Peter Brede
peterbrede@ucsb.edu
Perm #: 7052640
CS165A

MP1 Report

Architecture

My code consists of a lot of set up, including function definitions and list declarations in order to extract data from the training data, then transitions to the testing part which loops through each row of the testing data, predicting classes. In detail, this starts with the first section of my code, formatting the data. I used pandas.read_csv() in order to extract from the training data and testing and enter them into a numpy array. Then I clean the data in order to substitute boolean values ("no" and "yes") to '0's and '1's, as well as assign numeric values to each unique location and direction. I applied this to both the training and testing arrays. Next up on line 52, I created two arrays that each contained all of the rows of each class, either allRain: class where it rains tomorrow, or notRain: class where it doesn't rain tomorrow. Now I set up the probability distributions by creating functions of finding means, standard deviations. I put these means and standard deviations into important data lists for each class separately. I further use these values to find the single probabilities for each given value based on the statistics of the column. After these two probability-finding functions, I calculate a bunch of priors for the non-continuous columns, which are the locations and directions and rainToday, namely column 6,8,9, and 20. These probabilities are based on the count of rows with the same value given the class value. Finally, the testing loop iterates through each column (excluding 12) and multiplies each row's probability to the proper single probability of that column. I accumulated two probabilities by doing this, the probability that the new testing row was of class 0 or 1, and printed the classifier's prediction, whichever probability was more likely.

Preprocessing

All of my preprocessing is mentioned in the architecture portion of this report. To clarify, all data is imported and cleaned and placed into arrays to be further used by the rest of the program.

Model Building

This program uses the Naïve Bayes classifier in order to come up with estimates of probabilities that each column is a value given class 0 or class 1. There are functions that use the mean and standard deviation of the values of each column and predict the likelihood that a new value would be in each class. This is used to assign a class to future testing data.

Results

Final Accuracy: 0.7852

Final Runtime: 2.04s

Challenges

I faced a lot of challenges formatting and accessing elements of lists and arrays in Python, as I am more familiar with OOP in C++. This made the original process quite lengthy, and after I got the original build done I spent a ton of time tweaking and optimizing certain aspects of code.

Weaknesses

My method is most likely not very effective on other data sets, and is very oriented towards this particular classification problem. If this encountered another problem, it would not be able to be dynamic and provide an efficient solution for the new problem. Also, I chose to omit column 12 because I recognized better accuracy for both my testing data and for the private testing dataset using this modified column list. In order to overcome these problems, I could have created a more generalized class for classification to handle more situations.