Johann Ruiz

CS5644

Homework #2

Q1:

Introduction:

For homework 2, we are looking at the 1984 United States Congressional Voting Records Database. In this database, it shows what each congressional candidate voted for on different issues, ranging from immigration to religious groups in school. With this data, we want to see if we can predict which political party, democrat or republican, a candidate identifies as depending on their voting patterns. Normally, with politics, there are certain issues that each party stands for, therefore we can use supervised machine learning algorithms to classify between republican and democrat.

Experiment:

Data Processing:

The data came from UC Irvine Machine Learning Repository in two different files: house-votes-84.names and house-votes-84.data. The .names file contains the metadata, including the column names, and the .data contains all the rows in the dataset. The first step in the data processing process was gathering the columns names. While I could have manually typed all the columns from the .names file, I wanted to read the file using Python and extract the column names using regex pattern matching. This will help me automate extracting column names for future datasets. The next step was using the numpy python library to load the .data file and begin cleaning the data. The data is in string format, with the 'Class Name' column having either 'republican' or 'democrat' and every other column having with 'yes', 'no', or '?' values. To simplify the dataset, I created a numpy array where all the 'democrat' and 'no' was value was mapped to 0, ‘republican’ and ‘yes’ was mapped to 1, and all other values were mapped to –1.

By doing this conversion, I created the dataset that had a Unique Value for the ‘?’ fields with –1. Now that all these fields have –1, it made creating the other two datasets very easy. First, I iterated through the array and discarded any row that had a –1, creating the Discard Meaning dataset. Next, I used the scipy stats library to calculate the mode for every column in the Unique Value dataset, and loop through all rows and replace any instance of –1 with it’s mode. From here, I had all three datasets ready to be passed through the machine learning classifier models.

Models:

To classify the voting data, I used a Decision Tree Classifier model and Naive Bayes model. The Decision Tree Classifier model uses information gain, or entropy, to determine which feature provides the more insight on the final class and creates a decision tree by repeating this process. The Naive Bayes model uses conditional probability to determine the likelihood of each classification given each feature value and chooses higher likelihood between the classes.

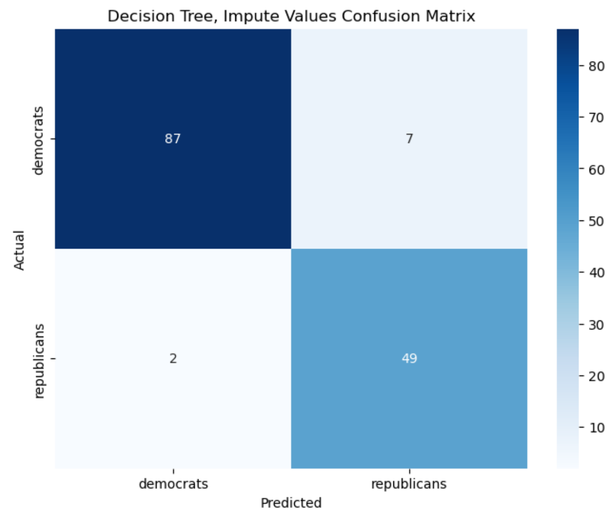
Results:

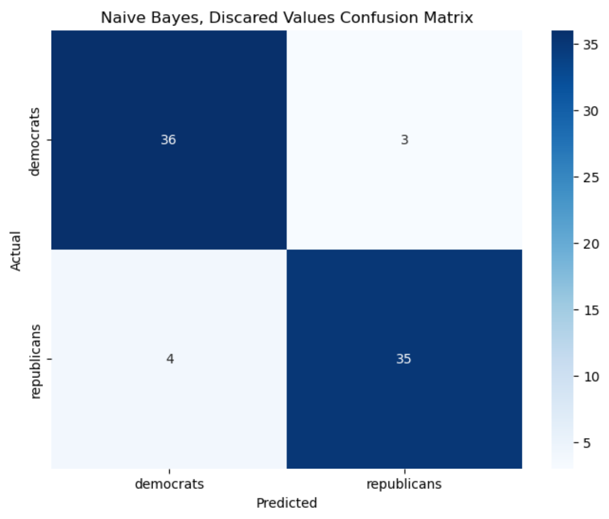
Below are the results from a 5-Fold cross validation using both the Naive Bayes and Decision Tree classifier models.

Model	Mean/StdDev of Precision	Mean/StdDev of Recall	Mean/StdDev of F1 Score
NB -- Discard Meaning	0.9190	0.9142	0.9142
NB -- Unique Value	0.9049	0.8965	0.8974
NB -- Impute Value	0.9108	0.9034	0.9042
DT -- Discard Missing	0.9535	0.9523	0.9523
DT -- Unique Value	0.9405	0.9379	0.9381
DT -- Impute Value	0.9547	0.9540	0.9540

The Decision Tree Classifier model produced higher precision and recall across each dataset than the Naive Bayes Classifier model. When looking at only the Decision Tree classifier results, one can observe a .1 dropoff in the values for the

Unique Value dataset. This result comes from introducing another type of value with ‘?’, which was converted into a –1 for processing. The model must take into consideration another variable for making decisions, which makes it harder to predict the class. It’s as simple understanding the more variables that can influence a classification, the more data the model will need to learn on different types of combinations. The metadata mentioned ‘?’ had relevant meaning, as it could have been used to avoid a conflict of interest in a vote, so introducing this variable does provide useful information. If we decide to impute ‘?’ with the most common value, then it can introduce a bias in the data since it’s defaulting to what most people vote on. However, if we simply discard rows with a ‘?’, then we have less data to train on. Below is a confusion matrix on the best performing datasets per model with 2/3 split on training and 1/3 on testing.





One can see that the Naive Bayes confusion matrix has significantly less data than was trained on, which shows that discarding rows with imperfect values isn’t always the best choice. The reason why the Decision Tree classifier performed better than the Naive Bates classifier is because the Decision Tree model is good as Boolean represented values, the ‘yes’ or ‘no’, this class or that class, datasets, while Naive Bayes is better for text classification.

Conclusion:

For this assignment, we processed the 1984 United States Congressional Voting dataset to determine if we can properly classify a congressman or congresswomen as a democrat or republican based on their votes on different bills. Since the data had a '?' option for certain votes, we explored three different ways of cleaning data: discarding any row with a question mark, converting the '?' to the most common vote for that bill, or leaving it as it is to introduce a new value. We used a Decision Tree classifier model and a Naive Bayes classifier model to learn from our data and conducted a 5-fold cross validation to determine the precision, recall, and f1 score of each model given each dataset. The results showed the Decision Tree classifier produced the best results compared to the Naive Bayes classifier, and the imputed dataset also performed the best, however we wanted to note that discarding rows or changing the values can add a bias to the data to be weary of.

Q2:

For what type of dataset would you choose decision trees as a classifier over Naive Bayes? Vice versa?

The decision tree classifier is useful for features with an exhaustive list of values, for example a car's color, true or false, yes or no, etc. Decision tree would also be used when the class is also discrete, so for Q1 situation whether the voter was democrat or republican. Naive Bayes classifier is very suitable for text classification datasets, in particular for determining spam. Naive Bayes is good for text classification datasets because it can handle high-dimensional data (IBM), which is good for document classification problems. Naive Bayes is also good for text classification because it's fast and easy to implement, and text datasets are very large since it deals with so many words, which makes it good for these.

https://www.ibm.com/topics/naive-bayes