Predicting Elections

Using the provided dataset, implement 4 different classification algorithms to determine if a candidate wins an election. For this assignment you may use the Pandas and Numpy library.

Data:

The attached csv file contains 1500 candidates and information about their campaign. The focus of this dataset is how financial resources affected the result of the campaign. A full description of the dataset is given here. For this assignment, you will use 5 features to predict if the candidate won or lost. For a given candidate, use the winner column (Y/N) as the target.

Features:

Feature	Tag	Type
Net Operating Expenditure	net_ope_exp	Numeric
Net Contributions	net_con	Numeric
Total Loan	tot_loa	Numeric
Candidate Office	can_off	P - President, S - Senate, H - House
Candidate Incumbent Challenger Open Seat	can_inc_cha_ope_sea	Incumbent, Challenger, Open

Implementation:

Each model will be tested in three steps. These functions are in the provided python file.

1. model.train(train_features, train_labels) for maximum of 1 minute
2. predictions = model.predict(test_features)
3. evaluate(predictions, test_labels)

K-Nearest Neighbor:

Use Euclidean distance between the features. For categorical data, convert each category to its own column with a value of 1 if the data is that category or otherwise a 0 (Called One Hot Encoding). Choose a k value and use majority voting to determine the class.

Perceptron & Multi-Layer perceptron (MLP):

As with KNN, use a one hot encoding of the input data. For the perceptron, multiply (dot multiply) the inputs by a weight matrix and then pass the output through a single sigmoid function to get the output. Don't forget the bias. For the MLP add at least one hidden layer. If you do everything in a modular fashion your code shouldn't get too complicated. Use a learning rate of .01 (1%) and the loss from the slides (Mean Squared Error).

Decision Tree:

Use the ID3 algorithm with a bucket size of 5.