Predicting Voting Propensity and Political Party Affiliation in Primary Elections

Rodrigo Pimentel[a,b] and Zachary deWardener[c]

[a]Department of Psychology, University of Rhode Island; [b]Department of Computer Science and Statistics, University of Rhode Island; [c]Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island

Project Report
CSC/DSP 461 Machine Learning (Fall 2022)
Department of Computer Science and Statistics
University of Rhode Island
Dr. Marco Alvarez

Author contributions: R.P. formulated the original hypothesis. R.P. and Z.D. designed and performed the research. R.P designed and performed experiment 1 and Z.D. designed and performed experiment 2. R.P and Z.D. wrote the paper.

1 Introduction

1.1 Background and Motivations

Billions of dollars are spent by campaigns to promote candidates in each election cycle. Rusch et. al further elaborate on the billions spent in campaigning: "Arguably, nowhere else is political campaigning a bigger business then in the US and nowhere else is more money being spent on convincing people to cast their ballots." [1] The majority of these funds are used for marketing purposes in a strategy commonly referred to as "Get Out The Vote" (GOTV) which is aimed at increasing voter registration, voter turnout, and swaying unaffiliated voters to one side or the other. Some of these strategies include enlisting volunteers to go door-to-door, sending targeted emails to certain groups of potential young voters, and deploying online marketing campaigns. The main problem that many researchers and political strategists have been tasked with solving is figuring out who campaigns should specifically target in order to optimize their spendings and increase the turnout of their supporters. The goal of this research is to train machine learning models to target potential voters, giving the campaigns the ability to target voters with messaging tailored to their propensity to vote - only by utilizing publicly available historical data.

This study herein focuses on utilizing voting history due to its predictive nature. Prior political science research has shown that voting in a prior election increases the probability of voting in a subsequent election by 13%. [2] Furthermore, the practicality of obtaining such data is also a consideration. Candidates for local offices may not have the capital required to purchase proprietary data, nor access proprietary voter mobilization software such as NGP VAN. The voter file is available from several states with costs ranging from gratis to several thousand dollars, although a substantial number offer the data for less than $100 [9]. For instance, in Rhode Island, one can obtain voter data for $25 by mailing an order form requesting a "List of Registered Voters", and are able to narrow down their request to data from a city or town, state representative district, or the entire statewide voter file [10].

Presently, when faced with a primary election, campaigns for local and state office may focus their voter outreach efforts (canvassing, phonebanking, and mailers) to those who meet requirements such as having voted in the last 3 out of 3 elections, 2 out of 3, or other similar criteria. For small campaigns in a large district, however, it may prove unfeasible to reach every single voter who meets such criteria, as the campaign would perhaps be better served by a narrower universe of voters. In other cases, a

campaign may also want to target a broader voting bloc beyond those who meet the targeting criteria. We hypothesize that by training a machine learning model that predicts the voting propensity of each individual voter, the model will achieve a better Matthews correlation coefficient than the traditional method currently used by campaigns (i.e. voted in the last three elections). Benefits of such a model include allowing campaigns to classify voters based on their likelihood to vote, allowing them to send targeted messages based on such, and scale their universe of voters to their campaign budget.

Applications for such a model that relies solely on the voter file and voting history may extend beyond stakeholders such as political campaigns and candidates themselves. Public opinion researchers, for instance, often ask voters how likely they think it is that they will vote. The issue with this, however, is that research has shown that vote self-prediction has poor predictive power, while voting is better predicted by past voting history [7]. Incorporating machine learning to provide individual probabilities for each voter may allow pollsters to formulate more accurate assessments as to who will show up to the polls on election day.

1.2 Related Work

There have been various attempts by researchers to tackle the problem of predicting voting propensity in upcoming elections. For instance, Tynan Challenor's work [5] follows a similar approach that we will be taking in terms of utilizing probabilistic supervised learning predictor models (such as Logistic Regression, SVM, and Naive Bayes) to determine voter propensity. However, Challenor's data structure differs from our selection as the predictions are based on data from a 2016 U.S. Census Bureau survey focusing on labor force demographics rather than directly predicting based on voter history for a specific town. Another instance of related research can be found in Pollard et al.'s preprint that utilized machine learning to predict individuals' voting behavior with the IPUMS-ASA U.S. Voting Behaviors dataset from the American Statistical Society. By utilizing data from 2004 through 2018, they achieved a Matthews correlation coefficient of 0.39 [4]. Other prior research has utilized Naive Bayes, gradient boosting machines, and super vector machines in predicting voter behavior [3][6][8].

2 Problem Definition

This project aims to provide a supervised machine learning model that can take historical voter data as an input and output the probability that an individual will vote

in the upcoming election. Along with this, the model should be able to predict the most probable party designation of unaffiliated voters prior to polling. These metrics will, in theory, help campaign managers to more efficiently direct their efforts towards likely voters.

3 Data

The dataset used is publicly available statewide voter registration data for the State of Rhode Island from the Rhode Island Secretary of State over the course of eight elections from 2018 to 2022 consisting of statewide primary/general, presidential preference primary, statewide referenda, and presidential elections. Along with specific elections, voter ID, party affiliation, year of birth, and locational information is also provided. A total of 816,279 registered voters are in the Statewide dataset.

The original voter file was split across two files. One contained voter information, and the other file contained vote history. After merging both files, the data contained 68 unique columns. After scrubbing, cleaning the data, we selected the following *fundamental features*. After which, experimental features representing criteria such as demographic identifiers will be added for further analysis. The *fundamental features* are as follows:

- **CITY:** Integer values pertaining to the general location of the voter.
- **ZIP CODE:** Four digit integer values representing the voter's zip code within their.
- **CURRENT PARTY:** Is representative of the three distinct party affiliation options given at registration; Unaffiliated (U), Democrat (D), and Republican (R) which are depicted as real integer values {1, 2, and 3} respectively.
- **YEAR OF BIRTH:** Year of birth of the voter.
- **ELECTION #:** Providing a binary value {1,0} representing the voter's history of voting in a given election {0 = did not vote, 1=did vote}. The "#" is a real integer value ranging from 3-8.
- **PARTY #:** Providing the party value for which a given voter affiliates with for a given statewide primary election.
- **TGT STATEWIDE PRIMARY:** represents one of the two target features that will be predicted. This feature specifically denotes whether the person of interest will be voting in the following primary election. This value is denoted by binary values {0,1}

- **TGT PARTY AFFILIATION:** is the second of the two target features, predicting which party the voter will affiliate with for the target election. This feature is in the same format as the CurrentParty feature listed above.

The *experimental features* described in the prior excerpt are listed with a level of uncertainty as whether a correlation exists between voter propensity or resultant party affiliation is to be determined. These features are listed below:

- **TRAVEL2POLL:** describes the distance in time that voter's must travel to their nearest polling location. Obtained by querying a third party API to first geocode the address to a latitude and longitude, and then using a third party navigation/maps API to obtain driving time.
- **ELECTION CONTROVERSY:** This feature would be obtained by several surveys or digital trace data as performed in [8] to measure controversy levels surrounding select candidates and their policies.
- **DEMOGRAPHIC FEATURES:** These features would be obtained from public Census Bureau surveys as was performed by Challenor in [5]. These features would contain information such as household income, employment status, and other information not provided in our current dataset.

4 Methods

4.1 Scaling methods

The basic reasoning behind the use of preprocessing techniques is to adjust the dataset's features so that they arrive at the machine learning (ML) algorithm in a more digestible form. In order to help our model converge at a solution more efficiently, *excluding logistic regression as it is not necessary to scale data for this model type*, we utilize a commonly used industry scaling method provided by Scikit Learn, MinMaxScaler(). The implementation of this method results in transformed column values proportionally within the range of 0 and 1. This technique is crucial for our neural network model as these types of ML algorithms require data to be on a 0-1 scale to avoid flat regions within the domain, which could result in a minimal learning ability [12]. This scaling method was only introduced to the neural networks dealing with voter propensity to evaluate the impact of scaling on an imbalanced dataset. The datasets used for both NN models are adjusted to be on a {0,1} scale prior to training so that binary cross entropy with sigmoid activation can be used for the output layers.

Another preprocessing technique utilized in this work is the resampling technique of RandomUnderSampling. This process is commonly used for imbalanced datasets, which falls in line with our voter files as about 16-20% of people voted in the target primary elections and 11-15% of voters are Republican (class = 1). The RandomUnderSampling function randomly selects and deletes or merges examples that fall in the majority class distribution (0 - did not vote in the case of our propensity models, and 1 - Republican for party affiliation models).

4.2 Machine Learning Models

Experiments 1 and 2 primarily focused on the following machine learning models: logistic regression and artificial neural networks. As such, we describe these models in detail in the following section.

4.2.1 Logistic Regression

Logistic regression is a simple, binary, linear classification method that inputs the product of training data (**x**) and a weight vector (**w**), processes the sum of resultant z values ($z=\mathbf{w}^T\mathbf{x}$) which are then sent through an activation function (*since there are only two classes, the logistic function is a sigmoid function*) and outputs a linear decision boundary that can be used to probabilistically classify targets when given new test data. A visualization of the logistic regression process can be seen below in **Figure 1**.

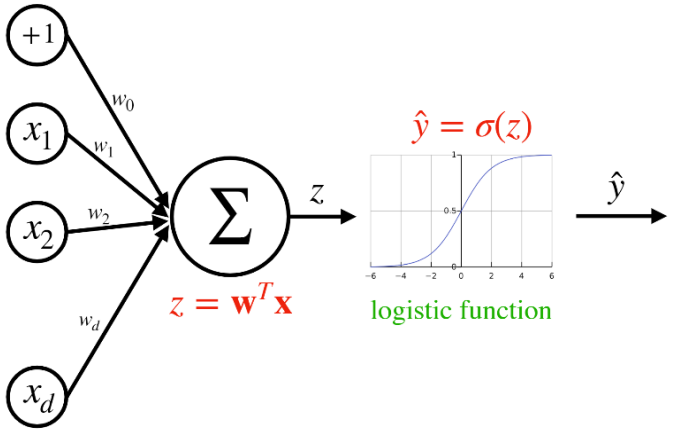


Figure 1: logistic regression visualization as a single "neuron" [11]

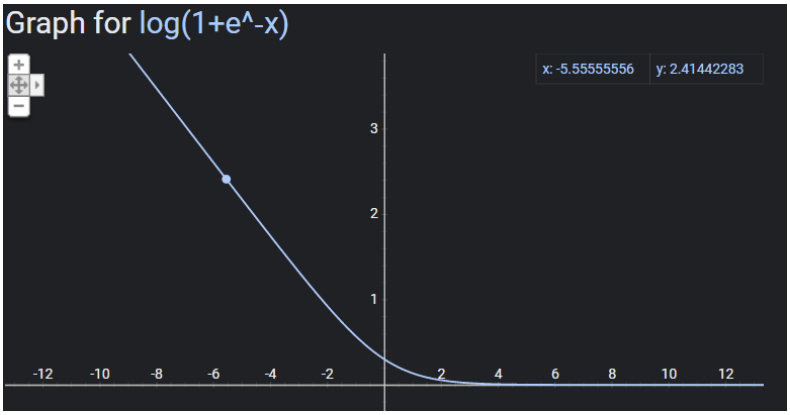
In order to solve logistic regression models, a crucial first step is to produce the weight vector that maximizes P(y | **x** ; w). The formula for **w*** can be derived through algebraic and logarithmic manipulations of the Maximum Likelihood Estimation (MLE) formula,

resulting in the negative log likelihood as seen in **Figure 2**. From this formulation we can define a loss function for showing the error in predicted versus actual target values within the supervised learning space. The loss function for logistic regression is referred to as the cross-entropy loss which is a common performance metric for classification models whose output is probability based. The basic idea of how this loss function works can be understood by looking at the graph of the formula $log(1 + e^{-x})$, shown in **Figure 3**, which will provide a function that in essence, shows the loss (vertical y-axis) increasing exponentially as the predicted probability (horizontal x-axis) decreases.

$$\text{negative log likelihood} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}}\right)$$

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}}\right)$$

Figure 2: Negative log likelihood formula & accompanying loss function respectively [11]



Source: Google Search Engine

Figure 3: Graph representing how the loss function penalizes inaccurate predictions

The next step towards completing this model is to take the partial derivative of the loss function so that gradient descent can be applied to find the best-fit decision boundary for the test and training data. In simple terms, the gradient descent update formula for logistic regression is as follows:

$$w^{\text{current} + 1} = w^{\text{current}} - (\text{learning rate})*(\text{gradient of the loss function})$$

In terms of probabilistic interpretation, we classify the propensity of voters and party affiliation of voters alike with the following example formulas [11]:

Probability of a registered voter partaking in the next election:

Let $y \in \{+1,-1\}$;

$$P(y = +1 \mid x) = \frac{1}{1+e^{-w^T x}} = \frac{e^{w^T x}}{e^{w^T x}+1} = \sigma(w^T x)$$

*Probability of a registered voter **not** partaking in the next election:*

$P(y = -1 \mid x) = 1 - P(y = +1 \mid x)$

-or-

$$P(y = -1 \mid x) = 1 - \frac{1}{1+e^{-w^T x}} = \frac{e^{-w^T x}}{e^{-w^T x}+1} = \sigma(-w^T x)$$

4.2.2 Artificial Neural Networks (ANN)

Neural networks, otherwise known as Muli-Layer Perceptron (MLP) models, have become increasingly common in day to day life. An example most highly related to our project, is the realm of directed marketing and sales applications. These algorithms enable platforms and campaigns to identify their target audiences' search histories, previous purchases, and other browsing patterns to promote relevant products and services that are most-likely to generate consumer interest [13]. The images depicted in **Figure 4** can be used to explain the basis of how MLPs work. *Section A* represents a single perceptron model similar to that which is used for logistic regression, seen more in depth in **Figure 1**, with input data being transformed into a single output ranging from 0 to 1 via transference through an activation function (*sigmoid activation is used in our case - allowing us to discern voters propensity and the swing of unaffiliated voters to Democrat or Republican)* seen more closely in the top-left portion of C. *Section B* on the other hand, represents an artificial neural network, connecting multiple perceptron layers to each other such that the output of one unit leads to the input of another with varying activation functions. The idea behind ANNs is that as the inputs journey through the "hidden layers" of the network of nodes, they are slightly modified by each step until reaching the output layer where the final representation of the input is predictive of the specified outcome[14].

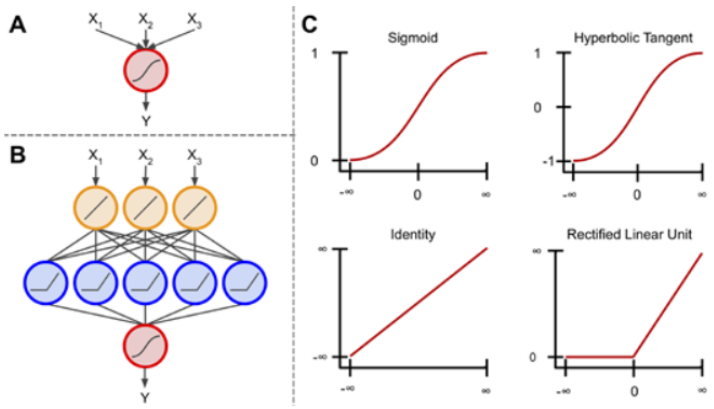


Figure 4: Neural network basics visualization [14]

4.2.3 Analysis

To perform model evaluation and selection, we will utilize k-fold cross validation with stratification. We first focused on the Matthews Correlation Coefficient and Area Under the ROC Curve (AUC-ROC) as our measure of quality for the classifications. However, after further research into the work presented in Saito et al. (2015) we have moved to primarily focusing on Area Under the Precision-Recall Curve (AUC-PRC) with Average Precision (AP) determining the model's performance with scikit-learn's PrecisionRecallDisplay() function. This is primarily due to the imbalanced nature of our data. According to the research performed by Saito et al. ROC is shown to perform poorly on imbalanced datasets because of its fixed baseline while PRC's baseline is determined by the ratio of positives and negatives while still retaining a 1:1 relationship with the points in the ROC curve. This is further proven by their re-evaluation of the MiRFinder study with ROC and PRC where the following results are displayed in **Figure 5 & 6**.

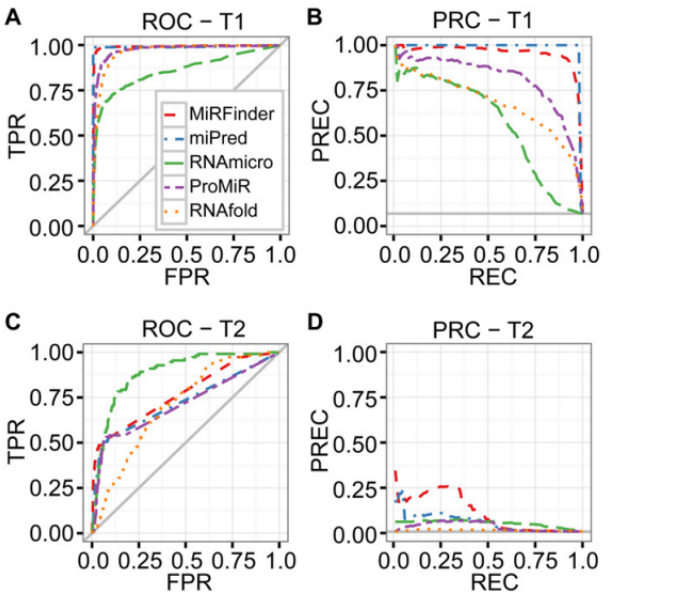


Fig 7. A re-analysis of the MiRFinder study reveals that PRC is stronger than ROC on imbalanced data. ROC and PRC plots show the performances of six different tools, MiRFinder (red), miPred (blue), RNAmicro (green), ProMiR (purple), and RNAfold (orange). A gray solid line represents a baseline. The re-analysis used two independent test sets, T1 and T2. The four plots are for (A) ROC on T1, (B) PRC on T1, (C) ROC on T2, and (D) PRC on T2.

doi:10.1371/journal.pone.0118432.g007

Figure 5: MiRFinder Study re-evaluation with ROC and PRC on balanced (A & B) and imbalanced (C & D) datasets.

Table 5. AUC scores of ROC and PRC for T1 and T2.

	T1 ROC	PRC	T2 ROC	PRC
MiRFinder	0.992*	0.945	0.772	0.106*
miPred	0.991	0.976*	0.707	0.024
RNAmicro	0.858	0.559	0.886*	0.054
ProMiR	0.974	0.801	0.711	0.035
RNAfold	0.964	0.670	0.706	0.015

Area under the curve (AUC) scores of ROC and PRC curves on datasets T1 and T2. The best AUC score in each column is marked with an asterisk (*).

doi:10.1371/journal.pone.0118432.t005

Figure 6: Resulting AUC scores for both balanced(T1) and imbalanced(T2) datasets

For parameter optimization, we utilized GridSearchCV and RandomizedSearchCV for experiment 1 (more information in section 5.2), and Weights and Biases (WandB) for neural networks in experiment 2 (5.3). WandB provides a tool for parameter optimization called “sweep”, which trains and evaluates models with randomly selected parameters set prior to network definition. An example of the parameters chosen to iteratively train the models is provided in **Figure 7**.

```
# W&B sweep configuration
sweep_configuration = {
    'method': 'random',
    'name': 'sweep',
    'metric': {
        'goal': 'maximize',
        'name': 'val_auc'
    },
    'parameters': {
        'batch_size': {'values': [16,32,64,128]},
        'epochs': {'values': [50,100,150,200]},
        'fc_layer_size': {'values': [7,9,11,13,15]},
        'sc_layer_size': {'values': [7,9,11,13,15]},
        'lr': {'max': 0.1, 'min': 0.0001},
        'optimizer':{'values': ['adam','SGD','RMSprop',
                                'nadam','Adadelta','Adagrad',
                                'Adamax','Ftrl']},
        'dropout': {'values': [0, 0.3, 0.5]},
     },

}
```

Figure 7: Wandb sweep configuration.

After the sweep is complete, the performance of each model can be visualized within the WandB user interface in the form of line plots, scatter plots, parameter importance plots, parallel coordinate plot, and more. Seen below in **Figure 8** is the parallel coordinate plot, illustrating the different parameter combinations that the sweep has iterated through, as well as and their validation AUC scores. Since the model is set to maximize this value, we will be choosing the top 2-3 models and training them independently in a NN outside of the sweep function and evaluate their accuracy with the AUC PRC score. Below the parallel coordinate plot is one of many evaluation metrics provided by WandB, the line plot, with interactive data visualization and separation with each sweep represented by varying colored markers.

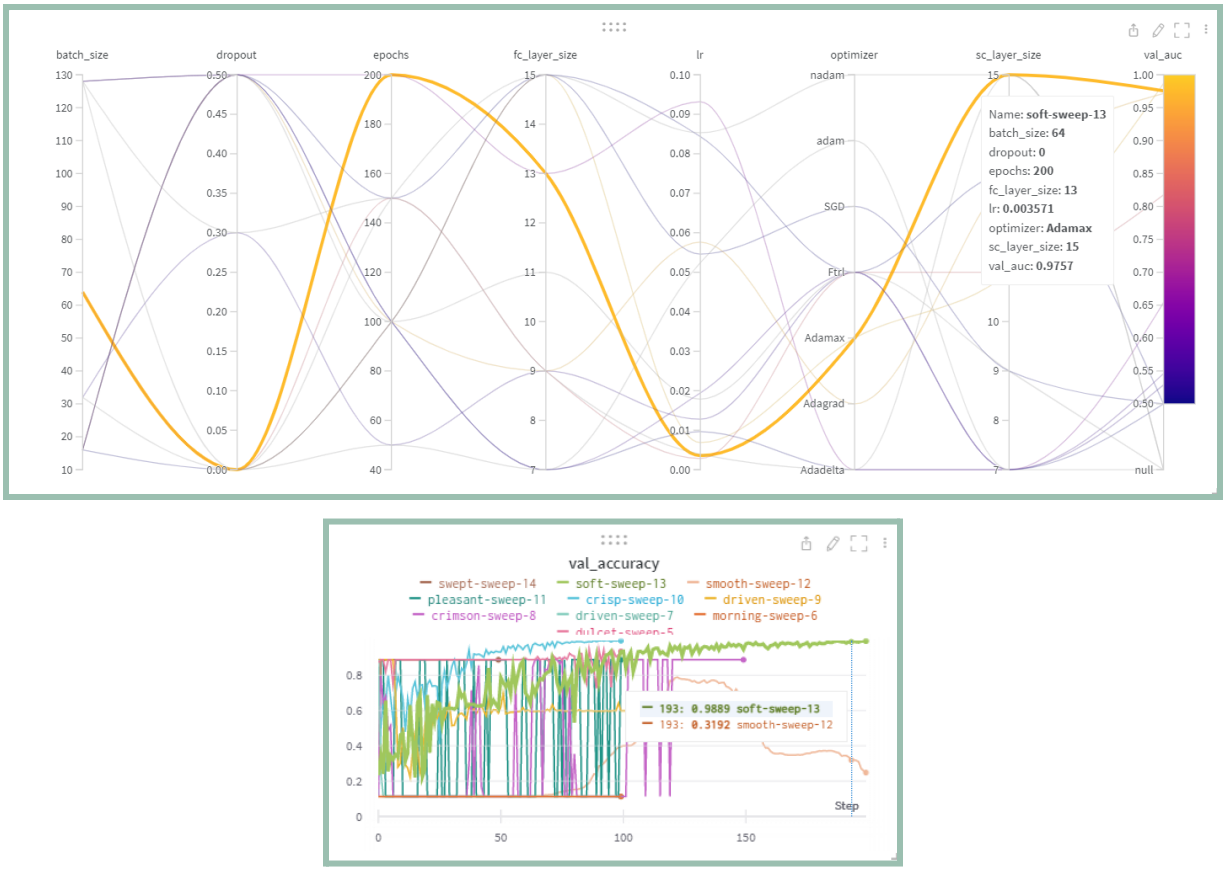


Figure 8: Parallel Coordinates Plot (top) and Validation Accuracy Line Plot (bottom)

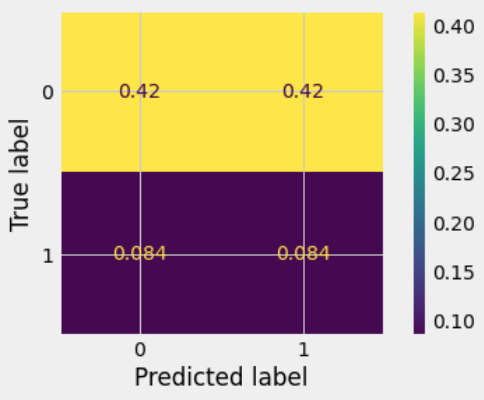
5 Experiments

5.1 Baseline Classifiers

In evaluating our classifiers in experiments 1 and 2, we compared our experimental models to our baseline models. Our baseline classifiers that we compared to included the following random methods provided in scikit-learn's DummyClassifier [15]: (1) uniform, (2) stratified and the following two constant methods: (1) constant, (2) most_frequent. Finally, we produced a confusion matrix for each selected model with scikit-learn performance metrics of average_precision_score(), f1_score(), precision_score(), recall_score(), and accuracy_score().

5.1.1 Uniform

The uniform strategy generates predictions at random in a uniform manner from the classes observed. Thus, the chance for predicting 0 “not-voting” or 1 “voting” is equal.



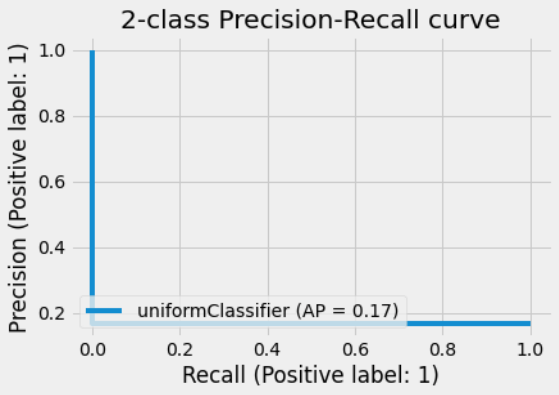


Figure 9: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Uniform Strategy

sklearn.dummy.DummyClassifier {"strategy": "uniform"}	
AUC-PR:	0.1685
F1 Score:	0.5617
Precision:	0.7198
Recall:	0.5000
Accuracy:	0.5000
MCC:	8.70622e-05

5.1.2 Stratified

The stratified strategy generates predictions by using the distributions of classes in the training dataset and outputting the majority class with that probability.

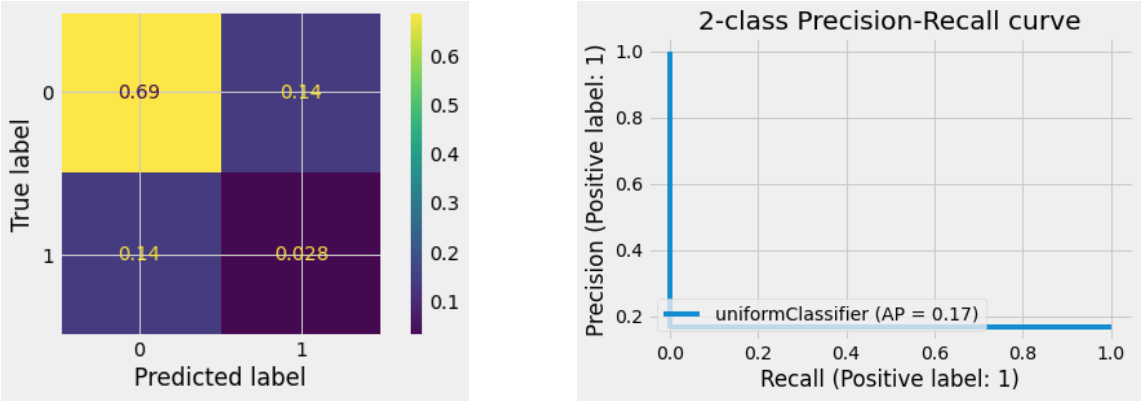


Figure 10: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Stratified Strategy

sklearn.dummy.DummyClassifier {"strategy": "stratified"}	
AUC-PR:	0.4992
F1 Score:	0.7180
Precision:	0.7198
Recall:	0.7201
Accuracy:	0.7201
MCC:	0.0004

5.1.3 Constant (non-voting)

The constant strategy always predicts the same constant label. For constant non-voting, the classifier always predicts 0 “non-voting”. Since the dataset is imbalanced, the accuracy score for this model is misleadingly high.

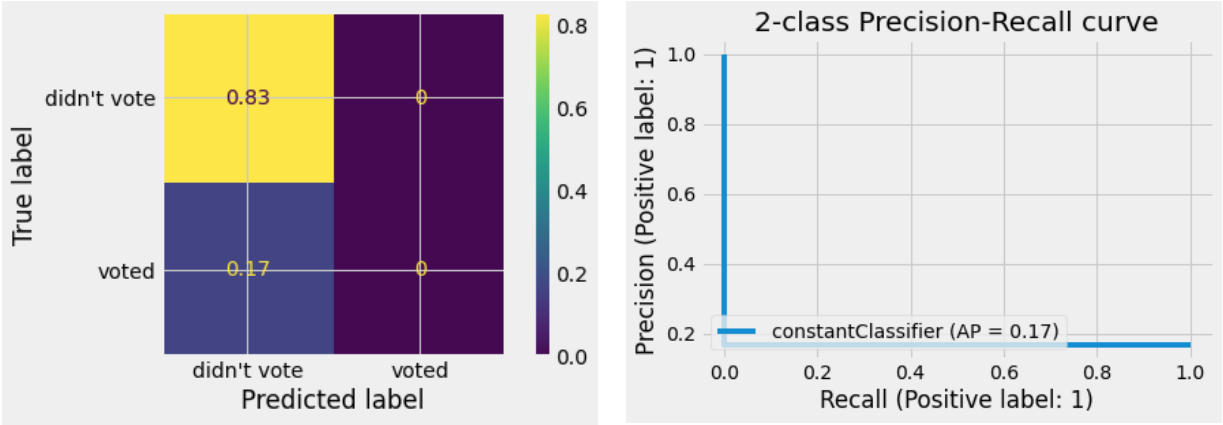


Figure 11: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Constant Non-voting Strategy

sklearn.dummy.DummyClassifier {"strategy": "constant", "constant": 0}	
AUC-PR:	0.1686
F1 Score:	0.7550
Precision:	0.6913
Recall:	0.8315
Accuracy:	0.8315
MCC:	0.0

5.1.4 Constant (voting)

The constant strategy always predicts the same constant label. For constant non-voting, the classifier always predicts 0 “non-voting”. Since the dataset is imbalanced, the accuracy score for this model is misleadingly high.

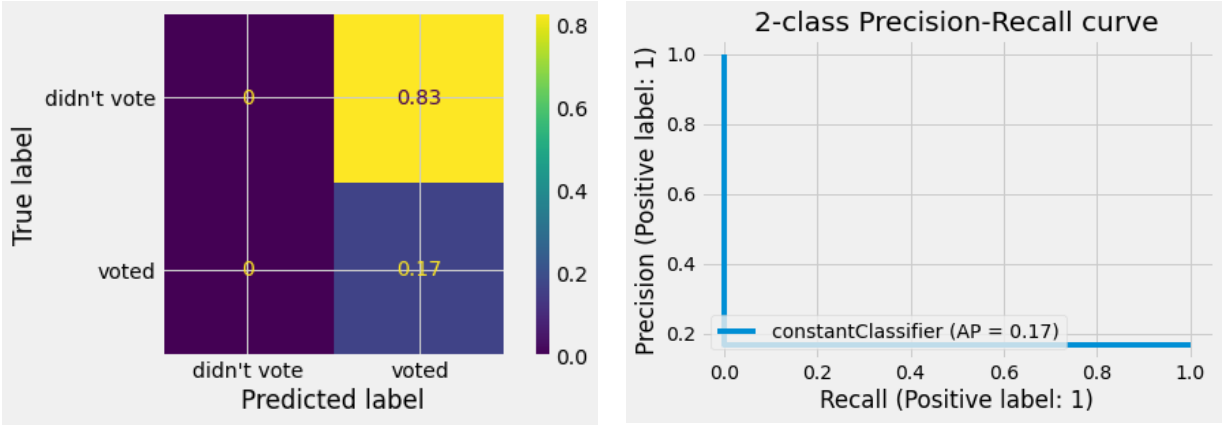


Figure 12: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Constant Voting Strategy

sklearn.dummy.DummyClassifier {"strategy": "constant", "constant": 1}	
AUC-PR:	0.1685
F1 Score:	0.0487
Precision:	0.0284
Recall:	0.1685
Accuracy:	0.1685
MCC:	0.0

5.1.5 Most Frequent

The most frequent strategy always predicts the most frequent label that has been observed. It is identical to constant (non-voting).

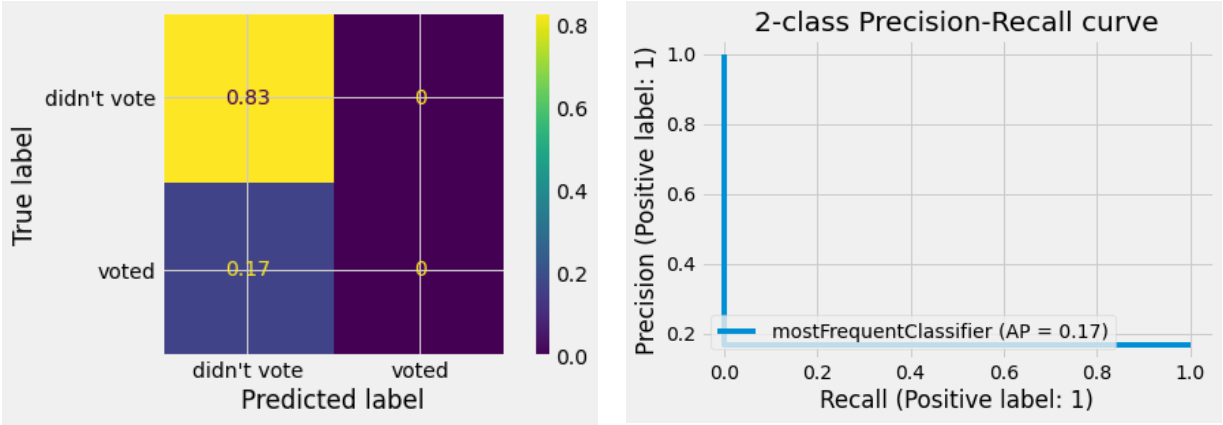


Figure 13: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Constant Non-voting Strategy

sklearn.dummy.DummyClassifier {"strategy": "most_frequent"}	
AUC-PR:	0.1686
F1 Score:	0.7550
Precision:	0.6913
Recall:	0.8315
Accuracy:	0.8315
MCC:	0.0

5.1.6 Prior

The prior strategy always predicts the most frequent label that has been observed. For probability, it returns the empirical class prior distribution.

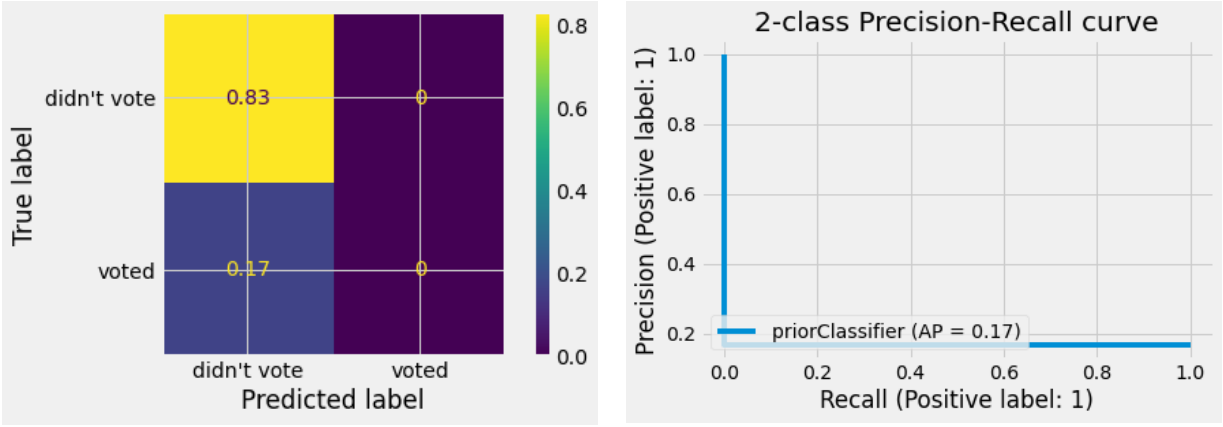


Figure 14: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for the Prior Strategy

sklearn.dummy.DummyClassifier {"strategy": "prior"}	
AUC-PR:	0.1686
F1 Score:	0.7550
Precision:	0.6913
Recall:	0.8315
Accuracy:	0.8315
MCC:	0.0

5.2 Experiment 1

For this experiment, the focus was on utilizing scikit-learn classifiers, including logistic regression, decision trees, random forests, and bagging estimators. The cleaned version of the dataset was used with no further transformations to the data.

5.2.1 Logistic Regression

A grid search with repeated 10-fold cross validation was conducted to find the best hyperparameters for the logistic regression model with L2 (ridge) regularization.

```
logreg = LogisticRegression()
solvers = ["lbfgs"]
penalties = ["l2"]
C_vals = [100, 10, 1.0, 0.1, 0.01]
max_iter = [1000]
weights = [None, "balanced"]
```

Figure 15: Hyperparameters Tested via GridSearchCV() for Logistic Regression

The best logistic regression model performed better than all of our baseline classifiers in regards to the average_precision_score(). The model, however, missed a substantial amount of voters due to the number of false negatives.

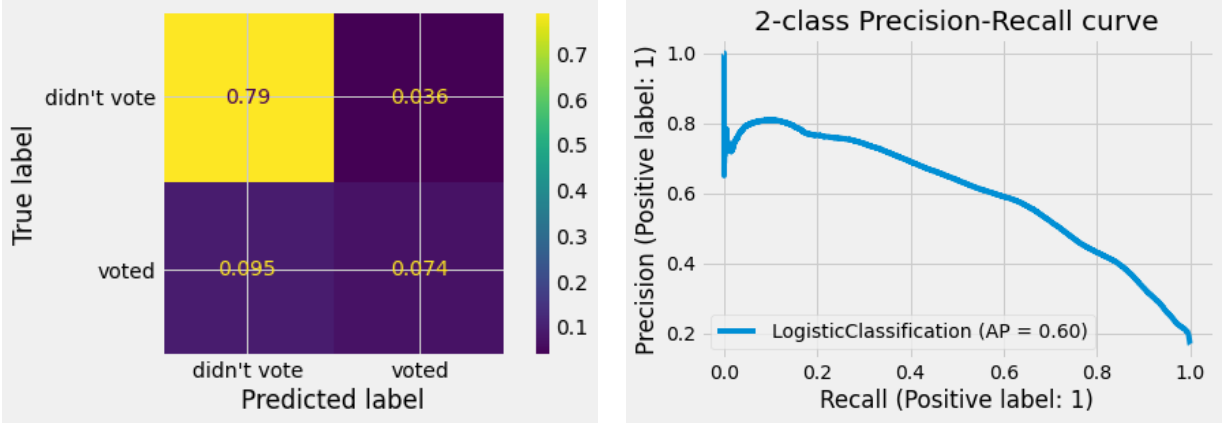


Figure 16: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for Logistic Regression

sklearn.linear_model.LogisticRegression {'C': 0.1, 'class_weight': None, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}	
AUC-PR:	0.6002
F1 Score:	0.8573
Precision:	0.8557
Recall:	0.8688
Accuracy:	0.8688
MCC:	0.4710

5.2.2 Decision Tree

A grid search with repeated 10-fold cross validation was conducted to find the best hyperparameters for the decision tree classifier.

```
decisionTree = tree.DecisionTreeClassifier()
criterion = ["gini", "entropy"]
class_weight = [None, "balanced"]
max_depth = np.arange(1, 10, 1).tolist()
```

Figure 17: Hyperparameters Tested via GridSearchCV() for DecisionTreeClassifier()

The best decision model performed better than all of our baseline classifiers in regards to the average_precision_score(). This model, unlike all of our prior models, identified a larger proportion of actual voters - although at the cost of a substantial rate of false positives.

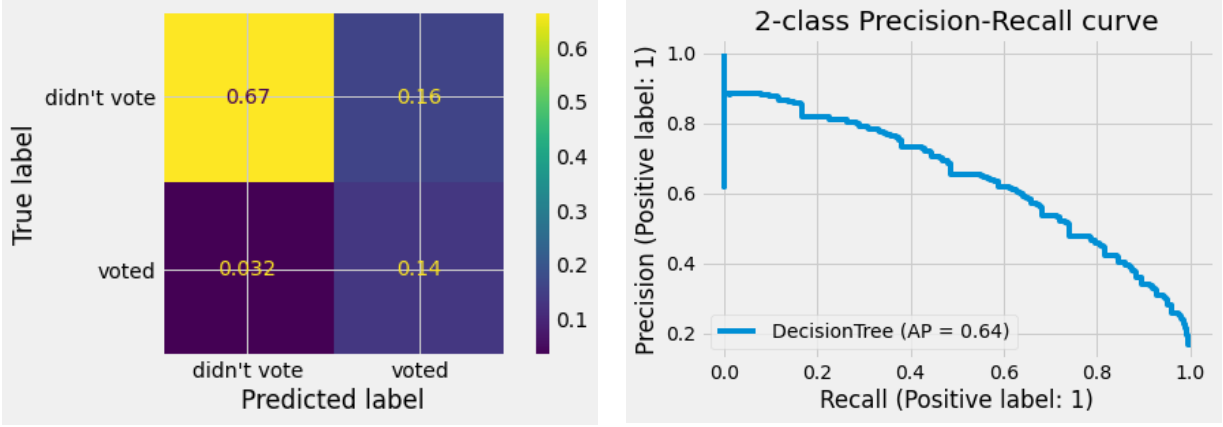


Figure 18: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for DecisionTreeClassifier()

sklearn.tree.DecisionTreeClassifier {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 9}	
AUC-PR:	0.6423
F1 Score:	0.8231
Precision:	0.8696

Recall:	0.8041
Accuracy:	0.8041
MCC:	0.4995

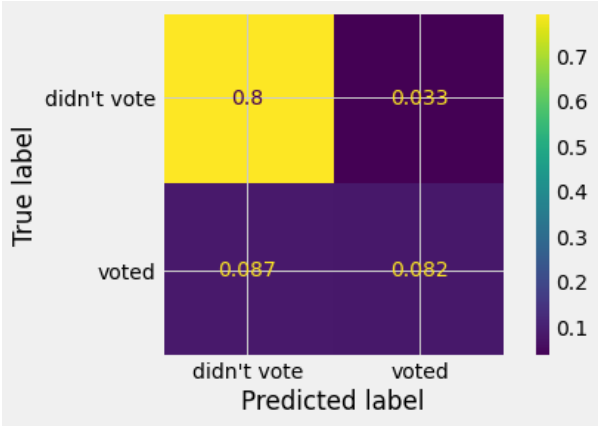
5.2.3 Random Forests

A randomized search with 5-fold cross validation was conducted to find the best hyperparameters for the random forest classifier. Our cross validation methodology was altered for this model due to training time.

```
randomForest = RandomForestClassifier()
n_estimators = np.arange(100, 2100, 100).tolist()
criterion = ["gini", "entropy"]
max_depth = np.arange(1, 20, 1).tolist()
max_features = ["sqrt", "log2"]
min_samples_leaf = np.arange(1, 5, 1).tolist()
min_samples_split = [2, 4, 5, 6, 8, 10]
```

Figure 19: Hyperparameters Tested via RandomizedSearchCV() for RandomForestClassifier()

The best random forest model performed better than all of our baseline classifiers in regards to the average_precision_score(). In addition, the model also outperformed all of the other models tested in experiment 1.



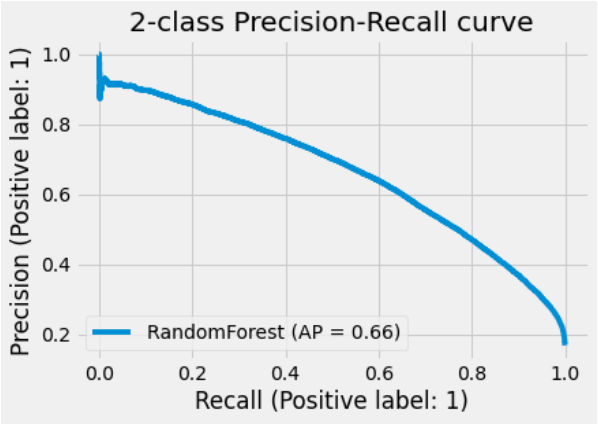


Figure 20: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for RandomForestClassifier()

sklearn.ensemble.RandomForestClassifier {'n_estimators': 2000, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 14, 'criterion': 'gini'}	
AUC-PR:	0.6635
F1 Score:	0.8704
Precision:	0.8697
Recall:	0.8799
Accuracy:	0.8799
MCC:	0.5221

5.2.3 Bagging Estimators

A grid search with repeated 10-fold cross validation was conducted to find the best hyperparameters for the random forest classifier.

```
baggingClassifier = BaggingClassifier()
n_estimators = np.arange(10, 100, 10).tolist()
max_samples = [0.05, 0.1, 0.2, 0.5]
```

Figure 21: Hyperparameters Tested via GridSearchCV() for BaggingClassifier()

The best bagging model performed better than all of our baseline classifiers in regards to the average_precision_score().

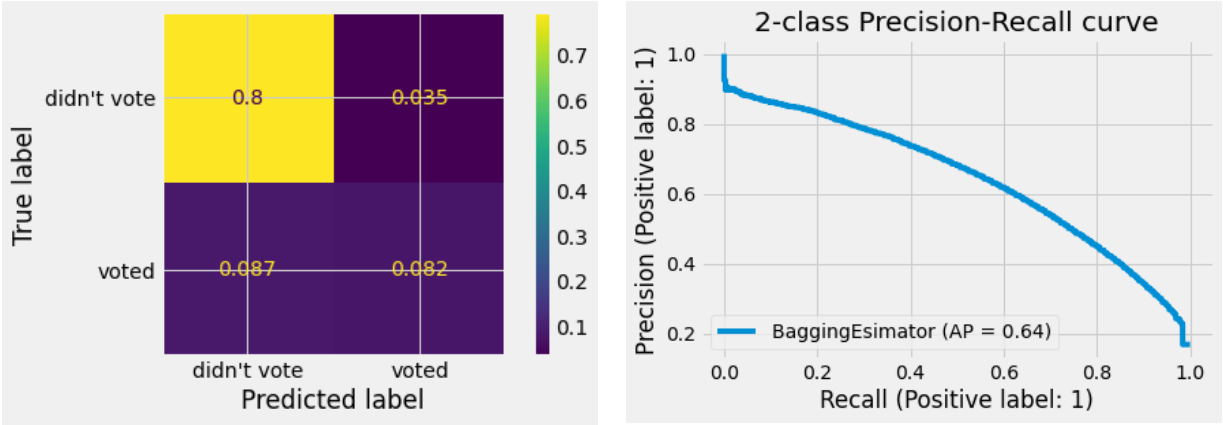


Figure 22: Confusion Matrix (left) and Area Under the Precision-Recall Curve Diagram (right) for BaggingClassifier()

sklearn.ensemble.BaggingClassifier {'n_estimators': 60, 'max_samples': 0.1}	
AUC-PR:	0.6409
F1 Score:	0.8680
Precision:	0.8663
Recall:	0.8767
Accuracy:	0.8767
MCC:	0.5127

5.3 Experiment 2

For this experiment, the focus was on ANN-based binary classification models for propensity and likely-party affiliations. The datasets used are the same as previous experiments with the exception of MinMaxScaler being applied to propensity models and strictly RandomUnderSampling to party affiliation models. Prior to building the NN, some data manipulation was required for binary classification on the party affiliation predictor. The data initially arrived as follows: "NaN" = 0 (did not vote), "U" = 1 (unaffiliated voter), "D" = 2 (Democrat), "R" = 3 (Republican), and "M" = 4

(Moderate). Since the aim is to predict the party affiliation of those who voted and our current political field mainly follows a two party system (Democrat and Republican), rows corresponding to classes "NaN", "U", and "M" are dropped from the data and our main predictors become Class 0 = Republican and Class 1 = Democrat. These models were constructed and implemented by Tensorflow's Keras framework, along with being evaluated using WandB sweeps and scikit-learn performance metrics. To save computation time at the start of this NN-building process, the sequential models were trained, tested, and optimized using the smaller, South Kingstown, dataset. After the model's parameters were optimized and chosen from WandB sweeps, the same NN framework was applied to the statewide prediction models. Since both models are using binary classification, ReLU (max(0,x)) activation functions are applied to the hidden and input layers along with a Sigmoid function assigned to the output layers and binary cross entropy for the loss function. As for the tunable parameters, the WandB sweep randomly trained and tested different batch sizes, epochs, learning rates, optimizers, and dropout rates, and the number of units per hidden layer all with the same goal of maximizing validation accuracy as seen in **Figure 7**. After sweeping was performed, the best parameters were as follows:

> *Propensity: {Batch_size = 128, Epochs = 100, Learning rate = 0.07087, Optimizer = AdaMax, Dropout rate = 0, Units in h1 = 15, and Units in h2 = 15}*
>
> *Party Affiliation: {Batch_size = 64, Epochs = 200, Learning rate = 0.003571, Optimizer = AdaMax, Dropout rate = 0, Units in h1 = 13, and Units in h2 = 15}*

The preferred optimizer for both model types, *AdaMax*, is a variant of the "Adam" optimizer, following Stochastic Gradient Descent with momentum, the L^2 norm based update rule is generalized to an $L^{infinity}$ norm based update rule [17]. After parameter tuning was completed, the models were trained, tested, and evaluated using the methods described in section 4.2.3 with the results presented in **Figures 9 & 10** below.

		Party Affiliation		Voter Propensity	
		SK	State	SK	State
Training					
	F1	0.9836	0.9528	0.8596	0.8048
	AUC_ROC	0.9769	0.9924	0.9258	0.8859
	Loss	0.1306	0.1047	0.3379	0.4232
	Accuracy	0.9838	0.9525	0.8526	0.8046
Validation (fit w/ test data)					
	F1	0.9721	0.8127	0.6488	0.5797
	AUC_ROC	0.9757	0.9924	0.9227	0.8863
	Loss	0.06682	0.134	0.3788	0.4302
	Accuracy	0.9937	0.9296	0.7996	0.8024

Figure 23: Resulting model performance table for training and test data

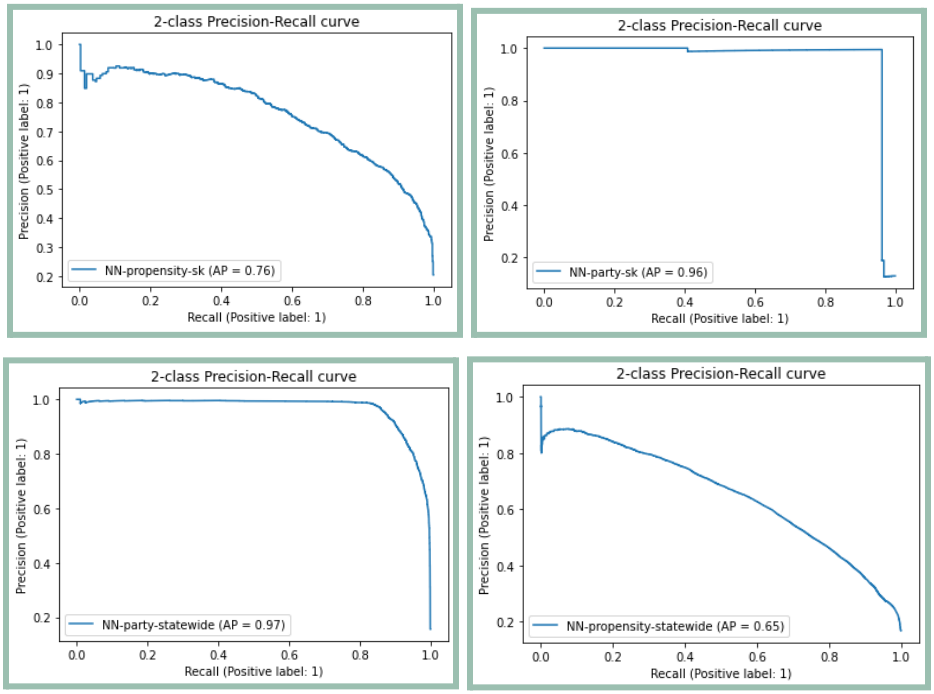


Figure 24: Area Under the Precision-Recall Curve diagrams

The table in **Figure 23** shows that the models are not overfitting when looking into training and test loss/accuracy, while also providing a glimpse into the generalizability of these NNs as they are applied to larger datasets (statewide vs local). As for the graphs following the table in **Figure 24**, it can be derived that the party affiliation models, with AP of 0.96 and 0.97 for SK and Statewide datasets respectively, perform very well whereas the remaining propensity models performed roughly on-par with the models used in Experiment 1. **Figures 25-28** below shows the resulting confusion matrices for each of the four models. These are accompanied by scikit-learn

performance metrics *F1*, *Precision*, *Recall*, and *Accuracy* along with the following configurations of prediction methods:

(a) *Undersampled Training Data Confusion Matrix with StratifiedKFold*,
(b) *Training Data Confusion Matrix*,
(c) *Undersample Training Data Confusion Matrix*, and
(d) *Test Data Confusion Matrix*.

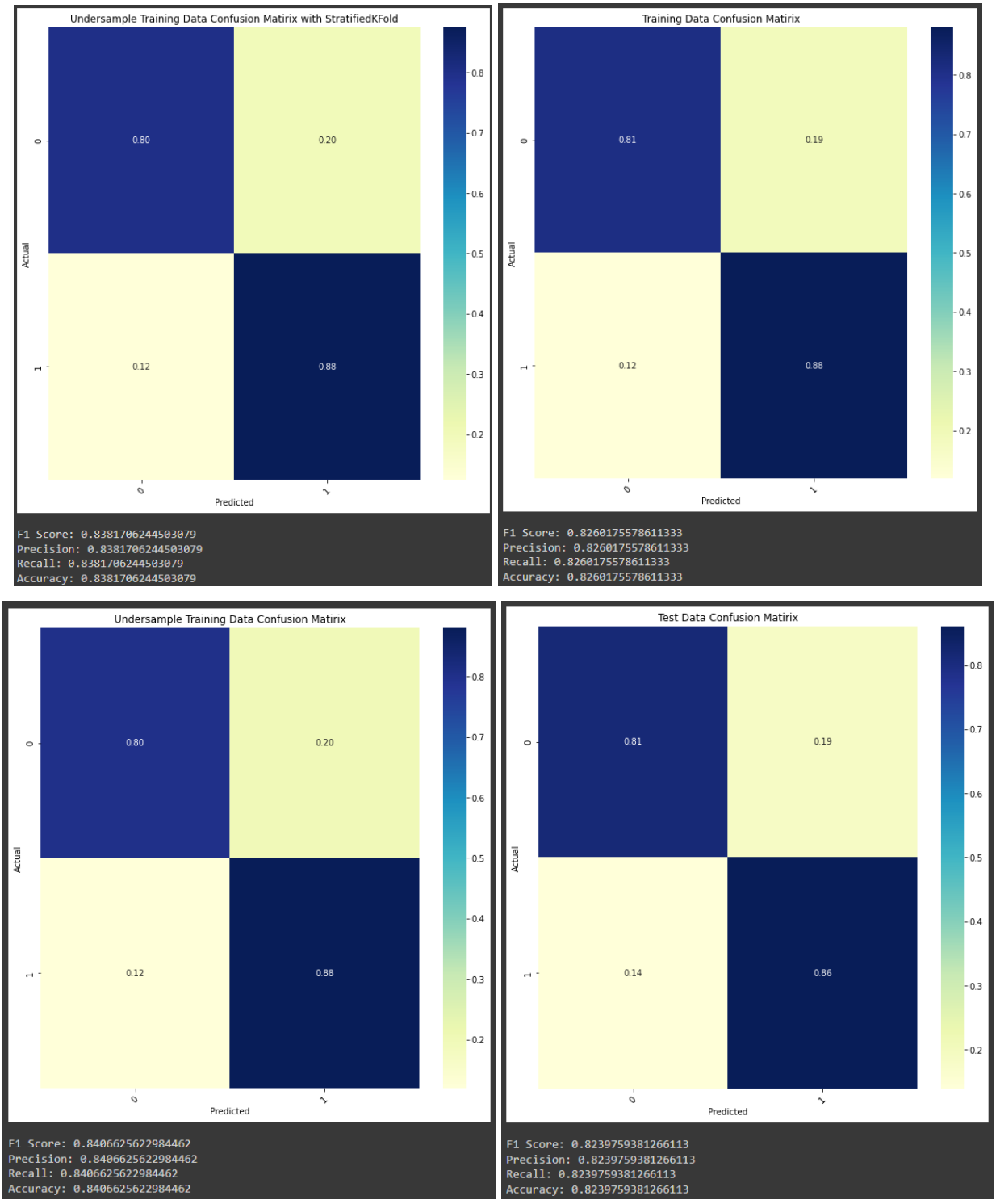


Figure 25: Propensity_SK confusion matrices with performance metrics.

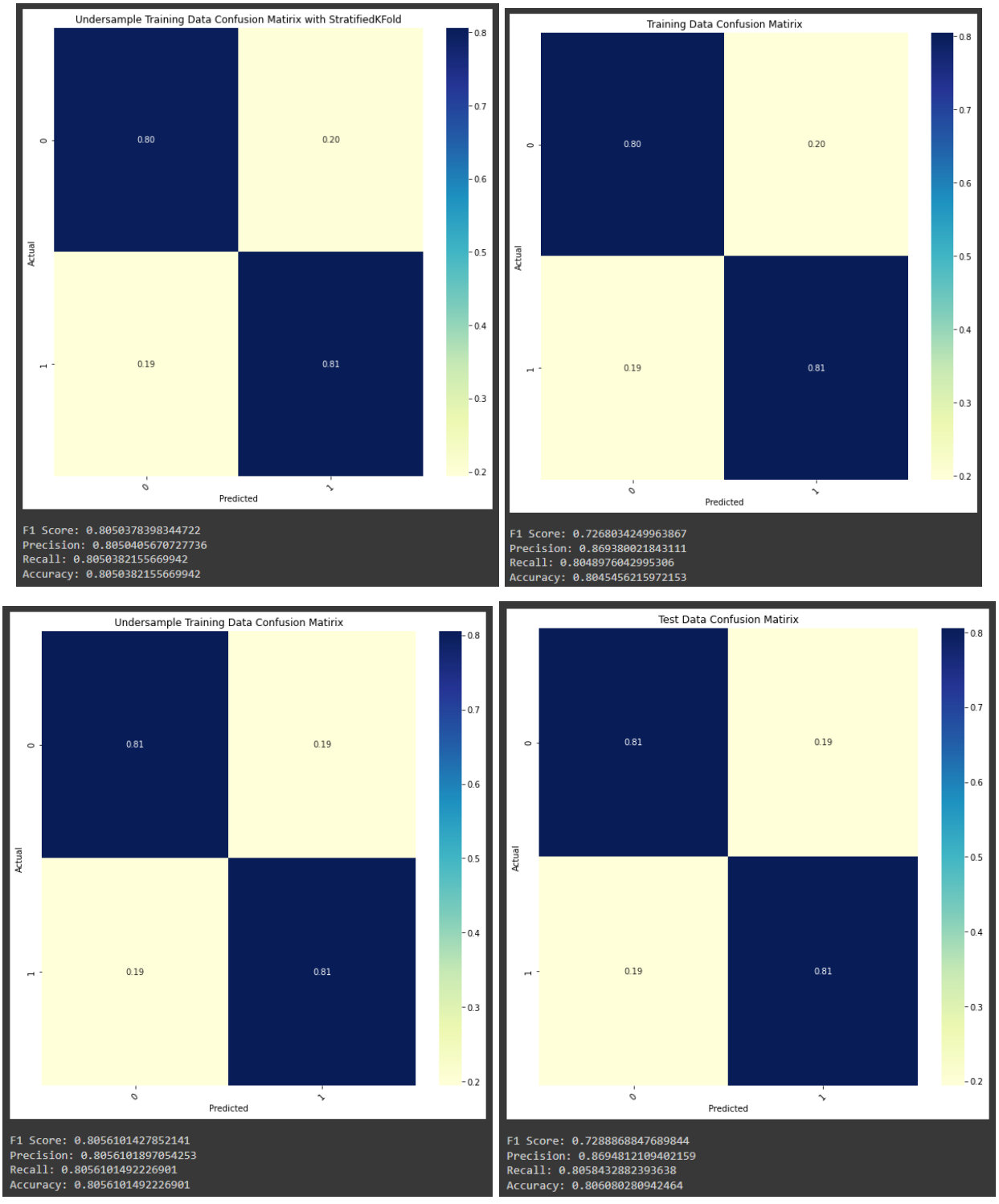


Figure 26: Propensity_State confusion matrices with performance metrics.

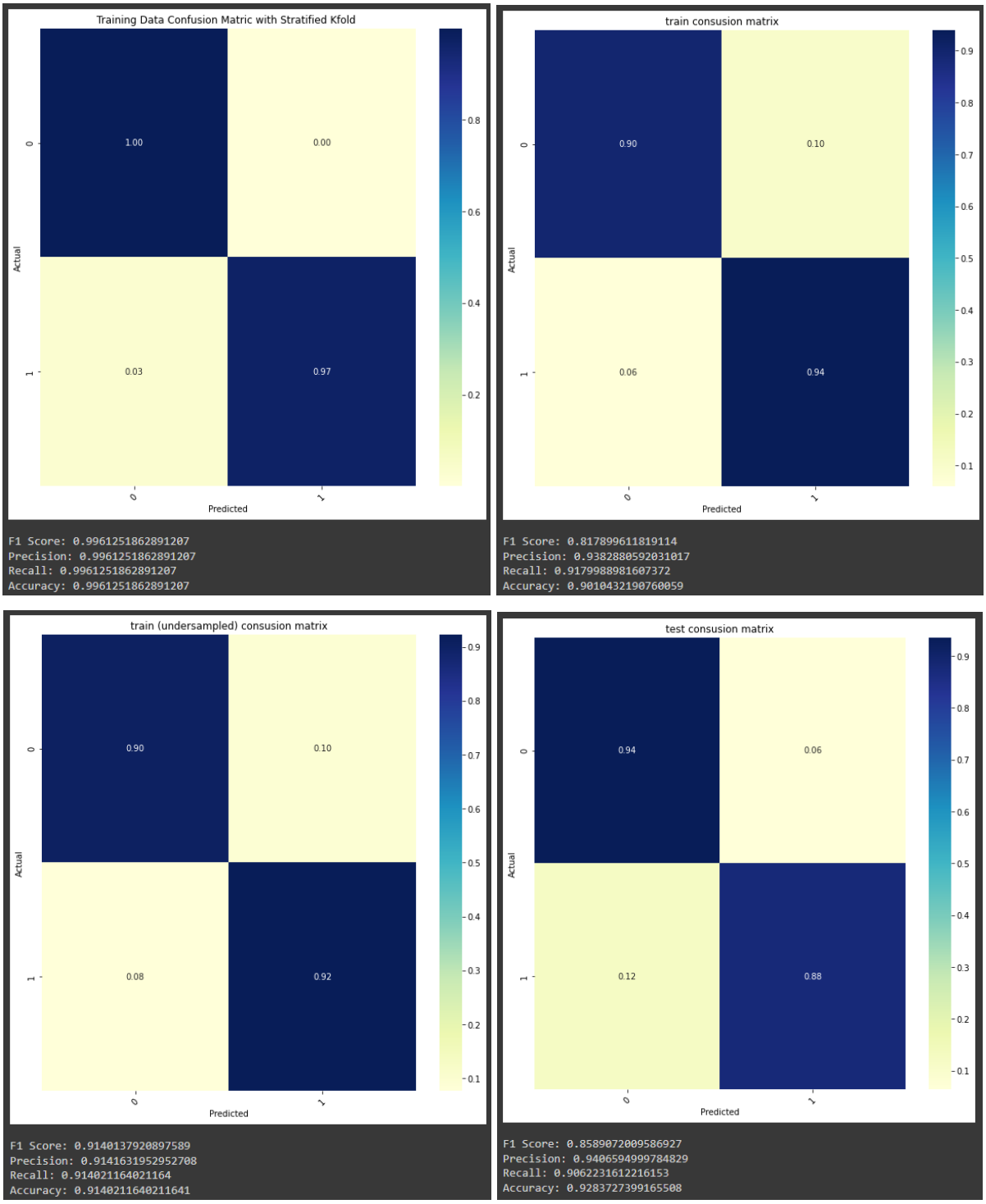


Figure 27: PartyAfilliation_SK confusion matrices with performance metrics.

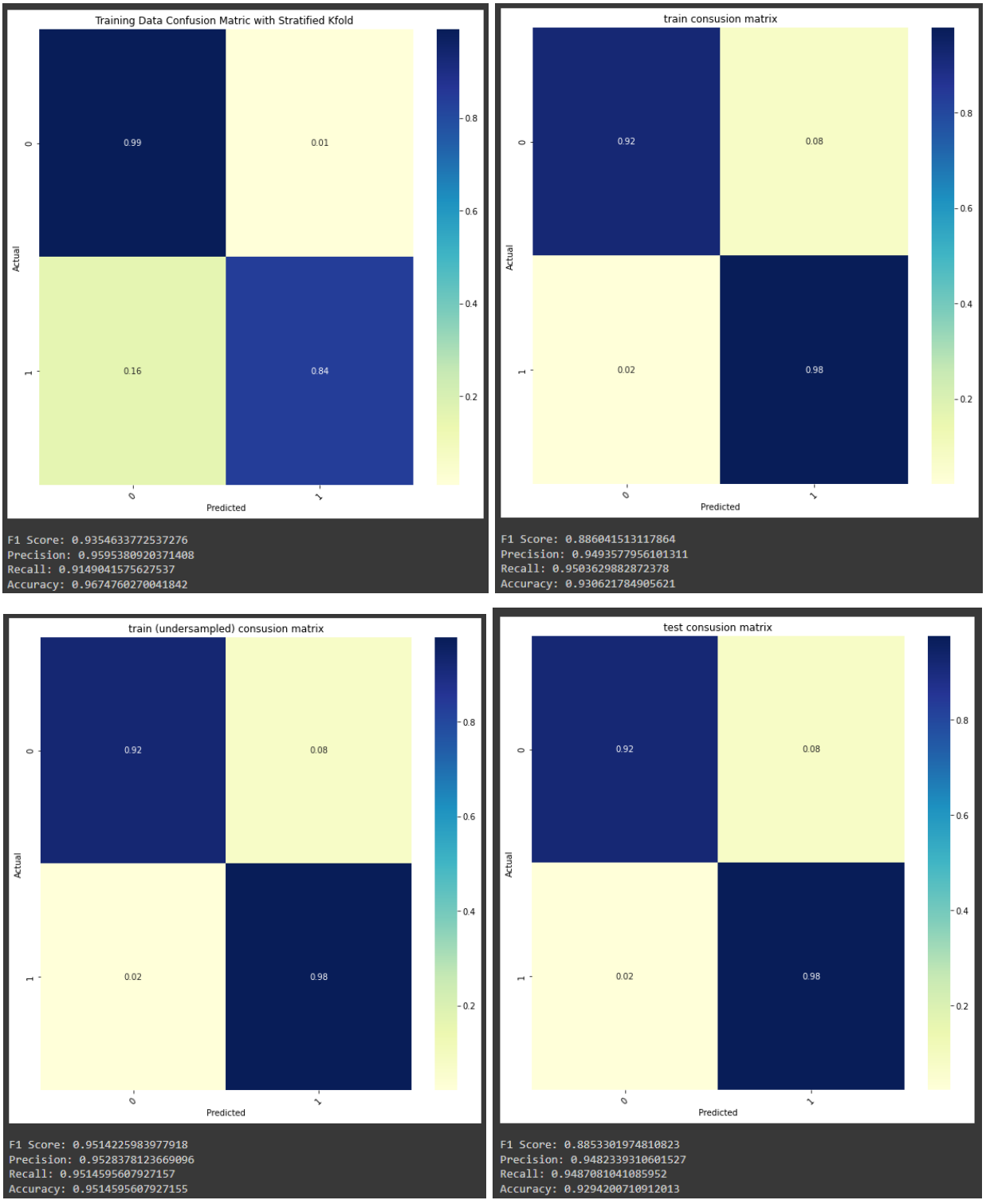


Figure 28: PartyAfilliation_State confusion matrices with performance metrics.

6 Discussion

In summation, our experiments successfully replicated findings in prior political science and statistical research showing that prior vote history is predictive of voting behavior. In addition, this research was able to extend prior findings to predicting voting behavior in primary elections, where the vast majority of voters do not participate in. In experiment 2, we successfully applied an artificial neural network to predict party affiliation in a partisan primary, achieving remarkable results with nearly a perfect model. Even with an imbalanced dataset, prior vote history with basic demographic variables provided reasonable results. Due to limitations in our study design, our findings represent a conservative attempt at predicting future voter behavior.

Our study has limitations that provide opportunities for future research. Due to time constraints, we were unable to test our experimental feature hypotheses and were limited to smaller WandB sweeps than we would prefer. Future research would consist of involving and evaluating the experimental features as presented in section 3 (to be described as "Experiment 3") and providing a more extensive parameter tuning process (e.g. using Optuna in conjunction with WandB for hyperparameter optimization).

The experiment shows that past voting history is predictive of future voting behavior. Building upon this finding in future research may allow the use of past vote history in elections as proxy features for future vote history; research should also analyze the performance of this suggested model with future primary elections. The use of incremental learning as new data becomes available is also a possibility. If the model is allowed to incrementally learn behavior as new data becomes available, the overall accuracy of the model may increase proportionally. Augmenting said dataset with additional public data, such as data from the U.S. Census Bureau, may assist with achieving this goal.

Acknowledgements: We thank Dr. Yichi Zhang for his suggestion regarding using AUC for model evaluation and Jimmy Dufurrena for his critical reading of earlier versions of this report.

REFERENCES

[1] Rusch, Lee, I., Hornik, K., Jank, W., & Zeileis, A. (2013). INFLUENCING ELECTIONS WITH STATISTICS: TARGETING VOTERS WITH LOGISTIC REGRESSION TREES. The Annals of Applied Statistics, 7(3), 1612–1639. https://doi.org/10.1214/13-AOAS648

[2] Denny, & Doyle, O. (2009). Does Voting History Matter Analysing Persistence in Turnout. American Journal of Political Science, 53(1), 17–35. https://doi.org/10.1111/j.1540-5907.2008.00355.x

[3] Costa, P., Nogueira, A.R., Gama, J. (2021). Modelling Voting Behaviour During a General Election Campaign Using Dynamic Bayesian Networks. In: Marreiros, G., Melo, F.S., Lau, N., Lopes Cardoso, H., Reis, L.P. (eds) Progress in Artificial Intelligence. EPIA 2021. Lecture Notes in Computer Science(), vol 12981. Springer, Cham. https://doi.org/10.1007/978-3-030-86230-5_41

[4] Pollard, R. D., Pollard, S. M., & Streit, S. (2021). Predicting Propensity to Vote with Machine Learning. arXiv preprint arXiv:2102.01535.

[5] Tynan Challenor. (2017). Predicting earnings from census data. Accessed: Jan. 16, 2021. [Online]. Available: http://cs229.stanford.edu/proj2017/finalreports/5232542.pdf.

[6] Hare, C., & Kutsuris, M. (2022). Measuring Swing Voters with a Supervised Machine Learning Ensemble. Political Analysis, 1-17. doi:10.1017/pan.2022.24

[7] Rogers, T., & Aida, M. (2014). Vote Self-Prediction Hardly Predicts Who Will Vote, and Is (Misleadingly) Unbiased. American Politics Research, 42(3), 503–528. https://doi.org/10.1177/1532673X13496453

[8] Bach, R. L., Kern, C., Amaya, A., Keusch, F., Kreuter, F., Hecht, J., & Heinemann, J. (2021). Predicting Voting Behavior Using Digital Trace Data. Social Science Computer Review, 39(5), 862–883. https://doi.org/10.1177/0894439319882896

[9] Availability of state voter file and confidential information. U.S. Election Assistance Commission. (2020, October 29). Retrieved November 16, 2022, from https://www.eac.gov/sites/default/files/voters/Available_Voter_File_Information.pdf

[10] *Publications and Forms*. Rhode Island Department of State. (n.d.). Retrieved November 16, 2022, from https://vote.sos.ri.gov/DataInformation/Publications

[11] Alvarez, Marco. "461-Fall-2022/11-Logistic-Regression.key.pdf at Main · Uri-CSC/461-Fall ..." *Github*, 7 Nov. 2022, https://github.com/URI-CSC/461-fall-2022/blob/main/lectures/11-logistic-regression.key.pdf.

[12] Vashisht, Raghav. "Machine Learning: When to Perform a Feature Scaling?" *Atoti*, 10 Oct. 2022, https://www.atoti.io/articles/when-to-perform-a-feature-scaling/.

[13] "Neural Network Tutorial: Step-by-Step Guide for Beginners." *UpGrad Blog*, 17 Oct. 2022, https://www.upgrad.com/blog/neural-network-tutorial-step-by-step-guide-for-beginners/.

[14] Rene Y. Choi, Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F. Chiang, J. Peter Campbell; Introduction to Machine Learning, Neural Networks, and Deep Learning. Trans. Vis. Sci. Tech. 2020;9(2):14. doi: https://doi.org/10.1167/tvst.9.2.14.

[15] Fenner, M. E. (2020). 6.2 Beyond Accuracy: Metrics for Classification. In *Machine Learning with Python for Everyone*. Pearson.

[16] Saito, Takaya & Rehmsmeier, Marc. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PloS one. 10. e0118432. 10.1371/journal.pone.0118432.

[17] Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.