PREDICTING 2024 PRESIDENTIAL ELECTIONS

through Detailed Analysis of Party, Contest, and State Combinations Using Neural Networks

CS 613-900,
Final Presentation
Professor Kim

Audrey Tin Latt

DREXEL UNIVERSITY
College of
Computing &
Informatics

SPRING 2024

THE PARADOX OF LEVEL TWO CHAOS:

ELECTION PREDICTIONS CREATE A FEEDBACK LOOP, ALTERING VOTER AND CANDIDATE BEHAVIOR

If a program predicts oil will be $100 tomorrow, traders will buy today, pushing the price to $100 immediately

If voters believe their candidate will lose, they may not vote.

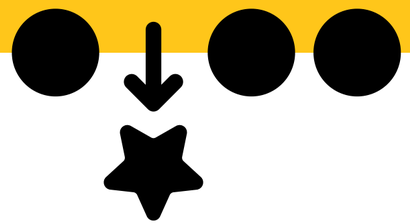
Understanding and managing level two chaos requires innovative approaches.

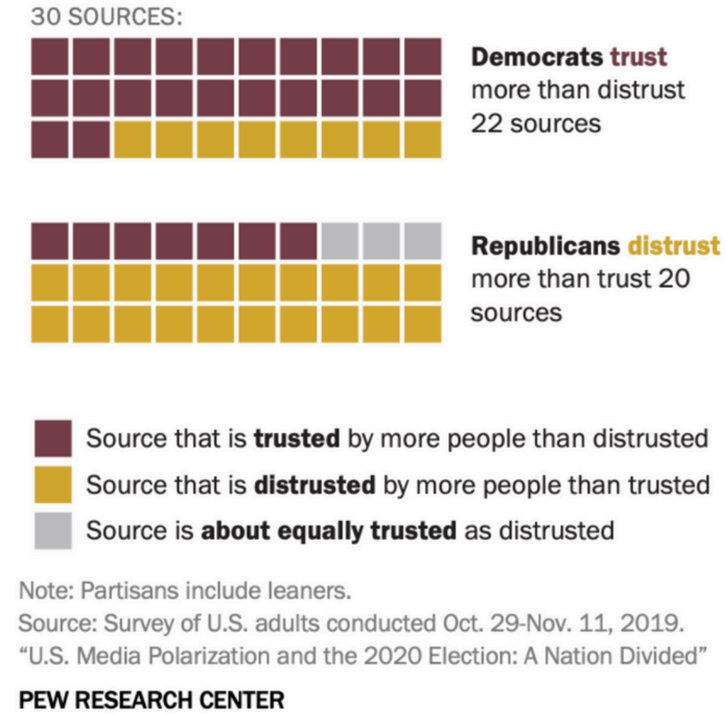
YUVAL NOAH HARRARI
HISTORIAN AND AUTHOR
OF 'SAPIENS', HOMO DEUS'

2020 elections

UNPRECEDENTED LEVEL OF POLARITY CULMINATED FROM SOCIAL MEDIA TRENDS, COVID, AND PARTY RHETORICS.

EXISTING PREDICTION MODELS FOCUS PRIMARILY ON ECONOMIC & SOCIAL MEDIA TRENDS





My ML Model aims to revisit the basics,
exploring micro-level variations and combinations of grassroots trends that might have been overlooked.

From house contests to other localized elections, leveraging Neural Network,
this paper aims to identify millions of potential combinations and patterns to improve the accuracy of election predictions.

Provides a nuanced understanding of voter behavior and election dynamics.

DATA SOURCES

HARVARD Dataverse

The Dataverse® Project

AK_tabular	6/12/2024 ...	TSV File	332 KB
AL_tabular	6/12/2024 ...	TSV File	263 KB
AR_tabular	6/12/2024 ...	TSV File	852 KB
AS_tabular	6/12/2024 ...	TSV File	10 KB
AZ_tabular	6/12/2024 ...	TSV File	393 KB

Election	Contest Selection	Party	ReportingUnit	VoteType	Count	Preliminary
2020 General	AS Governor	Gaoteote Palaie Tofau	Independent Party	American Samoa;(no district)	total	0	False
2020 General	AS Governor	Gaoteote Palaie Tofau	Independent Party	American Samoa;District of Manu'a	total	147	False
2020 General	AS Governor	Gaoteote Palaie Tofau	Independent Party	American Samoa;Eastern District	total	1322	False
2020 General	AS Governor	Gaoteote Palaie Tofau	Independent Party	American Samoa;Western District	total	1125	False
2020 General	AS Governor	I'aulualo Fa'afetai Talia	Independent Party	American Samoa;(no district)	total	5	False
2020 General	AS Governor	I'aulualo Fa'afetai Talia	Independent Party	American Samoa;District of Manu'a	total	279	False
2020 General	AS Governor	I'aulualo Fa'afetai Talia	Independent Party	American Samoa;Eastern District	total	559	False
2020 General	AS Governor	I'aulualo Fa'afetai Talia	Independent Party	American Samoa;Western District	total	618	False
2020 General	AS Governor	Lemanu Palepoi Mauga	Democratic Party	American Samoa;(no district)	total	3	False
2020 General	AS Governor	Lemanu Palepoi Mauga	Democratic Party	American Samoa;District of Manu'a	total	499	False
2020 General	AS Governor	Lemanu Palepoi Mauga	Democratic Party	American Samoa;Eastern District	total	3660	False
2020 General	AS Governor	Lemanu Palepoi Mauga	Democratic Party	American Samoa;Western District	total	2992	False
2020 General	AS Governor	Nua Sao	Independent Party	American Samoa;(no district)	total	0	False
2020 General	AS Governor	Nua Sao	Independent Party	American Samoa;District of Manu'a	total	123	False
2020 General	AS Governor	Nua Sao	Independent Party	American Samoa;Eastern District	total	261	False

HARVARD Dataverse

Add Data ▾ Search ▾ About User Guide Support S

Election Results

(Portland State University)

Harvard Dataverse > Stephanie Frank Singer > Election Results >

US 2020 General Official Election Results in Tabular Format

Version 1.0

Singer, Stephanie Frank, 2021, "US 2020 General Official Election Results in Tabular Format", https://doi.org/10.7910/DVN/RV80FW, Harvard Dataverse, V1, UNF:6:Cg1tQvodspHg6eXLQBKHIQ== [fileUNF]

Cite Dataset ▾ Learn about Data Citation Standards.

Access Data

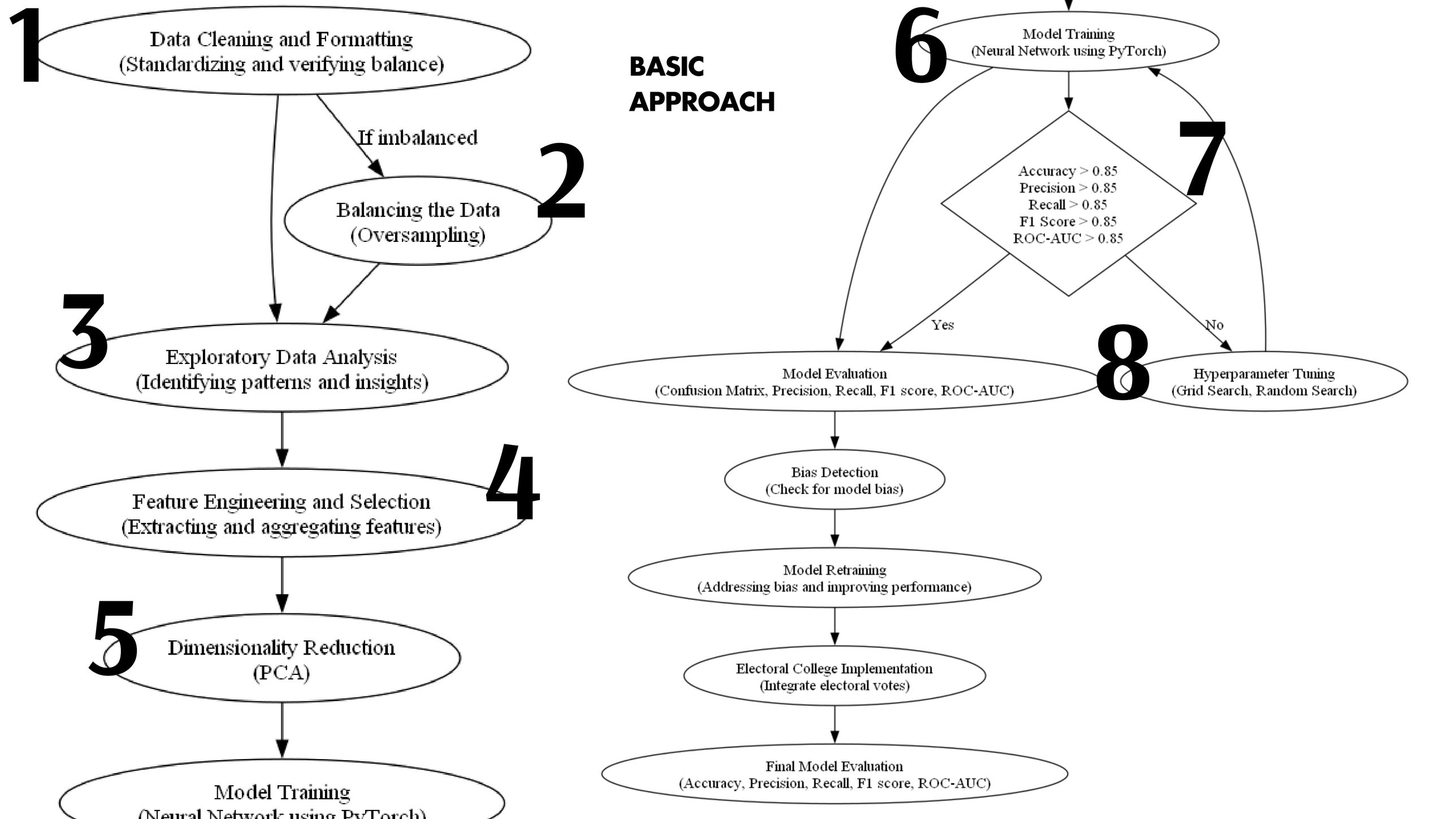
Contact Owner

Dataset Metrics

5,371 Downloads

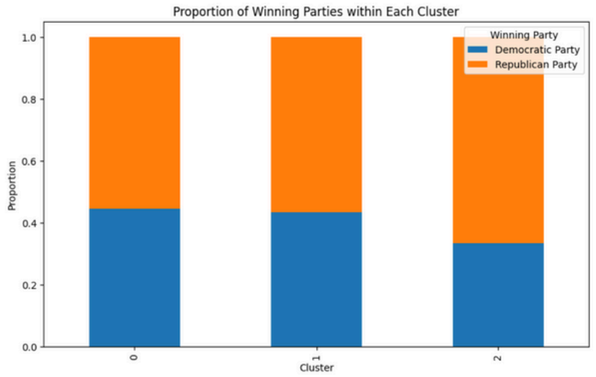
urisdictions -- 50 states, the District of
er major subdivision) including all federal
. The files are in tab-separated format. The
level from, files or web pages published by
vare used to consolidate the results and
e raw files are available here. (2021-08-24)

artisans = Election Verification. MAA Focus


BASIC APPROACH
1
Data Cleaning and Formatting
(Standardizing and verifying balance)
If imbalanced
2
Balancing the Data
(Oversampling)
3
Exploratory Data Analysis
(Identifying patterns and insights)
4
Feature Engineering and Selection
(Extracting and aggregating features)
5
Dimensionality Reduction
(PCA)
Model Training
(Neural Network using PyTorch)
6
Model Training
(Neural Network using PyTorch)
7
Accuracy > 0.85
Precision > 0.85
Recall > 0.85
F1 Score > 0.85
ROC-AUC > 0.85
Yes
No
8
Model Evaluation
(Confusion Matrix, Precision, Recall, F1 score, ROC-AUC)
Hyperparameter Tuning
(Grid Search, Random Search)
Bias Detection
(Check for model bias)
Model Retraining
(Addressing bias and improving performance)
Electoral College Implementation
(Integrate electoral votes)
Final Model Evaluation
(Accuracy, Precision, Recall, F1 score, ROC-AUC)

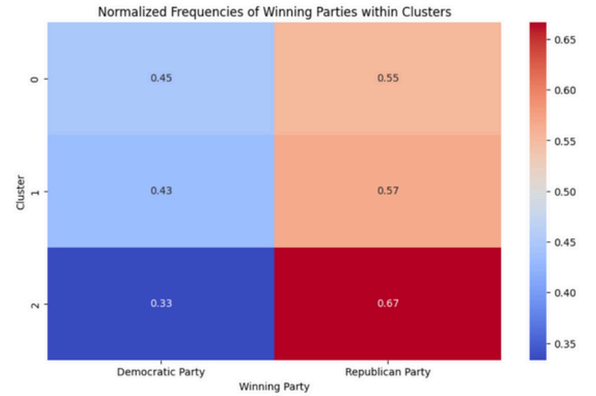
DATA CLEANING & VISUALIZATION

PRE-TRAINING / PRE-TESTING DATA




```
Correlation matrix between clusters and winning parties:
winning_party     Democratic Party  Republican Party
winning_party
Democratic Party          1.000000          0.999055
Republican Party          0.999055          1.000000
```

```
Normalized frequencies of winning parties within each cluster:
winning_party  Democratic Party  Republican Party
cluster
0                     0.445783          0.554217
1                     0.434066          0.565934
2                     0.333333          0.666667
```



convert categorical variables (such as party, contest type, state, and vote type) into numerical codes. This is necessary because machine learning algorithms work with numerical data.

[13]:
```
merged_data['party_encoded'] = merged_data['party'].astype('category').cat.codes
merged_data['contest_type_encoded'] = merged_data['contest_type'].astype('category').cat.codes
merged_data['state_encoded'] = merged_data['state'].astype('category').cat.codes
merged_data['votetype_encoded'] = merged_data['votetype'].astype('category').cat.codes
```

defined the features (input variables) and the target variable (what I want to predict), features include encoded contest type, state, party, vote type, and the number of votes in House and Senate contests, as well as total presidential votes. The target variable is the encoded winning party for the presidential election.

[14]:
```
features = ['contest_type_encoded', 'state_encoded', 'party_encoded', 'votetype_encoded', 'house_votes', 'senate_votes', 'total_presidential_votes']
X = merged_data[features]
y = merged_data['presidential_winner'].astype('category').cat.codes
```

standardized features

[15]:
```
X = (X - X.mean()) / X.std()
```

Split the data into training and testing sets

[16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

verified sets

[17]:
```
print("Training set size:", X_train.shape[0])
print("Testing set size:", X_test.shape[0])
```

```
Training set size: 516
Testing set size: 222
```

BALANCING DATA

BY OVERSAMPLING MINORITY CLASS

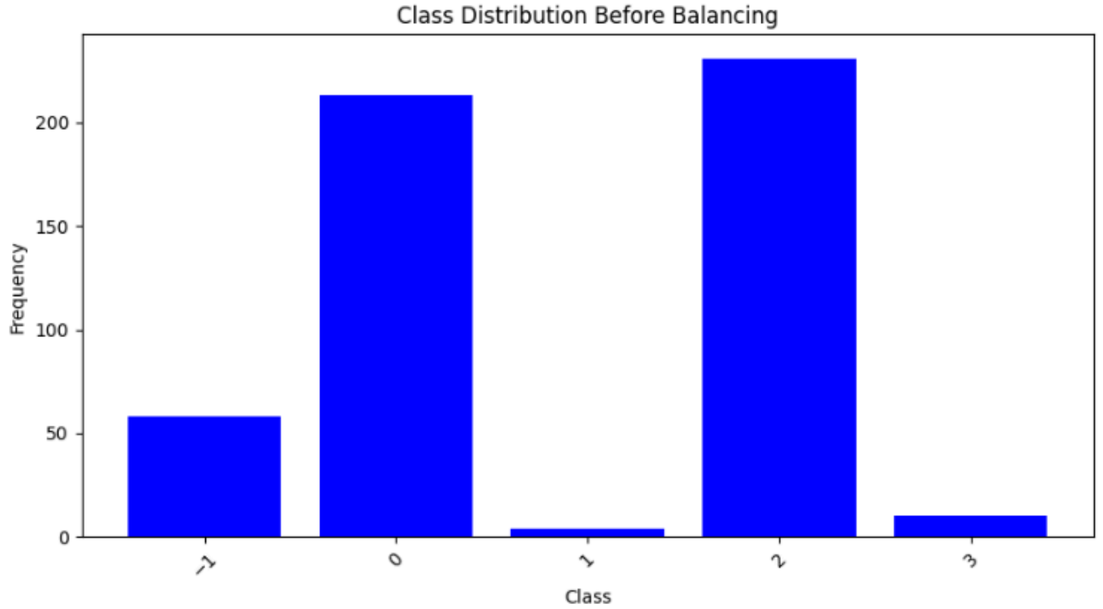
- Calculated the maximum class count.
- Identified indices of each class.
- Randomly oversampled indices to balance the classes.
- Combined oversampled data to create a balanced dataset.

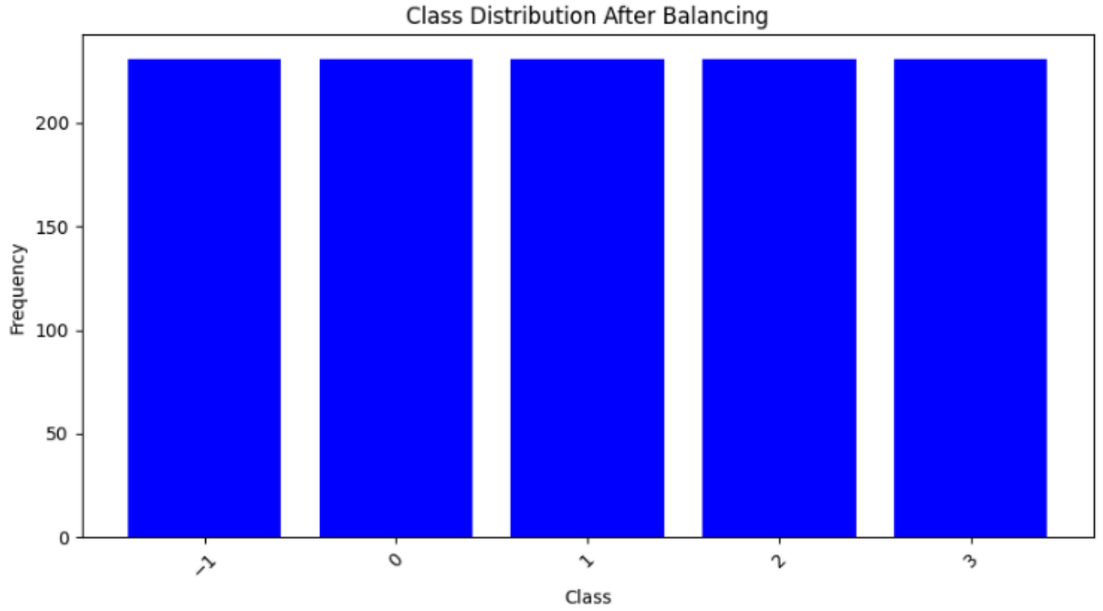
Prevent Bias

Improve Accuracy

Model Fairness

Fair Class Representation





I noticed an issue where some classes were labeled as -1. This means there is some invalid or undefined class labels, which could really impact the performance and accuracy of the machine learning model. The -1 labels were not representative of any valid class and could lead to misleading results and biased predictions. So, I wrote additional lines below to identify and remove samples with -1 labels from both the training and testing datasets

LITERATURE REVIEW

REPORT | SEPTEMBER 30, 2021

SHARE

Charting Congress on Social Media in the 2016 and 2020 Elections

The 2020 election featured dramatic increases in lawmaker posts and audience engagement, but less overlap in the sources shared by members of each party

BY SONO SHAH

Aspect	Description
Existing focus	Broad economic indicators, social media trends
Identified gap	Lack of attention to micro-level variations
Research goal	Focus on grassroots precinct-level trends per state
Data	General election contests, house district, senate races
Aim	Understand localized election impacts on overall results
Method	Leverage neural networks
Objective	Identify intricate patterns, enhance election prediction accuracy

was the case in 2016.

PREDICTING PRESIDE

ELECTIONS AND OTH

SECOND EDITION

A Hybrid Method of Sentiment Analysis and Machine Learning Algorithm for the U.S. Presidential Election Forecasting

g 2020 election vs. 2016:
agement, but fewer links
n parties

MODEL SELECTION + EVALUATION METRICS

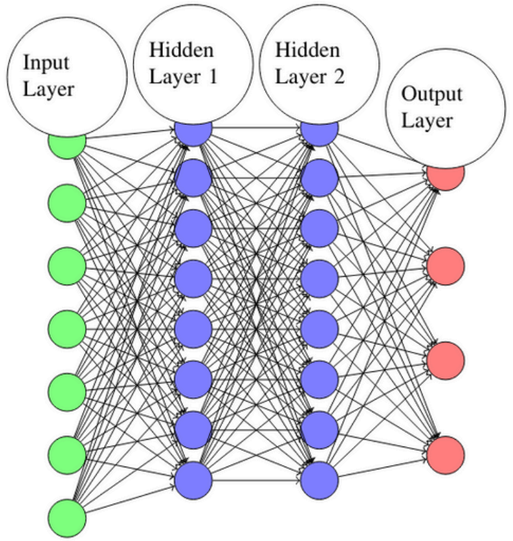


Fig. 5: Neural Network Architecture

TABLE III: Training Components

Component	Description
Loss Function	Cross-Entropy Loss for multi-class classification
Optimizer	Stochastic Gradient Descent (SGD) for weight updates
Training Loop	Iterative forward and backward passes over epochs

A neural network model was built using PyTorch. The model was trained with standardized and balanced data, aiming to optimize accuracy and generalization. The training process involved multiple epochs and careful tuning of hyperparame-

TABLE IV: Data Preparation and DataLoader Setup

Step	Description
Convert Data	Cleaned datasets to PyTorch tensors
Create Datasets	Combine features, labels to TensorDataset objects
Setup DataLoaders	mini-batches for training and testing

TABLE V: Neural Network Architecture

Layer	Description
Input Layer	Fully connected layer transforming input features
Hidden Layer 1	Fully connected layer with ReLU activation
Hidden Layer 2	Fully connected layer with ReLU activation
Output Layer	Fully connected layer producing class scores

C. Loss Function

The Cross-Entropy Loss for multi-class classification is computed as:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

In PyTorch, this is implemented using:

```
criterion = nn.CrossEntropyLoss()
```

where y_i are the true labels and $\hat{y}_i$ are the predicted probabilities.

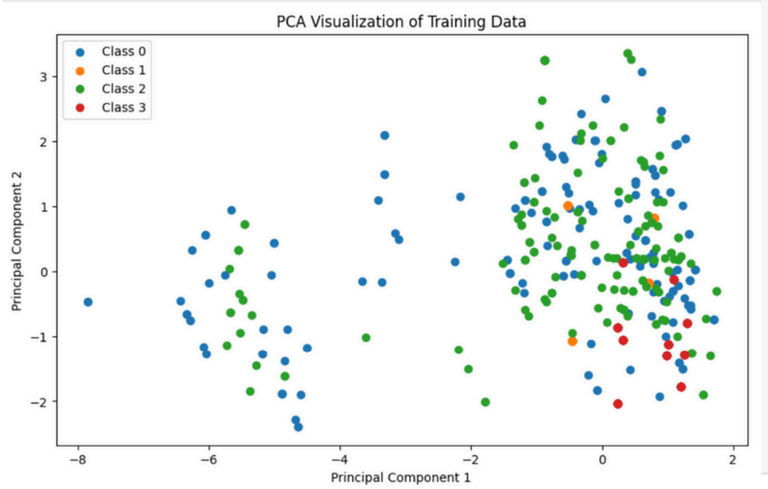
D. Backpropagation

Backpropagation computes the gradient of the loss function with respect to each weight using the chain rule and updates the weights in the opposite direction of the gradient. In PyTorch, this is handled by:

```
loss.backward()
```

I converted the cleaned training and testing data into PyTorch tensors and created DataLoader objects for both datasets to facilitate batch processing during model training and evaluation. The DataLoader for the training set was set to shuffle the data, while the test set DataLoader was not shuffled.

```
]:
import torch
from torch.utils.data import DataLoader, TensorDataset

X_train_tensor = torch.tensor(X_train_cleaned, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test_cleaned, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train_cleaned, dtype=torch.long)
y_test_tensor = torch.tensor(y_test_cleaned, dtype=torch.long)

_train_tensor, y_train_tensor)
test_tensor, y_test_tensor)

_dataset, batch_size=32, shuffle=True)
ataset, batch_size=32, shuffle=False)

r:
ape)

[32])
```



It was not that good at seperating so I went straight ahead to NN

neural network using PyTorch to classify the election data. The network consists of three fully connected nctions. The model was trained using SGD optimizer with a learning rate of 0.01 and a momentum of 0.9, and ossEntropyLoss. The training loop ran for 50 epochs, during which the model's parameters were updated rom the predictions and actual labels. After training, the model's accuracy was evaluated on the test set, the model's predictions.

```
_init__()
(X_train_cleaned.shape[1], 128)
(128, 64)
(64, len(np.unique(y_train_cleaned)))  # Number of classes

er1(x))
er2(x))
```

RELATION TO CLASS TOPICS

C. Loss Function

The Cross-Entropy Loss for multi-class classification is computed as:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

In PyTorch, this is implemented using:

```
criterion = nn.CrossEntropyLoss()
```

where y_i are the true labels and $\hat{y}_i$ are the predicted probabilities.

D. Backpropagation

Backpropagation computes the gradient of the loss function with respect to each weight using the chain rule and updates the weights in the opposite direction of the gradient. In PyTorch, this is handled by:

```
loss.backward()
```

E. Stochastic Gradient Descent (SGD)

SGD updates the weights as:

$$W = W - \eta \frac{\partial L}{\partial W}$$

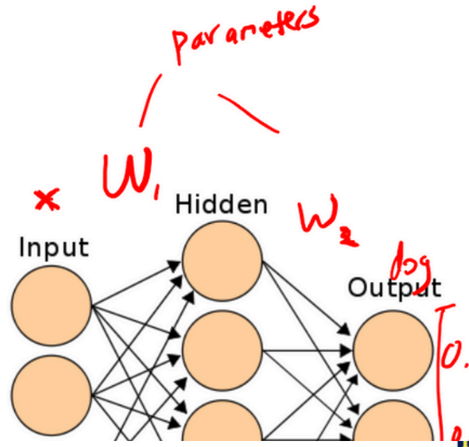
where η is the learning rate and $\frac{\partial L}{\partial W}$ is the gradient of the loss with respect to the weights. In PyTorch, this is implemented using:

```
optimizer = optim.SGD(..)
optimizer.step()
```

F. Handling Invalid Labels

Training an ANN

- Where do these weights come from?
- We need to *learn* them of course!
- And now there's a lot of weights to learn:
  - Each hidden layer node weights the output of the input layer.
  - Each output layer node weights the output of the hidden layer node.
- Our goal is to learn all of to minimize the training likelihood)
- To do this, we use what's *backward propagation.*



6/12/24

Multi-Class Evaluation

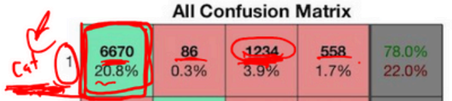
- Just like binary classification, we can evaluate the accuracy of a multi-class classifier:

$$accuracy = \frac{1}{N}\sum_{i=1}^{N}(Y_i = \hat{Y}_i)$$

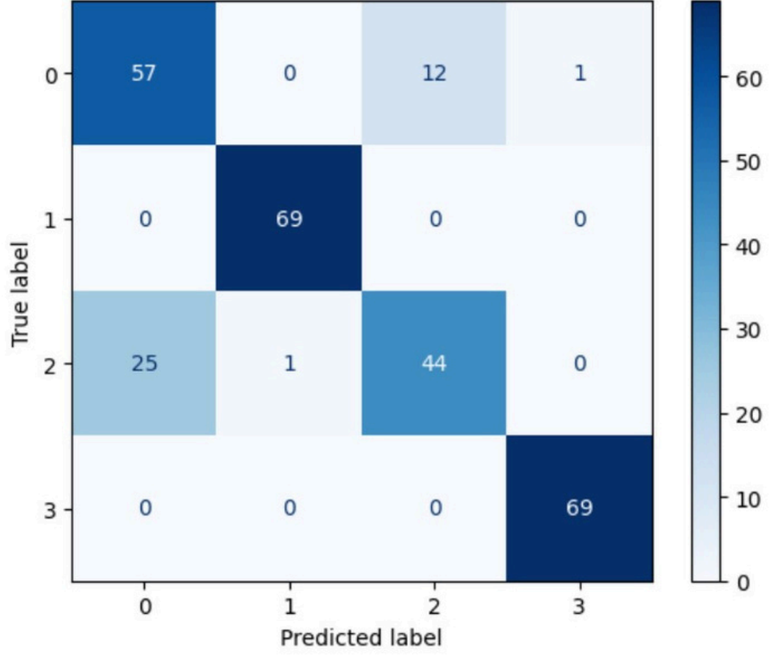
- In addition, particular to multi-class classification, we may be interested in investigating which classes get confused with which other classes
- To observe this, we can look at a *confusion matrix*

4/26/24 Edward Kim, Drexel University

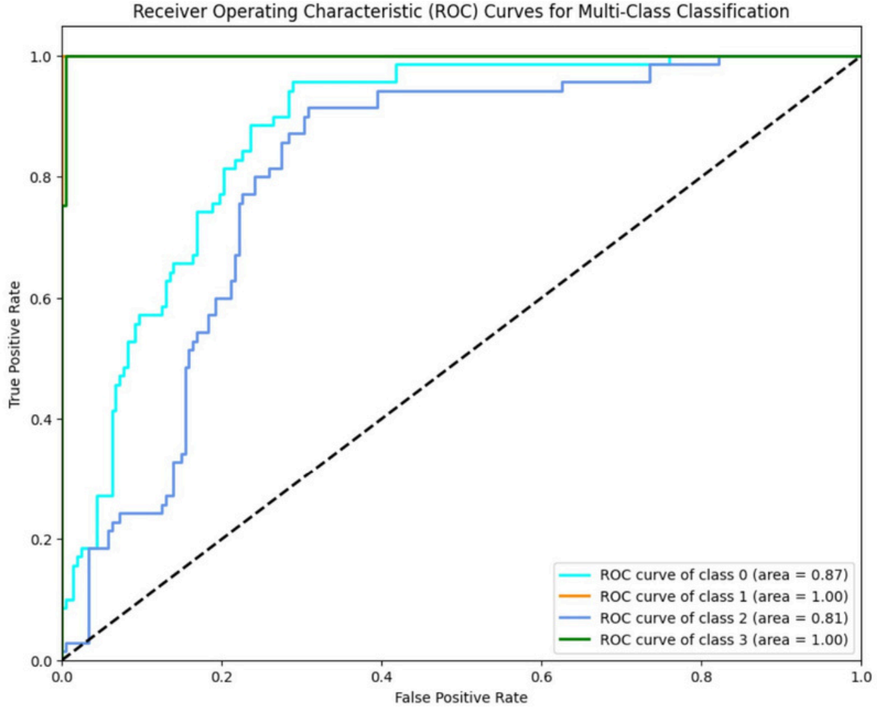
Confusion Matrix



RESULTS AND DISCUSSION




```
Epoch 38/50, Loss: 0.2344554393064408
Epoch 39/50, Loss: 0.22328700444528035
Epoch 40/50, Loss: 0.18254523600141206
Epoch 41/50, Loss: 0.17685370431059882
Epoch 42/50, Loss: 0.17913885237205596
Epoch 43/50, Loss: 0.1851764523557254
Epoch 44/50, Loss: 0.1684659073750178
Epoch 45/50, Loss: 0.1634257925408227
Epoch 46/50, Loss: 0.1821093513142495
Epoch 47/50, Loss: 0.1769843807532674
Epoch 48/50, Loss: 0.16087626523914791
Epoch 49/50, Loss: 0.1362867449365911
Epoch 50/50, Loss: 0.14275680553345454
Test Accuracy: 88.85%
```



	precision	recall	f1-score	support
0	0.70	0.81	0.75	70
1	0.99	1.00	0.99	69
2	0.79	0.63	0.70	70
3	0.99	1.00	0.99	69
accuracy			0.86	278
macro avg	0.86	0.86	0.86	278
weighted avg	0.86	0.86	0.86	278

- The neural network was trained using a robust pipeline, with cross-entropy loss and stochastic gradient descent with momentum for efficient learning.
- The training process involved multiple epochs and batch processing to ensure thorough learning and convergence.
- The model was evaluated and fine-tuned iteratively to optimize performance.
- Future efforts will focus on expanding the feature set to include real-time social media trends, utilizing APIs from platforms like Twitter (now X).
- This will enable the model to capture dynamic voter sentiments and social media influence, providing a more holistic view of the electoral landscape.
- Additionally, further optimization techniques and model variants will be explored to enhance predictive performance and fairness.

CONCLUSION

slide 2, Yuval's photo https://www.gq.com/story/yuval-noah-harari-tech-future-survival
slide 3 infographic
https://www.pewresearch.org/journalism/2020/01/24/u-s-media-polarization-and-the-2020-election-a-nation-divided/pj_2020-01-24_media-polarization_0-01-png/
slide 4 dataverse logos https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/RV80FW dataverse website
slide 6 literary articles/books shown
https://ar5iv.labs.arxiv.org/html/2312.05584https://fairmodel.econ.yale.edu/rayfair/pdf/2011b.pdfhttps://www.pewresearch.org/politics/2021/09/30/charting-congress-on-social-media-in-the-2016-and-2020-elections/
slide 10 screenshots of slides professor Kim's lecture slides from Blackboard

SLIDE IMG REFERENCE LINKS

[1] S. F. Singer, "US 2020 General Official Election Results in Tabular Format," Version 1.0, 2021. Available: https://doi.org/10.7910/DVN/RV80FW. Harvard Dataverse, V1, UNF:6:Cg1tQvodspHg6eXLQBKHlQ==.
[2] A. Burkov, The Hundred-Page Machine Learning Book, 1st ed. ISBN-10: 199957950X, ISBN-13: 978-1999579500.
[3] A. Burkov, Machine Learning Engineering. ISBN-10: 1999579577, ISBN-13: 978-1999579579.
[4] A. Muller and S. Guido, Introduction to Machine Learning with Python. ISBN-10: 1449369413, ISBN-13: 978-1449369415.
[5] G. James, D. Witten, An Introduction to Statistical Learning. ISBN-10: 1461471370, ISBN-13: 978-1461471370.
[6] E. Stevens, L. Antiga, Deep Learning with PyTorch. ISBN-13: 978-1617297120.
[7] G. Feng, K. Chen, H. Cai, Z. Li, "A Hybrid Method of Sentiment Analysis and Machine Learning Algorithm for the U.S. Presidential Election Forecasting," Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College, Zhuhai, China.
[8] R. C. Fair, Predicting Presidential Elections and Other Things, 2nd ed. Stanford, CA: Stanford University Press, 2012. ISBN-13: 978-0-8047-6049-2.
[9] National Archives and Records Administration, "Distribution of Electoral Votes," Available: https://www.archives.gov/electoral-college/allocation.

REFERENCES