Logistic Regression

The c++ implementation of logistic regression is rather simple and leaves room for heavy optimization. Essentially at the heart of the algorithm is the gradient descent function which adjusts the weights continuously in a for loop for a fixed number of iterations. In this project, the weights seem to converge extremely fast, I initially was using 500,000 iterations for the weights. However, I soon discovered that at even 1000 iterations the weights converged to the same values. The logistic regression algorithm utilizes a lot of matrix multiplication, so I used an external library called eigen in order to help with that. Eigen allowed me to multiply and transpose matrices and vectors. After the gradient descent function, I have another function to convert the log odds to raw probabilities. Finally, I have another function which converts those raw probabilities into classifications. From the results of the predictions and the actual outcomes I then utilize a confusion matrix function which calculates and outputs a confusion matrix. The r implementation of logistic regression is done utilizing a combination of r code and c/c++ code. In order to boost the performance of the algorithm many computationally expensive operations are performed in c/c++ for performance reasons. In terms of run times the algorithm that I made in c++ actually performed significantly worse than the r version. I suspect that my algorithm, while working, is not efficient. It is also worth noting that in r, the variable was coded as a factor and log odds were given to each separate class within the factor. This differentiated from the c++ algorithm I made. I instead opted for the pclass to be an integer instead, which may lead to performance decrease. However, there was not a significant difference in performance, the c++ algorithm I created was slightly more accurate though. For measuring execution time I opted to use chrono in c++ and sys.time() in r. Both of these were only done directly on the training algorithms themselves. Text

Description automatically generatedA picture containing table

Description automatically generated

Text

Description automatically generated with low confidence

Table

Description automatically generated with medium confidence

Naïve Bayes

The Naïve Bayes algorithm in c++ was implemented utilizing three essential functions. One function for calculating the a priori probabilities. One for calculating the conditional probabilities of discrete variables and one for calculating the conditional probabilities of continuous variables. After all the conditional and a priori probabilities have been discovered, the Bayesian formula is applied in another function to get a two-dimensional vector that contains the raw probability for that particular observation of being in the survived or perished class. Finally, those raw probabilities are turned into classes by another function. A confusion matrix is then built in order to display useful statistics about the model. The runtime of both algorithms was almost identical. They were both extremely quick and moreover the both performed about the same. Each one of them performed better than both logistic regression models. Text

Description automatically generated

Text

Description automatically generated

A picture containing text, receipt

Description automatically generated