The first approach that I implemented was the naïve bayes approach. In this approach, I calculated the posterior using the prior times the likelihood over the evidence. No explanation of libraries is needed for this approach because no major libraries were used (coded from scratch). Because the features are binary and not continuous, probabilities are not difficult to calculate and thus methods such as Gaussian Naïve Bayes is not necessary. For the prior, because we were not given the distribution of the classification in the testing set, I decided to estimate the classification probability of the testing set by using the distribution of the training set instead of assuming equiprobability of either class. Later, I tested found that doing so resulted in a slightly higher rank.

In this Naïve Bayes approach, I counted the frequencies of each feature and added them to a frequency map of either an active feature map or inactive feature map. With this, I was able to calculate the “prior” and “likelihood” values. The prior value was simply two values (one for the inactive class and one for the active class) that represented the percentage of the number of records classified as one class over the total records. However, with the imbalance of the distribution of classifications within the training set, I faced a decision to either keep this prior value true to the general Naïve Bayes approach or to anticipate and change the prior value to better match the classification distribution of the testing set. I ended deciding to keep the prior value as the original percentage explained above because I felt that I did not have enough information to confidently assume that the testing set would have a different distribution from the training set. For the likelihood values (active class likelihood and inactive class likelihood), I took the product of the probability of each feature given a class (active or inactive). With these prior and likelihood values, I multiplied the prior and likelihood for each class and compared them against each other to determine which classification has a higher probability.

During implementation, I was getting a lot of zero probabilities during the algorithm’s classification phase. This caused a lot of ties in class 0 and class 1 probabilities, all with values of zero. I found that a lot of the probabilities calculated to 0 due to the sheer sparsity of the available data. Because of this, I had to determine how to implement feature selection to heavily limit the amount of 0s that the calculation would receive. The first feature selection technique I decided to use was a simple threshold against any shared features between the active and inactive input set. If there were about the same amount of a given active feature in the active set as there is in the inactive set, then the feature would be removed from consideration. When put to the test, this technique did not yield a much higher score as I anticipated. With this, I did not think to consider normalization for this technique as I did not think it would prove useful for receiving a higher score anyway.

Because of the disappointing results from the implementation of the first feature selection technique, the second feature selection technique I decided to use on the Naïve Bayes model was an extreme measure to combat the sparsity of the input data and thus the amount of zero probabilities calculated. I removed all features that exclusively resided in either the active set or the inactive set. This yielded in safer results with ~85% of the classifications residing in the inactive/zero class. However, even with this, I ended with my highest F1 score for my Naïve Bayes approach of 0.57.

Neural Network

Input layer, one hidden layer, and one output layer. Hard sigmoid activation on the output layer. Using a binary accuracy metric to optimize the neural network. Added one more hidden dropout layer to prevent overfitting. Found a neuron count sweet spot of 300 for the hidden layer to minimize loss.