Sam Nosenzo

CS1571 – AI HW4: Programming

Naïve Bayes

Fold Statistics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | +SpamTrain | -SpamTrain | +SpamTest | -SpamTest |
| 1 | 1451 | 2230 | 362 | 558 |
| 2 | 1451 | 2230 | 362 | 558 |
| 3 | 1450 | 2231 | 363 | 557 |
| 4 | 1450 | 2231 | 363 | 557 |
| 5 | 1450 | 2230 | 363 | 558 |

|  |  |  |
| --- | --- | --- |
| Iteration | P(+SpamTrain) | P(+SpamTest) |
| 1 | .3942 | .3935 |
| 2 | .3942 | .3935 |
| 3 | .3939 | .3946 |
| 4 | .3939 | .3946 |
| 5 | .3940 | .3941 |

The **Probability Table** is located in the ProbabilityTable.xlsx excel document as it is too large to fit in the report.

**Results Table** (FP = false positive, FN = false negative, ERR = error)

|  |  |  |  |
| --- | --- | --- | --- |
| Fold 1 | FP: 0.05734767025089606 | FN: 0.16298342541436464 | ERR: 0.09891304347826087 |
| Fold 2 | FP: 0.05734767025089606 | FN: 0.13259668508287292 | ERR: 0.08695652173913043 |
| Fold 3 | FP: 0.05385996409335727 | FN: 0.14049586776859505 | ERR: 0.08804347826086957 |
| Fold 4 | FP: 0.05206463195691203 | FN: 0.19559228650137742 | ERR: 0.10869565217391304 |
| Fold 5 | FP: 0.05555555555555555 | FN: 0.18732782369146006 | ERR: 0.10749185667752444 |
| Average | FP: 0.05523509842152339 | FN: 0.163799217691734 | ERR: 0.09802011046593966 |

Each fold has practically the same ratio of spam to non-spam examples to train and test on. The ratio of spam to non-spam examples is around 39.4% spam to 60.6% non-spam examples. This makes the majority class the non-spam samples for each fold. So, a majority class predictor would be right 60.6% of the time, where it would just guess non-spam for each test sample. This is better than a random guesser which would be correct 50% of the time. In terms of how the rates affect the scores that the algorithm would output, it definitely affects each and every guess that the it makes because each conditional probability is multiplied by the overall probability that it is a certain class from the training set.

The results came out rather well given the <11% error on all folds, which is considerably better than both random guessing (50%) and the majority class predictor (39.4% error). This makes sense because NaiveBayes uses all of the features within the vector to train the predicator and associates their value with whether the vector was marked spam or not. Inherently it uses more information available than both majority class (which uses one value), and random guessing (which uses 0 information), and this information being used to create associations between the marking of the vector and its features allows it to be a much more accurate predictor than either of the two other aforementioned methods.

Also as a note, I set only the probabilities that were equal to zero in the probability table to 0.0014 and not everything less than 0.0014. So, there are probabilities in the probability table that are less than 0.0014. This shouldn’t make too much of a difference but is worth noting.