation. "Plot Confusion Matrix." *Scikit-learn*, https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html. Accessed 21 Nov. 2024.

Daróczy, Bálint, et al. *Expressive Power of Outer Product Manifolds on Feed-Forward Neural Networks.* Institute for Computer Science and Control, Hungarian Academy of Sciences, 2018.

Emmert-Streib, Frank, et al. "An Introductory Review of Deep Learning for Prediction Models with Big Data." *Frontiers in Artificial Intelligence*, vol. 3, no. 4, 28 Feb. 2020, doi:10.3389/frai.2020.00004.

"Feedforward Neural Network Python Example." *Analytics Yogi*, https://vitalflux.com/feed-forward-neural-network-python-example/. Accessed 19 Nov. 2024.

GeeksforGeeks. "Visualization of Training and Validation Metrics in Deep Learning Models." GeeksforGeeks, https://www.geeksforgeeks.org. Accessed Nov. 2024.

Gupta, Tarun Kumar, and Khalid Raza. *Optimizing Deep Neural Network Architecture: A Tabu Search Based Approach.* Department of Computer Science, Jamia Millia Islamia, New Delhi, 2020.

Hagberg, Aric, Pieter Swart, and Daniel S. Chult. *NetworkX Documentation*. NetworkX, 2024, https://networkx.org/documentation/stable/.

Haykin, Simon. *Feedforward Neural Networks: An Introduction.* Wiley, 1998.

Hunter, J. D. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, vol. 9, no. 3, 2007, pp. 90–95.

"OneHotEncoder." *Scikit-learn Documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html. Accessed 30 Nov. 2024.

"Sequential Model Example." *Keras Documentation*, https://keras.io/guides/sequential_model/. Accessed 30 Nov. 2024.

TensorFlow. *Basic Machine Learning Workflow.* Google, https://www.tensorflow.org. Accessed Nov. 2024.

CODE

Machine Learning Final Project

Author: Oscar Hernandez Mata

Title: "Predicting Voter Affiliation: Democrat, Republican, Independent, and Other in the U.S."

In [1]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\13055\appdata\local\programs\pytho
n\python312\lib\site-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\13055\appdata\local\program
s\python\python312\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\13055\appdata\loca
l\programs\python\python312\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\13055\appdata\local\programs
\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\13055\appdata\local\progra
ms\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\13055\appdata\local\programs\pyt
hon\python312\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

```
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
pip install graphviz pydot
```

```
Requirement already satisfied: graphviz in c:\users\13055\appdata\local\programs\pyt
hon\python312\lib\site-packages (0.20.3)
Requirement already satisfied: pydot in c:\users\13055\appdata\local\programs\python
\python312\lib\site-packages (3.0.2)
Requirement already satisfied: pyparsing>=3.0.9 in c:\users\13055\appdata\local\prog
rams\python\python312\lib\site-packages (from pydot) (3.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:
```
pip install networkx
```

```
Requirement already satisfied: networkx in c:\users\13055\appdata\local\programs\pyt
hon\python312\lib\site-packages (3.4.2)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [4]:
```
import pandas as pd

file_path = 'anes_timeseries_2020_csv_20220210.csv'
anes = pd.read_csv(file_path)

rows, columns = anes.shape
```

```
print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")
```

```
C:\Users\13055\AppData\Local\Temp\ipykernel_18404\2491514130.py:7: DtypeWarning: Col
umns (15,17,18,19,21,22,23,25,26,27,29,30,31,33,34,35,37,38,1508,1509) have mixed ty
pes. Specify dtype option on import or set low_memory=False.
  anes = pd.read_csv(file_path)
Number of rows: 8280
Number of columns: 1771
```

In [5]:
```
# Renaming dictionary
rename_dict = {
    "V201600": "Gender",
    "V201005": "Attention_to_Politics",
    "V201006": "Attention_to_Political_Campaigns",
    "V201018": "Party_of_Registration",
    "V201020": "Voted_in_Presidential_Primary",
    "V201032": "Intend_to_Vote_for_President",
    "V201033": "Voting_for_President",
    "V201034": "Strength_of_Candidate_Preference",
    "V201041": "Intend_to_Vote_for_US_House",
    "V201042": "Candidate_Intend_to_Vote_for_US_House",
    "V201103": "Voted_for_President_in_2016",
    "V201104": "Voted_for_President_in_2012",
    "V201200": "Ideology_Scale",
    "V201206": "Placement_of_Democratic_Party",
    "V201207": "Placement_of_Republican_Party",
    "V201208": "Democratic_Candidate_Strong_Leadership",
    "V201209": "Democratic_Candidate_Cares_for_People",
    "V201212": "Republican_Candidate_Strong_Leadership",
    "V201213": "Republican_Candidate_Cares_for_People",
    "V201228": "Party_Identification",
    "V201235": "Government_Wastes_Tax_Money",
    "V201239": "Party_Better_Handling_Economy",
    "V201240": "Party_Better_Handling_Health_Care",
    "V201241": "Party_Better_Handling_Immigration",
    "V201242": "Party_Better_Handling_Taxes",
    "V201243": "Party_Better_Handling_Environment",
    "V201300": "Federal_Spending_Social_Security",
    "V201303": "Federal_Spending_Public_Schools",
    "V201306": "Federal_Spending_Border_Security",
    "V201309": "Federal_Spending_Crime",
    "V201312": "Federal_Spending_Welfare_Programs",
    "V201318": "Federal_Spending_Aid_to_Poor",
    "V201338": "Position_on_Abortion_Joe_Biden",
    "V201339": "Position_on_Abortion_Donald_Trump",
    "V201340": "Abortion_Rights_Supreme_Court",
    "V201343": "Favor_Oppose_Death_Penalty",
    "V201346": "US_Position_in_World",
    "V201347": "Country_Better_Staying_Out",
    "V201350": "Use_of_Military_Force",
    "V201351": "Accuracy_of_Vote_Count",
    "V201357": "Favor_Oppose_Voter_ID",
    "V201377": "Trust_in_News_Media",
    "V201398": "Income_Differences_Larger",
    "V201412": "Favor_Protect_LGBT_Job_Discrimination",
    "V201415": "Allow_Gay_Adoption",
    "V201416": "Position_on_Gay_Marriage",
    "V201417": "Policy_Toward_Unauthorized_Immigrants",
    "V201418": "Favor_Oppose_Ending_Birthright_Citizenship",
    "V201424": "Favor_Oppose_Border_Wall",
    "V201435": "Respondent_Religion",
    "V201453": "Religious_Service_Attendance",
    "V201502": "Financial_Situation_Compared_to_Last_Year",
    "V201503": "Financial_Situation_Next_Year",
    "V201508": "Marital_Status",
    "V201510": "Education_Level",
    "V201516": "Active_Duty_in_Armed_Forces",
    "V201529": "Employment_Status",
    "V201544": "Household_Union_Member",
    "V201546": "Hispanic_Origin",
    "V201553": "Parents_Native_Status",
    "V201563": "Spouse_Hispanic_Origin",
    "V201566": "Spouse_Gender",
    "V201567": "Number_of_Children_in_Household",
    "V201575": "State_or_Country_of_Upbringing",
    "V201596": "Job_Loss_in_Household",
    "V201601": "Sexual_Orientation",
    "V201103": "Voted_for_President_in_2016"
}

# Renaming columns in the DataFrame
anes_selected = anes[rename_dict.keys()]
anes_selected.rename(columns=rename_dict, inplace=True)

# Verify renaming
print(anes_selected.columns)
```

```
Index(['Gender', 'Attention_to_Politics', 'Attention_to_Political_Campaigns',
       'Party_of_Registration', 'Voted_in_Presidential_Primary',
       'Intend_to_Vote_for_President', 'Voting_for_President',
       'Strength_of_Candidate_Preference', 'Intend_to_Vote_for_US_House',
       'Candidate_Intend_to_Vote_for_US_House', 'Voted_for_President_in_2016',
       'Voted_for_President_in_2012', 'Ideology_Scale',
       'Placement_of_Democratic_Party', 'Placement_of_Republican_Party',
       'Democratic_Candidate_Strong_Leadership',
       'Democratic_Candidate_Cares_for_People',
       'Republican_Candidate_Strong_Leadership',
       'Republican_Candidate_Cares_for_People', 'Party_Identification',
       'Government_Wastes_Tax_Money', 'Party_Better_Handling_Economy',
       'Party_Better_Handling_Health_Care',
       'Party_Better_Handling_Immigration', 'Party_Better_Handling_Taxes',
       'Party_Better_Handling_Environment', 'Federal_Spending_Social_Security',
       'Federal_Spending_Public_Schools', 'Federal_Spending_Border_Security',
       'Federal_Spending_Crime', 'Federal_Spending_Welfare_Programs',
       'Federal_Spending_Aid_to_Poor', 'Position_on_Abortion_Joe_Biden',
       'Position_on_Abortion_Donald_Trump', 'Abortion_Rights_Supreme_Court',
       'Favor_Oppose_Death_Penalty', 'US_Position_in_World',
       'Country_Better_Staying_Out', 'Use_of_Military_Force',
       'Accuracy_of_Vote_Count', 'Favor_Oppose_Voter_ID',
       'Trust_in_News_Media', 'Income_Differences_Larger',
       'Favor_Protect_LGBT_Job_Discrimination', 'Allow_Gay_Adoption',
       'Position_on_Gay_Marriage', 'Policy_Toward_Unauthorized_Immigrants',
       'Favor_Oppose_Ending_Birthright_Citizenship',
       'Favor_Oppose_Border_Wall', 'Respondent_Religion',
       'Religious_Service_Attendance',
       'Financial_Situation_Compared_to_Last_Year',
       'Financial_Situation_Next_Year', 'Marital_Status', 'Education_Level',
       'Active_Duty_in_Armed_Forces', 'Employment_Status',
       'Household_Union_Member', 'Hispanic_Origin', 'Parents_Native_Status',
       'Spouse_Hispanic_Origin', 'Spouse_Gender',
       'Number_of_Children_in_Household', 'State_or_Country_of_Upbringing',
       'Job_Loss_in_Household', 'Sexual_Orientation'],
      dtype='object')
```

```
C:\Users\13055\AppData\Local\Temp\ipykernel_18404\4257259139.py:95: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  anes_selected.rename(columns=rename_dict, inplace=True)
```

In [6]:
```python
# Number of rows in anes_selected
number_of_rows = anes_selected.shape[0]

print(f"The dataset has {number_of_rows} rows.")

#Number of columns in the dataset
number_of_columns = anes_selected.shape[1]

print(f"The dataset has {number_of_columns} columns.")
```

```
The dataset has 8280 rows.
The dataset has 66 columns.
```

```
In [7]: # Missing values in the dataset
        total_missing_values = anes_selected.isnull().sum().sum()

        print(f"The dataset has {total_missing_values} missing values.")
```

```
The dataset has 0 missing values.
```

```
In [8]: #Eliminating usless observations
        anes_selected_cleaned = anes_selected[(anes_selected != -1) & (anes_selected != -8)

        #Number of rows in the cleaned dataset
        num_rows_cleaned = anes_selected_cleaned.shape[0]

        print(f"Number of rows left in the cleaned dataset: {num_rows_cleaned}")
```

```
Number of rows left in the cleaned dataset: 601
```

```
In [9]: print(anes_selected_cleaned).head()
```

```
      Gender  Attention_to_Politics  Attention_to_Political_Campaigns  \
1        2.0                    4.0                               3.0
18       2.0                    3.0                               1.0
34       2.0                    2.0                               1.0
46       1.0                    1.0                               1.0
68       2.0                    1.0                               2.0
...      ...                    ...                               ...
8139     1.0                    2.0                               3.0
8179     2.0                    2.0                               1.0
8187     2.0                    2.0                               1.0
8198     1.0                    1.0                               1.0
8257     2.0                    2.0                               1.0

      Party_of_Registration  Voted_in_Presidential_Primary  \
1                       4.0                            1.0
18                      5.0                            2.0
34                      4.0                            2.0
46                      1.0                            1.0
68                      1.0                            1.0
...                     ...                            ...
8139                    4.0                            2.0
8179                    2.0                            1.0
8187                    4.0                            2.0
8198                    1.0                            1.0
8257                    1.0                            1.0

      Intend_to_Vote_for_President  Voting_for_President  \
1                              1.0                   3.0
18                             1.0                   1.0
34                             1.0                   1.0
46                             1.0                   1.0
68                             1.0                   1.0
...                            ...                   ...
8139                           1.0                   2.0
8179                           1.0                   2.0
8187                           1.0                   1.0
8198                           1.0                   1.0
8257                           1.0                   1.0

      Strength_of_Candidate_Preference  Intend_to_Vote_for_US_House  \
1                                  1.0                          1.0
18                                 1.0                          1.0
34                                 1.0                          1.0
46                                 1.0                          1.0
68                                 1.0                          1.0
...                                ...                          ...
8139                               2.0                          1.0
8179                               1.0                          1.0
8187                               1.0                          1.0
8198                               1.0                          1.0
8257                               1.0                          1.0

      Candidate_Intend_to_Vote_for_US_House  ...  Employment_Status  \
1                                       2.0  ...                1.0
18                                      1.0  ...                1.0
34                                      1.0  ...                2.0
```

```
46                                  1.0  ...               1.0
68                                  1.0  ...               1.0
...                                 ...  ...               ...
8139                               11.0  ...               2.0
8179                               11.0  ...               1.0
8187                                1.0  ...               9.0
8198                                1.0  ...               1.0
8257                                1.0  ...               9.0

      Household_Union_Member  Hispanic_Origin  Parents_Native_Status  \
1                        2.0              2.0                    1.0
18                       2.0              2.0                    1.0
34                       2.0              2.0                    1.0
46                       1.0              2.0                    1.0
68                       2.0              2.0                    1.0
...                      ...              ...                    ...
8139                     2.0              1.0                    1.0
8179                     2.0              2.0                    1.0
8187                     2.0              2.0                    2.0
8198                     2.0              2.0                    1.0
8257                     2.0              2.0                    1.0

      Spouse_Hispanic_Origin  Spouse_Gender  Number_of_Children_in_Household  \
1                        2.0            1.0                              1.0
18                       2.0            1.0                              0.0
34                       2.0            1.0                              0.0
46                       2.0            2.0                              0.0
68                       2.0            1.0                              2.0
...                      ...            ...                              ...
8139                     2.0            2.0                              2.0
8179                     2.0            1.0                              1.0
8187                     2.0            1.0                              0.0
8198                     2.0            2.0                              0.0
8257                     2.0            1.0                              0.0

      State_or_Country_of_Upbringing  Job_Loss_in_Household  \
1                                6.0                    1.0
18                              17.0                    2.0
34                              36.0                    2.0
46                              42.0                    2.0
68                               8.0                    2.0
...                              ...                    ...
8139                            48.0                    2.0
8179                            40.0                    2.0
8187                             6.0                    2.0
8198                            51.0                    1.0
8257                            37.0                    1.0

      Sexual_Orientation
1                    1.0
18                   1.0
34                   1.0
46                   1.0
68                   1.0
...                  ...
8139                 1.0
```

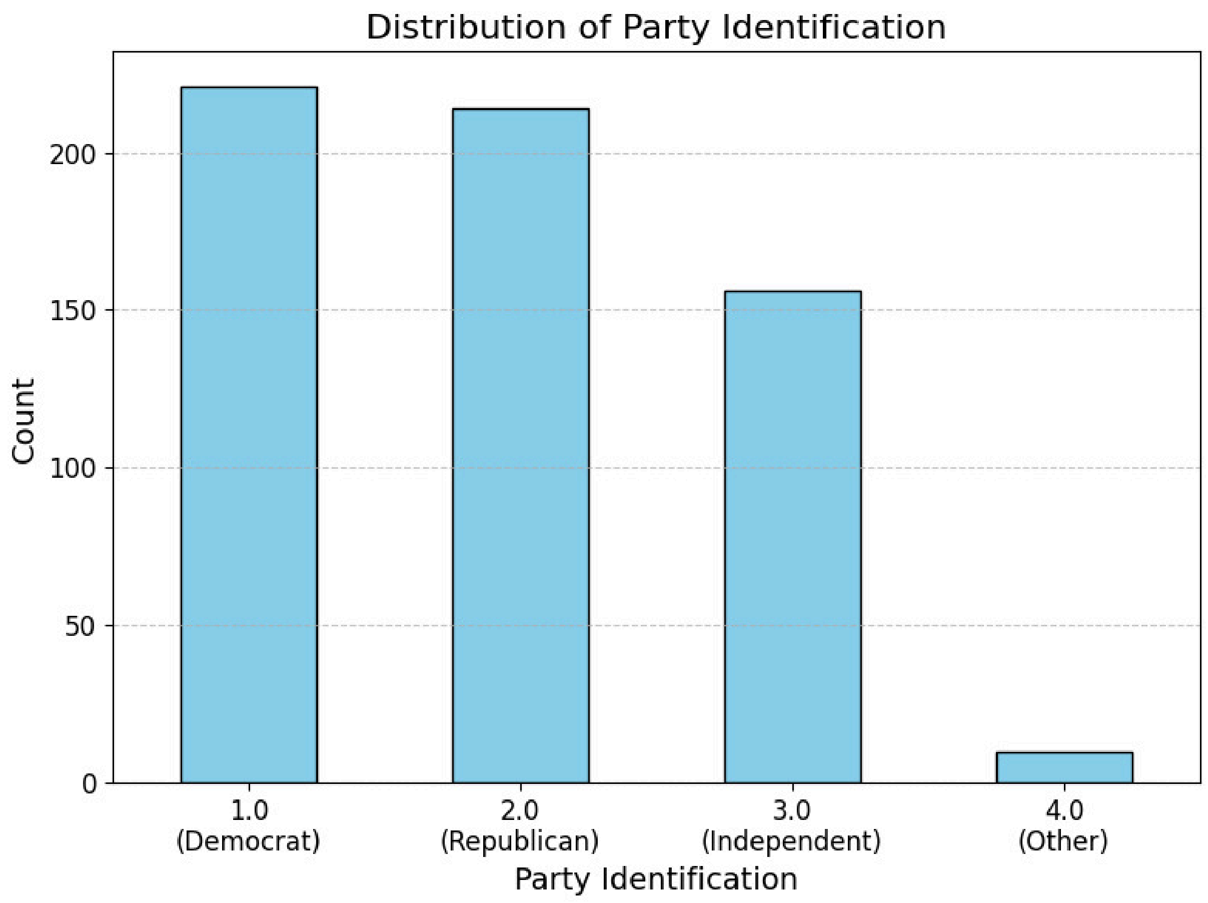
```
8179                  1.0
8187                  1.0
8198                  1.0
8257                  1.0

[601 rows x 66 columns]
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[9], line 1
----> 1 print(anes_selected_cleaned).head()

AttributeError: 'NoneType' object has no attribute 'head'
```

In [10]:
```
# Convert all columns back to float32
anes_selected_cleaned = anes_selected_cleaned.astype('float32')
```

In [11]:
```
# Replace 5 with 4 in Party_Identification
anes_selected_cleaned['Party_Identification'] = anes_selected_cleaned['Party_Identi

unique_values = anes_selected_cleaned['Party_Identification'].unique()

print(unique_values)
```

```
[4. 1. 2. 3.]
```

In [22]:
```
# Citation: ("Bar Plot Tutorial." Matplotlib, 11, 2024)

import matplotlib.pyplot as plt

# Convert Party_Identification back to integers
anes_selected_cleaned['Party_Identification'] = anes_selected_cleaned['Party_Identi

# Count the occurrences of each unique value in Party Identification
party_counts = anes_selected_cleaned['Party_Identification'].value_counts()

# Define custom labels for the x-axis
labels = ['1.0\n(Democrat)', '2.0\n(Republican)', '3.0\n(Independent)', '4.0\n(Othe

# Bar plot
plt.figure(figsize=(8, 6))
party_counts.sort_index().plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribution of Party Identification', fontsize=16)
plt.xlabel('Party Identification', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(ticks=range(len(labels)), labels=labels, fontsize=12, rotation=0)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [12]:
```python
# Citation: ("OneHotEncoder." Scikit-learn Documentation, 11, 2024)

import numpy as np

# Extract the target variable y
y = anes_selected_cleaned['Party_Identification'].values

# One-hot encode the target variable
from tensorflow.keras.utils import to_categorical
y_encoded = to_categorical(y - 1)  # Subtract 1 to shift classes to start from 0

print(y_encoded)

# Unique labels and their one-hot encodings
unique_labels = np.unique(y)
for label in unique_labels:
    print(f"Label {label} -> One-hot: {to_categorical([label - 1], num_classes=4)}"
[[0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 ...
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]]
Label 1.0 -> One-hot: [[1. 0. 0. 0.]]
Label 2.0 -> One-hot: [[0. 1. 0. 0.]]
Label 3.0 -> One-hot: [[0. 0. 1. 0.]]
Label 4.0 -> One-hot: [[0. 0. 0. 1.]]
```

In [13]:
```
# Define X by dropping the target variable 'Party_Identification'
X = anes_selected_cleaned.drop(columns=['Party_Identification'])

# Shape of X
print("Shape of X:", X.shape)
```

```
Shape of X: (601, 65)
```

In [14]:
```
from sklearn.model_selection import train_test_split

# Training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, ra

# Shapes of the splits
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (480, 65)
X_test shape: (121, 65)
y_train shape: (480, 4)
y_test shape: (121, 4)
```

In [15]:
```
# Citation: "Sequential Model Example." Keras Documentation, 11, 2024)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

# FFN model
model = Sequential([
    Dense(128, input_dim=X_train.shape[1], activation='relu'), # Input layer
    Dropout(0.18),
    Dense(32, activation='relu'),  # Hidden layer
    Dropout(0.2),
    Dense(4, activation='softmax')  # Output layer for 4 classes
])

model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',

# Training the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0
C:\Users\13055\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\l
ayers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
nt to a layer. When using Sequential models, prefer using an `Input(shape)` object a
s the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 9s 56ms/step - accuracy: 0.3940 - loss: 1.8423 - val_accu
racy: 0.5625 - val_loss: 0.9528
Epoch 2/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.4969 - loss: 1.1807 - val_accu
racy: 0.6042 - val_loss: 0.9706
Epoch 3/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6143 - loss: 1.0208 - val_accur
acy: 0.7292 - val_loss: 0.8160
Epoch 4/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5664 - loss: 0.9574 - val_accur
acy: 0.7708 - val_loss: 0.7472
Epoch 5/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6348 - loss: 0.8645 - val_accur
acy: 0.7917 - val_loss: 0.7397
Epoch 6/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6564 - loss: 0.8183 - val_accur
acy: 0.7292 - val_loss: 0.7389
Epoch 7/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6397 - loss: 0.8959 - val_accur
acy: 0.7917 - val_loss: 0.7325
Epoch 8/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6919 - loss: 0.7909 - val_accur
acy: 0.7917 - val_loss: 0.7067
Epoch 9/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6504 - loss: 0.7884 - val_accur
acy: 0.7917 - val_loss: 0.7034
Epoch 10/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6398 - loss: 0.7856 - val_accur
acy: 0.7708 - val_loss: 0.7131
Epoch 11/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6812 - loss: 0.7217 - val_accur
acy: 0.8125 - val_loss: 0.6602
Epoch 12/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6462 - loss: 0.7768 - val_accur
acy: 0.7917 - val_loss: 0.6756
Epoch 13/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7106 - loss: 0.7014 - val_accur
acy: 0.7917 - val_loss: 0.6735
Epoch 14/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7017 - loss: 0.6948 - val_accur
acy: 0.8125 - val_loss: 0.6383
Epoch 15/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7220 - loss: 0.6818 - val_accur
acy: 0.7917 - val_loss: 0.6783
Epoch 16/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7208 - loss: 0.6621 - val_accur
acy: 0.7708 - val_loss: 0.6600
Epoch 17/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7411 - loss: 0.6565 - val_accur
acy: 0.8125 - val_loss: 0.6537
Epoch 18/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7307 - loss: 0.6501 - val_accur
acy: 0.8125 - val_loss: 0.6655
Epoch 19/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7153 - loss: 0.6401 - val_accur
```

```
acy: 0.7708 - val_loss: 0.6571
Epoch 20/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7206 - loss: 0.6543 - val_accur
acy: 0.8125 - val_loss: 0.6506
Epoch 21/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7470 - loss: 0.6288 - val_accur
acy: 0.7917 - val_loss: 0.6541
Epoch 22/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7442 - loss: 0.6535 - val_accur
acy: 0.7917 - val_loss: 0.6617
Epoch 23/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7034 - loss: 0.7062 - val_accur
acy: 0.7917 - val_loss: 0.6368
Epoch 24/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7276 - loss: 0.6564 - val_accur
acy: 0.8125 - val_loss: 0.6185
Epoch 25/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7453 - loss: 0.6019 - val_accur
acy: 0.8125 - val_loss: 0.6434
Epoch 26/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7538 - loss: 0.5956 - val_accur
acy: 0.7917 - val_loss: 0.6567
Epoch 27/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7507 - loss: 0.6022 - val_accur
acy: 0.8125 - val_loss: 0.6663
Epoch 28/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8024 - loss: 0.5550 - val_accur
acy: 0.8125 - val_loss: 0.6391
Epoch 29/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7561 - loss: 0.5808 - val_accur
acy: 0.7917 - val_loss: 0.6335
Epoch 30/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7723 - loss: 0.5707 - val_accur
acy: 0.7917 - val_loss: 0.6440
Epoch 31/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7727 - loss: 0.5465 - val_accur
acy: 0.7708 - val_loss: 0.6410
Epoch 32/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7701 - loss: 0.5465 - val_accur
acy: 0.7708 - val_loss: 0.6621
Epoch 33/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7901 - loss: 0.5704 - val_accur
acy: 0.7708 - val_loss: 0.6286
Epoch 34/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8064 - loss: 0.4758 - val_accur
acy: 0.7917 - val_loss: 0.6502
Epoch 35/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7956 - loss: 0.5360 - val_accur
acy: 0.7917 - val_loss: 0.6656
Epoch 36/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7750 - loss: 0.5293 - val_accur
acy: 0.7917 - val_loss: 0.6884
Epoch 37/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7714 - loss: 0.5510 - val_accur
acy: 0.8333 - val_loss: 0.6146
Epoch 38/100
```

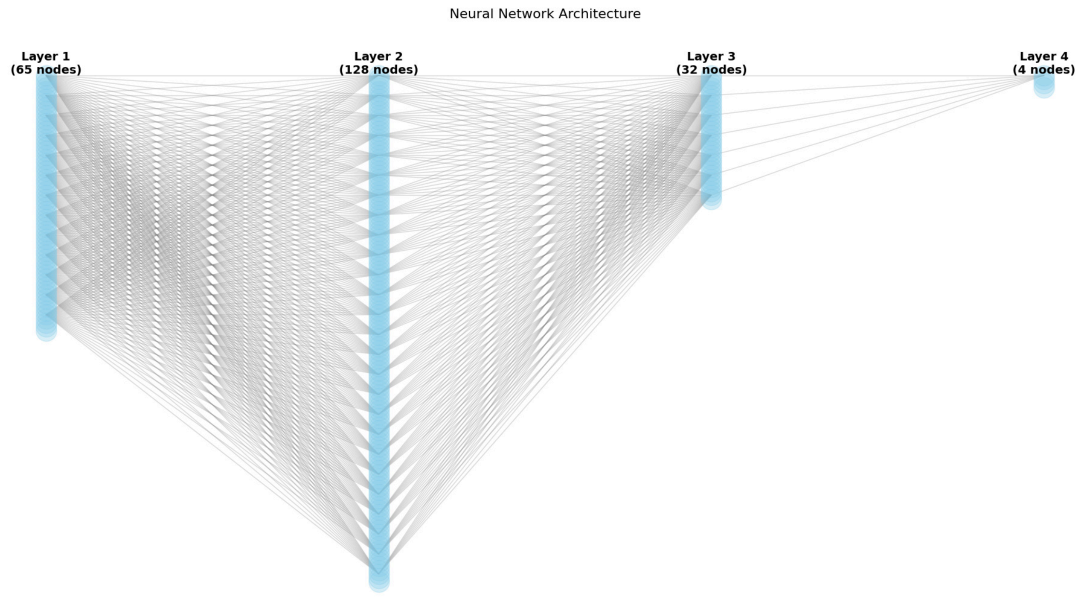
```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7820 - loss: 0.5125 - val_accur
acy: 0.8333 - val_loss: 0.6264
Epoch 39/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8108 - loss: 0.5296 - val_accur
acy: 0.7917 - val_loss: 0.6629
Epoch 40/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7715 - loss: 0.5398 - val_accur
acy: 0.8125 - val_loss: 0.6589
Epoch 41/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7887 - loss: 0.5241 - val_accur
acy: 0.8125 - val_loss: 0.6386
Epoch 42/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8210 - loss: 0.4817 - val_accur
acy: 0.7917 - val_loss: 0.6582
Epoch 43/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8223 - loss: 0.4511 - val_accur
acy: 0.8125 - val_loss: 0.6694
Epoch 44/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8134 - loss: 0.4562 - val_accur
acy: 0.8333 - val_loss: 0.6679
Epoch 45/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8363 - loss: 0.4405 - val_accur
acy: 0.7917 - val_loss: 0.6538
Epoch 46/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8141 - loss: 0.4394 - val_accur
acy: 0.7917 - val_loss: 0.6577
Epoch 47/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7917 - loss: 0.4993 - val_accur
acy: 0.8333 - val_loss: 0.6676
Epoch 48/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8299 - loss: 0.4275 - val_accur
acy: 0.7917 - val_loss: 0.6364
Epoch 49/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8332 - loss: 0.4043 - val_accur
acy: 0.7917 - val_loss: 0.6494
Epoch 50/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8515 - loss: 0.4000 - val_accur
acy: 0.7708 - val_loss: 0.6339
Epoch 51/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7943 - loss: 0.4769 - val_accur
acy: 0.7917 - val_loss: 0.6691
Epoch 52/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8263 - loss: 0.4274 - val_accur
acy: 0.7917 - val_loss: 0.6408
Epoch 53/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8110 - loss: 0.4337 - val_accur
acy: 0.7708 - val_loss: 0.6540
Epoch 54/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8514 - loss: 0.3970 - val_accur
acy: 0.7708 - val_loss: 0.6798
Epoch 55/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8544 - loss: 0.3713 - val_accur
acy: 0.7917 - val_loss: 0.6369
Epoch 56/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8489 - loss: 0.3682 - val_accur
acy: 0.7917 - val_loss: 0.6747
```

```
Epoch 57/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8646 - loss: 0.3991 - val_accur
acy: 0.8125 - val_loss: 0.6456
Epoch 58/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7998 - loss: 0.4602 - val_accur
acy: 0.7917 - val_loss: 0.6838
Epoch 59/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8347 - loss: 0.4064 - val_accur
acy: 0.8125 - val_loss: 0.6490
Epoch 60/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8468 - loss: 0.3756 - val_accur
acy: 0.7917 - val_loss: 0.6706
Epoch 61/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8845 - loss: 0.3221 - val_accur
acy: 0.7917 - val_loss: 0.6555
Epoch 62/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8460 - loss: 0.4020 - val_accur
acy: 0.8125 - val_loss: 0.6683
Epoch 63/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8467 - loss: 0.3964 - val_accur
acy: 0.7917 - val_loss: 0.6518
Epoch 64/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8572 - loss: 0.3587 - val_accur
acy: 0.7917 - val_loss: 0.6290
Epoch 65/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8532 - loss: 0.3632 - val_accur
acy: 0.7917 - val_loss: 0.6715
Epoch 66/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8413 - loss: 0.3976 - val_accur
acy: 0.8750 - val_loss: 0.6504
Epoch 67/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8430 - loss: 0.3835 - val_accur
acy: 0.7917 - val_loss: 0.6602
Epoch 68/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8425 - loss: 0.3760 - val_accur
acy: 0.8125 - val_loss: 0.6518
Epoch 69/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8400 - loss: 0.3954 - val_accur
acy: 0.8333 - val_loss: 0.6786
Epoch 70/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8932 - loss: 0.3449 - val_accur
acy: 0.8750 - val_loss: 0.7170
Epoch 71/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8940 - loss: 0.2743 - val_accur
acy: 0.7500 - val_loss: 0.7171
Epoch 72/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8444 - loss: 0.3468 - val_accur
acy: 0.7917 - val_loss: 0.7042
Epoch 73/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8824 - loss: 0.3185 - val_accur
acy: 0.7917 - val_loss: 0.6441
Epoch 74/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8740 - loss: 0.3189 - val_accur
acy: 0.8125 - val_loss: 0.6740
Epoch 75/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8749 - loss: 0.3163 - val_accur
```

```
acy: 0.7917 - val_loss: 0.6913
Epoch 76/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8807 - loss: 0.3275 - val_accur
acy: 0.8333 - val_loss: 0.6828
Epoch 77/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8779 - loss: 0.3204 - val_accur
acy: 0.7917 - val_loss: 0.6570
Epoch 78/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8818 - loss: 0.3254 - val_accur
acy: 0.8333 - val_loss: 0.6978
Epoch 79/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8718 - loss: 0.3026 - val_accur
acy: 0.8125 - val_loss: 0.6924
Epoch 80/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9140 - loss: 0.2611 - val_accur
acy: 0.7917 - val_loss: 0.6842
Epoch 81/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9190 - loss: 0.2524 - val_accur
acy: 0.8542 - val_loss: 0.6886
Epoch 82/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9057 - loss: 0.2422 - val_accur
acy: 0.8333 - val_loss: 0.7022
Epoch 83/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8853 - loss: 0.2827 - val_accur
acy: 0.8750 - val_loss: 0.7280
Epoch 84/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8725 - loss: 0.2750 - val_accur
acy: 0.8750 - val_loss: 0.6855
Epoch 85/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9116 - loss: 0.2655 - val_accu
racy: 0.7917 - val_loss: 0.6808
Epoch 86/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8819 - loss: 0.3037 - val_accur
acy: 0.8333 - val_loss: 0.7121
Epoch 87/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8942 - loss: 0.3040 - val_accur
acy: 0.8333 - val_loss: 0.7457
Epoch 88/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9103 - loss: 0.2210 - val_accur
acy: 0.8333 - val_loss: 0.7186
Epoch 89/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8623 - loss: 0.3152 - val_accur
acy: 0.8333 - val_loss: 0.6940
Epoch 90/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8927 - loss: 0.2782 - val_accur
acy: 0.8542 - val_loss: 0.6961
Epoch 91/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9045 - loss: 0.2769 - val_accur
acy: 0.8333 - val_loss: 0.7179
Epoch 92/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8964 - loss: 0.2627 - val_accur
acy: 0.7708 - val_loss: 0.7219
Epoch 93/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9066 - loss: 0.2435 - val_accur
acy: 0.8333 - val_loss: 0.7749
Epoch 94/100
```

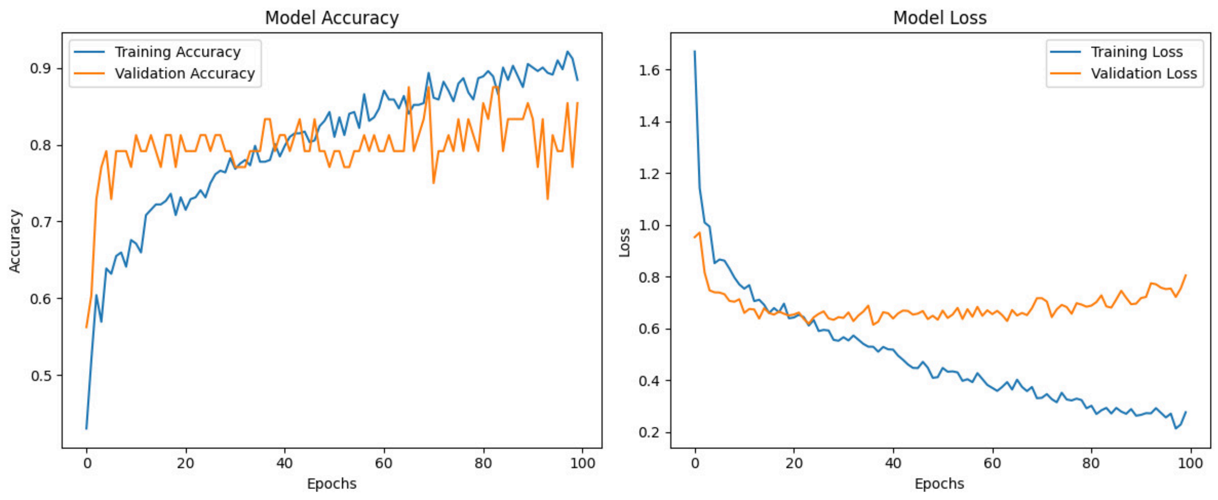
```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9144 - loss: 0.2643 - val_accur
acy: 0.7292 - val_loss: 0.7711
Epoch 95/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8858 - loss: 0.2908 - val_accur
acy: 0.8125 - val_loss: 0.7578
Epoch 96/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9124 - loss: 0.2461 - val_accur
acy: 0.7917 - val_loss: 0.7522
Epoch 97/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8893 - loss: 0.2923 - val_accur
acy: 0.7917 - val_loss: 0.7544
Epoch 98/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9234 - loss: 0.2200 - val_accur
acy: 0.8542 - val_loss: 0.7220
Epoch 99/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9122 - loss: 0.2239 - val_accur
acy: 0.7708 - val_loss: 0.7547
Epoch 100/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8845 - loss: 0.2821 - val_accur
acy: 0.8542 - val_loss: 0.8053
```

In [16]:
```python
# Citation: (Hagberg, Aric, Pieter Swart, and Daniel S. Chult, 2024)
# Citation: Hunter, J. D, 2007

import matplotlib.pyplot as plt
import networkx as nx

# Function to draw the neural network with improved spacing
def draw_neural_network_v2(layer_sizes):
    G = nx.DiGraph()

    # Nodes for each layer
    for i, size in enumerate(layer_sizes):
        for j in range(size):
            G.add_node(f"L{i}N{j}", layer=i)

    # Edges between layers
    for i in range(len(layer_sizes) - 1):
        for j in range(layer_sizes[i]):
            for k in range(layer_sizes[i + 1]):
                if j % 5 == 0 and k % 5 == 0:  # Reduce visible edges
                    G.add_edge(f"L{i}N{j}", f"L{i+1}N{k}")

    # Positions for nodes
    pos = {}
    x_offset = 12  # Increased horizontal spacing
    y_spacing = 3  # Increased vertical spacing
    for i, size in enumerate(layer_sizes):
        for j in range(size):
            pos[f"L{i}N{j}"] = (i * x_offset, -j * y_spacing)

    # Network
    plt.figure(figsize=(20, 10))  # Larger figure size
    nx.draw(
        G,
        pos,
        with_labels=False,  # Hide individual node labels
        node_size=600,
        node_color="skyblue",
        edge_color="gray",
        alpha=0.3,  # Lighter edge transparency
        arrows=False,
    )

    # Layer labels
    for i, size in enumerate(layer_sizes):
        plt.text(
            i * x_offset,
            1.5,  # Adjust vertical placement of labels
            f"Layer {i + 1}\n({size} nodes)",
            horizontalalignment="center",
            fontsize=14,
            color="black",
            weight="bold",
        )

    plt.title("Neural Network Architecture", fontsize=16)
    plt.show()

# Layer sizes
layer_sizes = [65, 128, 32, 4]  # Input layer, hidden layers, output layer

# Neural network
draw_neural_network_v2(layer_sizes)
```



In [17]:
```
# Citation: (GeeksforGeeks, 11/2024)
# Citation: (TensorFlow. Basic Machine Learning Workflow, 11/2024)

import matplotlib.pyplot as plt

# Plot of the training and validation accuracy
plt.figure(figsize=(12, 5))
```

```
# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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

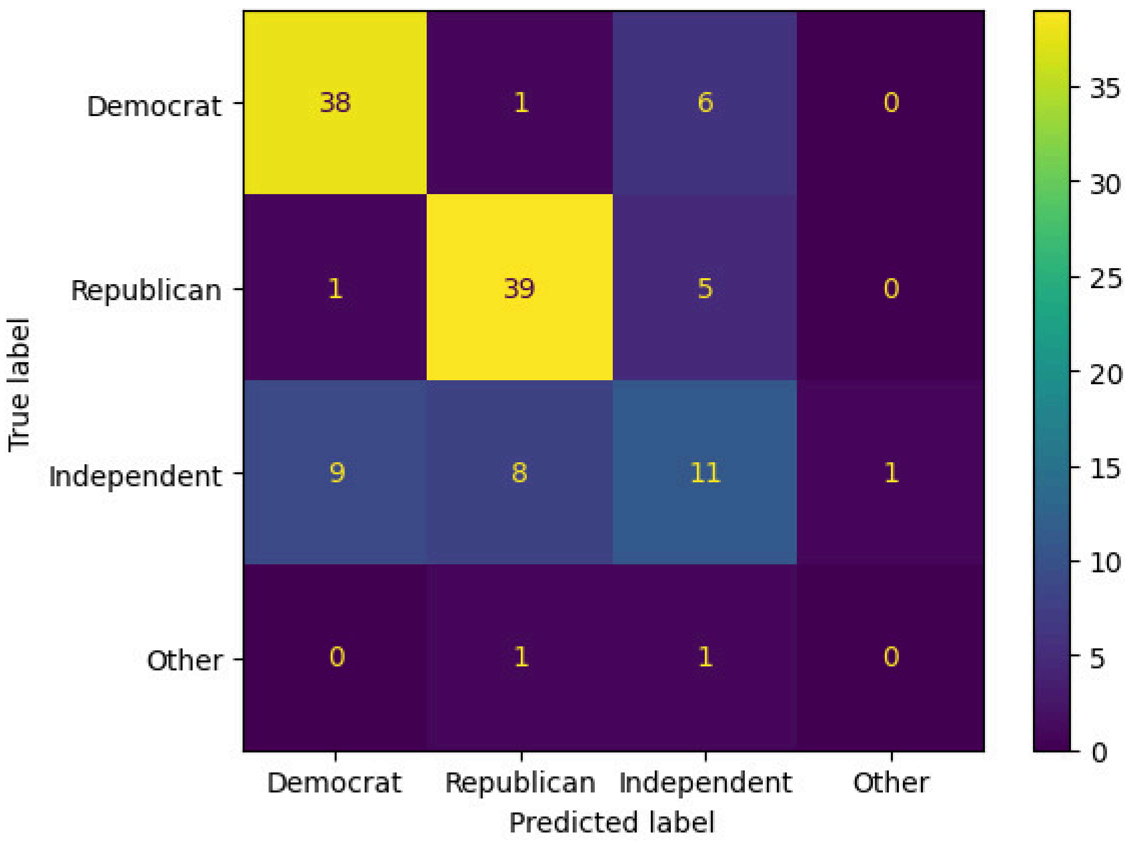


In [18]:
```
# Citation: (Scikit-learn Documentation. "Plot Confusion Matrix.", 11/2024)

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# Predict class probabilities
y_pred_prob = model.predict(X_test)

# Convert probabilities to class predictions
y_pred = np.argmax(y_pred_prob, axis=1)

# Convert one-hot encoded labels back to single class integers
y_true = np.argmax(y_test, axis=1)

# Confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot of the confusion matrix
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Democrat", "Rep
disp.plot(cmap='viridis')
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
```

Out[18]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1866f5dd4f0>
```



In [19]:
```
# All columns and unique values for each
for column in anes_selected_cleaned.columns:
    unique_values = anes_selected_cleaned[column].unique()
    print(f"Column: {column}")
    print(f"Unique values: {unique_values}")
    print("-" * 40)  # Separator for better readability
```

```
Column: Gender
Unique values: [2. 1.]
----------------------------------------
Column: Attention_to_Politics
Unique values: [4. 3. 2. 1. 5.]
----------------------------------------
Column: Attention_to_Political_Campaigns
Unique values: [3. 1. 2.]
----------------------------------------
Column: Party_of_Registration
Unique values: [4. 5. 1. 2.]
----------------------------------------
Column: Voted_in_Presidential_Primary
Unique values: [1. 2.]
----------------------------------------
Column: Intend_to_Vote_for_President
Unique values: [1.]
----------------------------------------
Column: Voting_for_President
Unique values: [ 3.  1.  2.  5. 11.  4.]
----------------------------------------
Column: Strength_of_Candidate_Preference
Unique values: [1. 2.]
----------------------------------------
Column: Intend_to_Vote_for_US_House
Unique values: [1.]
----------------------------------------
Column: Candidate_Intend_to_Vote_for_US_House
Unique values: [ 2.  1.  5.  7.  3. 11. 10.  6.  4.]
----------------------------------------
Column: Voted_for_President_in_2016
Unique values: [5. 1. 2.]
----------------------------------------
Column: Voted_for_President_in_2012
Unique values: [1. 2.]
----------------------------------------
Column: Ideology_Scale
Unique values: [4. 2. 7. 3. 6. 5. 1.]
----------------------------------------
Column: Placement_of_Democratic_Party
Unique values: [3. 4. 2. 1. 5. 7. 6.]
----------------------------------------
Column: Placement_of_Republican_Party
Unique values: [6. 7. 5. 1. 4. 2. 3.]
----------------------------------------
Column: Democratic_Candidate_Strong_Leadership
Unique values: [5. 2. 1. 4. 3.]
----------------------------------------
Column: Democratic_Candidate_Cares_for_People
Unique values: [5. 1. 2. 4. 3.]
----------------------------------------
Column: Republican_Candidate_Strong_Leadership
Unique values: [5. 1. 3. 2. 4.]
----------------------------------------
Column: Republican_Candidate_Cares_for_People
Unique values: [5. 1. 3. 2. 4.]
```

```
----------------------------------------
Column: Party_Identification
Unique values: [4. 1. 2. 3.]
----------------------------------------
Column: Government_Wastes_Tax_Money
Unique values: [1. 2. 3.]
----------------------------------------
Column: Party_Better_Handling_Economy
Unique values: [3. 2. 1. 5. 4.]
----------------------------------------
Column: Party_Better_Handling_Health_Care
Unique values: [3. 1. 5. 4. 2.]
----------------------------------------
Column: Party_Better_Handling_Immigration
Unique values: [3. 1. 2. 5. 4.]
----------------------------------------
Column: Party_Better_Handling_Taxes
Unique values: [3. 2. 1. 5. 4.]
----------------------------------------
Column: Party_Better_Handling_Environment
Unique values: [3. 2. 1. 5. 4.]
----------------------------------------
Column: Federal_Spending_Social_Security
Unique values: [3. 1. 2.]
----------------------------------------
Column: Federal_Spending_Public_Schools
Unique values: [1. 2. 3.]
----------------------------------------
Column: Federal_Spending_Border_Security
Unique values: [3. 2. 1.]
----------------------------------------
Column: Federal_Spending_Crime
Unique values: [3. 2. 1.]
----------------------------------------
Column: Federal_Spending_Welfare_Programs
Unique values: [2. 3. 1.]
----------------------------------------
Column: Federal_Spending_Aid_to_Poor
Unique values: [3. 1. 2.]
----------------------------------------
Column: Position_on_Abortion_Joe_Biden
Unique values: [4. 3. 2. 1.]
----------------------------------------
Column: Position_on_Abortion_Donald_Trump
Unique values: [2. 1. 4. 3.]
----------------------------------------
Column: Abortion_Rights_Supreme_Court
Unique values: [2. 1. 3.]
----------------------------------------
Column: Favor_Oppose_Death_Penalty
Unique values: [1. 2.]
----------------------------------------
Column: US_Position_in_World
Unique values: [1. 3. 2.]
----------------------------------------
Column: Country_Better_Staying_Out
```

```
Unique values: [1. 2.]
----------------------------------------
Column: Use_of_Military_Force
Unique values: [3. 4. 2. 5. 1.]
----------------------------------------
Column: Accuracy_of_Vote_Count
Unique values: [2. 5. 4. 3. 1.]
----------------------------------------
Column: Favor_Oppose_Voter_ID
Unique values: [1. 3. 2.]
----------------------------------------
Column: Trust_in_News_Media
Unique values: [1. 3. 4. 5. 2.]
----------------------------------------
Column: Income_Differences_Larger
Unique values: [1. 2.]
----------------------------------------
Column: Favor_Protect_LGBT_Job_Discrimination
Unique values: [1. 2.]
----------------------------------------
Column: Allow_Gay_Adoption
Unique values: [1. 2.]
----------------------------------------
Column: Position_on_Gay_Marriage
Unique values: [1. 2. 3.]
----------------------------------------
Column: Policy_Toward_Unauthorized_Immigrants
Unique values: [3. 2. 4. 1.]
----------------------------------------
Column: Favor_Oppose_Ending_Birthright_Citizenship
Unique values: [3. 1. 2.]
----------------------------------------
Column: Favor_Oppose_Border_Wall
Unique values: [3. 1. 2.]
----------------------------------------
Column: Respondent_Religion
Unique values: [12.  2. 11.  1.  5.  4.  3.  8.  7. 10.  6.]
----------------------------------------
Column: Religious_Service_Attendance
Unique values: [5. 2. 4. 1. 3.]
----------------------------------------
Column: Financial_Situation_Compared_to_Last_Year
Unique values: [5. 3. 4. 2. 1.]
----------------------------------------
Column: Financial_Situation_Next_Year
Unique values: [2. 4. 3. 1. 5.]
----------------------------------------
Column: Marital_Status
Unique values: [1. 4. 3. 6. 5.]
----------------------------------------
Column: Education_Level
Unique values: [ 3.  6.  8.  4.  7.  5.  2.  1. 95.]
----------------------------------------
Column: Active_Duty_in_Armed_Forces
Unique values: [3. 2. 1.]
----------------------------------------
```

```
Column: Employment_Status
Unique values: [1. 2. 4. 7. 3. 8. 9. 6.]
----------------------------------------
Column: Household_Union_Member
Unique values: [2. 1.]
----------------------------------------
Column: Hispanic_Origin
Unique values: [2. 1.]
----------------------------------------
Column: Parents_Native_Status
Unique values: [1. 3. 2.]
----------------------------------------
Column: Spouse_Hispanic_Origin
Unique values: [2. 1.]
----------------------------------------
Column: Spouse_Gender
Unique values: [1. 2.]
----------------------------------------
Column: Number_of_Children_in_Household
Unique values: [1. 0. 2. 4. 3.]
----------------------------------------
Column: State_or_Country_of_Upbringing
Unique values: [ 6. 17. 36. 42.  8. 40. 25. 34. 59. 21. 26. 12. 55. 31. 20. 24. 48.
49.
 18. 19. 27. 41. 37.  9. 45. 35. 39. 28. 54. 57. 16. 44. 32. 51.  5. 13.
 29.  4. 23. 22. 46. 53. 11. 38. 33. 10.  1.]
----------------------------------------
Column: Job_Loss_in_Household
Unique values: [1. 2.]
----------------------------------------
Column: Sexual_Orientation
Unique values: [1. 3. 2. 4.]
----------------------------------------
```

In [20]:
```python
import numpy as np
import pandas as pd

# New example with exactly 65 values (corresponding to the columns in X_train)
example_input = [
    1, 3, 2, 1, 1, 1, 3, 2, 1, 3, 1, 2, 4, 3, 6, 5, 2, 3, 1, 2, 1, 3, 2, 3, 1,
    2, 1, 3, 2, 1, 2, 3, 4, 1, 1, 3, 2, 3, 2, 1, 3, 2, 1, 2, 3, 4, 1, 2, 3, 3,
    1, 4, 3, 1, 4, 2, 1, 1, 2, 1, 2, 1, 12, 1, 4
]

# Converting the example_input to a DataFrame with the same columns as X_train
example_df = pd.DataFrame([example_input], columns=X_train.columns)

# Ensuring the input data type matches the training data
example_df = example_df.astype('float32')

# Checking the shape of the input to ensure it matches the model's expected input
print("Input shape:", example_df.shape)

# Predicting Party Identification
predictions = model.predict(example_df)
```

```
# Decoding the predictions
predicted_classes = predictions.argmax(axis=1) + 1 # classes start from 1
print("Predicted Party Identification Classes:", predicted_classes)
```

```
Input shape: (1, 65)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 187ms/step
Predicted Party Identification Classes: [2]
```

```
In [ ]:
```